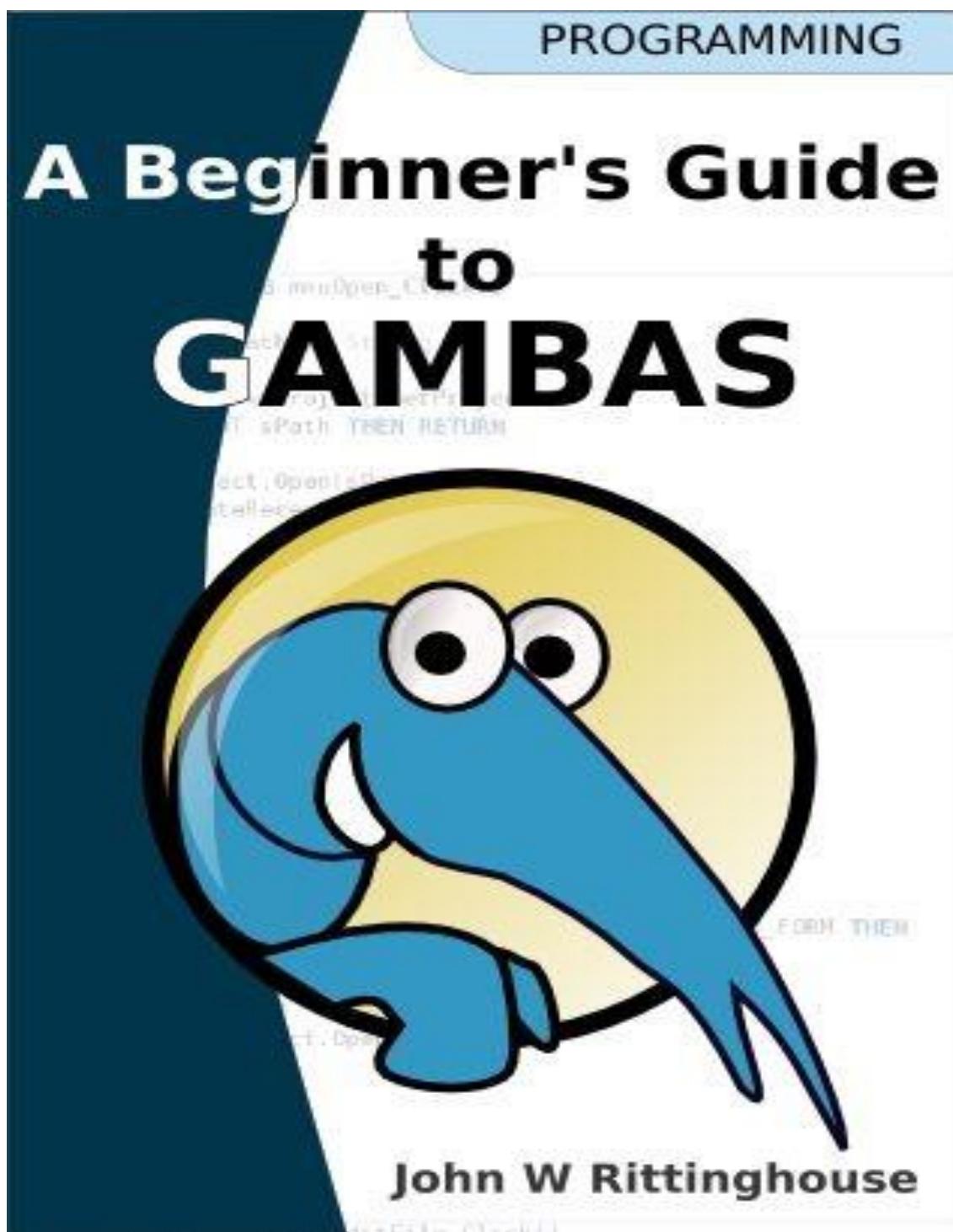


Una guida per principianti a



Cover design di Fabien Bodard

Prefazione di Fabien Bodard e Benoît Minisini

Una guida per principianti a

Avviso sul copyright per la versione stampata di questo lavoro:

Una guida per principianti a Gambas(quest'opera) è copyright © 2005 di John W. Rittinghouse, tutti i diritti sono riservati. È consentito l'uso personale di questo materiale. Tuttavia, il permesso di ristampare / ripubblicare questo materiale per scopi pubblicitari o promozionali o per creare nuove opere collettive per la rivendita o la ridistribuzione su server o elenchi, o per riutilizzare qualsiasi componente protetto da copyright di questo lavoro in altre opere deve essere ottenuto dall'autore, John W. Rittinghouse. L'autore concede una licenza perpetua alla comunità di utenti Gambas per l'uso della versione elettronica di questo lavoro stampato secondo i termini e le condizioni della Licenza OpenContent stampata nella pagina seguente.

This document is copyrighted by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed or modified without the express written consent of the author.

Una guida per principianti a

Avviso sul copyright per la versione elettronica (online) di quest'opera, basato sulla licenza OpenContent (OPL), versione 1.0, 14 luglio 1998.

Questo documento delinea i principi alla base del movimento OpenContent (OC) e può essere ridistribuito a condizione che rimanga inalterato. Per scopi legali, questo documento è la licenza con cui OpenContent è reso disponibile per l'uso. La versione originale di questo documento può essere trovata su <http://opencontent.org/opl.shtml>.

LICENZA

Termini e condizioni per la copia, la distribuzione e la modifica

Elementi diversi dalla copia, distribuzione e modifica del Contenuto con cui è stata distribuita questa licenza (come l'utilizzo, ecc.) non rientrano nell'ambito di questa licenza.

1. È possibile copiare e distribuire repliche esatte di OpenContent (OC) così come lo si riceve, con qualsiasi mezzo, a condizione di pubblicare in modo ben visibile e appropriato su ciascuna copia un avviso di copyright appropriato e una dichiarazione di non responsabilità della garanzia; conservare intatti tutti gli avvisi che si riferiscono alla presente Licenza e all'assenza di qualsiasi garanzia; e fornire a qualsiasi altro destinatario dell'OC una copia di questa Licenza insieme all'OC. Puoi, a tua discrezione, addebitare una commissione per i media e / o la gestione coinvolti nella creazione di una copia unica dell'OC da utilizzare offline, puoi a tua scelta offrire supporto didattico per l'OC in cambio di una commissione, o puoi opzione offerta garanzia in cambio di una tassa. Non puoi addebitare una commissione per l'OC stesso. Non è possibile addebitare una commissione per il solo servizio di fornire l'accesso e / o l'utilizzo del CO tramite una rete (ad esempio Internet),FTP o qualsiasi altro metodo.

2. Puoi modificare la tua copia o le copie di OpenContent o qualsiasi parte di esso, formando così lavori basati sul Contenuto e distribuire tali modifiche o lavori secondo i termini della Sezione 1 di cui sopra, a condizione che tu soddisfi anche tutte queste condizioni:

a) Devi fare in modo che il contenuto modificato includa avvisi in evidenza di cui l'hai cambiato, la natura e il contenuto esatti di le modifiche e la data di qualsiasi modifica.

b) Devi fare in modo che qualsiasi lavoro che distribuisce o pubblichi, che in tutto o in parte contenga o sia derivato dall'OC o da qualsiasi parte di esso, venga concesso in licenza nel suo insieme senza alcun costo a tutte le terze parti secondo i termini di questa Licenza, a meno che altrimenti consentito dalla legge sul Fair Use applicabile.

Questi requisiti si applicano all'opera modificata nel suo insieme. Se le sezioni identificabili di quell'opera non derivano dall'OC e possono essere ragionevolmente considerate opere indipendenti e separate di per sé, la presente Licenza e i suoi termini non si applicano a quelle sezioni quando vengono distribuite come opere separate. Ma quando si distribuiscono le stesse sezioni come parte di un tutto che è un'opera basata sull'OC, la distribuzione del tutto deve avvenire nei termini di questa Licenza, i cui permessi per gli altri licenziati si estendono all'intero intero, e quindi a ciascuno e ogni parte indipendentemente da chi l'ha scritta. Sono previste eccezioni a questo requisito per il rilascio gratuito di opere modificate ai sensi della presente licenza solo in conformità con la legge sull'uso corretto, ove applicabile.

3. Non sei obbligato ad accettare questa Licenza, poiché non l'hai firmata. Tuttavia, nient'altro ti concede il permesso di copiare, distribuire o modificare l'OC. Queste azioni sono proibite dalla legge se non si accetta questa Licenza. Pertanto, distribuendo o traducendo l'OC, o ricavandone i lavori, indichi la tua accettazione di questa Licenza per farlo, e tutti i suoi termini e condizioni per la copia, la distribuzione o la traduzione del CO.

NESSUNA GARANZIA

4. POICHÉ L'OPENCONTENT (OC) È LICENZIATO GRATUITAMENTE, NON ESISTE ALCUNA GARANZIA PER L'OC, NELLA MISURA CONSENTITA DALLA LEGGE VIGENTE. SALVO QUANDO ALTRIMENTI DICHIARATO PER SCRITTO I TITOLARI DEL COPYRIGHT E / O ALTRE PARTI FORNISCONO L'OC "COSÌ COM'È" SENZA ALCUN TIPO DI GARANZIA, ESPLICITA O IMPLICITA, INCLUSE, MA NON SOLO, LE GARANZIE IMPLICITE DI COMMERCIALIBILITÀ E IDONEITÀ PER UN PARTICOLARE . L'INTERO RISCHIO DI UTILIZZO DELL'OC È CON VOI. NEL CASO IN CUI IL CO DOVESSE RISULTARE DIFETTOSO, INACCURATO O IN ALTRO MODO INACCETTABILE, VI ASSUMETTE IL COSTO DI TUTTE LE RIPARAZIONI O CORREZIONI NECESSARIE.

5. IN NESSUN CASO A MENO CHE NON SIA RICHIESTO DALLA LEGGE APPLICABILE O NON SIA ACCETTATO PER SCRITTO QUALSIASI TITOLARE OR UNY O ILR PARTEY WHO MAY MIRROR UND / OR REDISTRIBUTE THE OC UNS CONSENTITO UNBOVE, ESSERE RESPONSABILE PER DANNI, COMPRESI QUALSIASI DANNO GENERALE,

Una guida per principianti a
SPECIALE, INCIDENTALE O CONSEQUENZIALE DERIVANTE DALL'USO O DALL'INCAPACITÀ DI UTILIZZARE IL CO,
ANCHE SE TALE TITOLARE O ALTRA PARTE È STATA AVVISATA DELLA POSSIBILITÀ DI TALI DANNI.

Una guida per principianti a

Sommario

| | |
|--|-----------|
| Ringraziamenti | 16 |
| Prefazione..... | 17 |
| Capitolo 1 Presentazione di Gambas..... | 19 |
| Architettura Gambas | 19 |
| L'ambiente di programmazione Gambas | 22 |
| Componenti IDE di Gambas..... | 25 |
| Capitolo 2 - Concetti linguistici di Gambas..... | 30 |
| Variabili, tipi di dati e costanti di Gambas | 30 |
| Assegnazione variabile..... | 35 |
| Assegnazione utilizzando l'istruzione WITH..... | 35 |
| Operatori ed espressioni | 36 |
| Operatori di confronto | 36 |
| Operatori aritmetici..... | 36 |
| Iniziamo a programmare Gambas | 37 |
| Dichiarazioni END, RETURN e QUIT..... | 37 |
| Operatori di stringa..... | 44 |
| Capitolo 3 Parole chiave e controllo del flusso del programma | 46 |
| L'istruzione PRINT | 46 |
| La dichiarazione IF | 47 |
| L'istruzione SELECT / CASE | 48 |
| GOTO ed ETICHETTE | 49 |
| L'istruzione FOR / NEXT | 49 |
| FARE [MENTRE] LOOP | 51 |
| WHILE [Expression] WEND Loop..... | 52 |
| Il ciclo REPEAT UNTIL | 53 |
| Definizione e utilizzo di array in Gambas..... | 53 |
| Collezioni | 55 |
| La dichiarazione FOR EACH | 55 |
| Capitolo 4 - Presentazione di Gambas ToolBox..... | 57 |
| Il controllo dei pulsanti | 61 |
| Proprietà di controllo comuni..... | 62 |
| Metodi dei pulsanti | 70 |

| | |
|--|------------|
| Una guida per principianti a | |
| Eventi pulsante | 78 |
| La classe delle immagini | 78 |
| Capitolo 5 - Controlli per la raccolta dell'input | 80 |
| TextLabel | 82 |
| Casella di testo | 83 |
| Combo box | 85 |
| ListBox | 89 |
| Telaio | 92 |
| Interruttore..... | 93 |
| Casella di controllo..... | 93 |
| Pannello | 95 |
| RadioButton | 95 |
| Capitolo 6 - Menu, moduli, finestre di dialogo e finestre di messaggio..... | 97 |
| L'editor del menu Gambas | 98 |
| Menu di costruzione | 101 |
| Dialoghi | 103 |
| Moduli..... | 105 |
| MessageBoxes..... | 112 |
| Messaggi di informazione..... | 112 |
| Interroga / Conferma messaggi | 114 |
| Messaggio di errore | 115 |
| Messaggi di avviso o di avviso..... | 115 |
| Elimina messaggi | 116 |
| Funzioni relative ai file della classe di dialogo | 117 |
| Funzione di dialogo OpenFile | 117 |
| Finestra di dialogo Funzione SaveFile | 118 |
| Finestra di dialogo SelectDirectory Function | 119 |
| Elenco completo di esempio | 120 |
| Modulo1. Elenco del modulo | 123 |
| Capitolo 7 - Gestione delle stringhe e conversione dei tipi di dati | 125 |
| Funzioni di stringa | 125 |
| Len..... | 126 |

| | |
|---|------------|
| Una guida per principianti a | |
| Superiore \$ / Ucase \$ / Ucase e inferiore \$ / Lcase \$ / Lcase | 126 |
| | |
| Trim \$, LTrim \$ e RTrim \$ | 127 |
| Left \$, Mid \$ e Right \$ | 128 |
| Spazio \$ | 130 |
| Sostituisci \$ | 130 |
| Stringa \$ | 131 |
| Subst \$ | 131 |
| InStr | 132 |
| RInStr | 134 |
| Diviso | 134 |
| Conversione dei tipi di dati | 135 |
| Asc e Chr \$ | 135 |
| Bin \$ | 136 |
| CBool | 137 |
| CByte | 138 |
| CDate | 138 |
| CFloat | 139 |
| CInt / Cinteger e CShort | 140 |
| CStr / CString | 140 |
| Hex \$ | 141 |
| Conv \$ | 141 |
| Val e Str \$ | 142 |
| Str \$ | 142 |
| Formato \$ | 145 |
| Gestione dei tipi di dati | 147 |
| Tipo di | 148 |
| Capitolo 8 - Utilizzo dei controlli avanzati | 149 |
| Controllo IconView | 149 |
| Controllo ListView | 158 |
| Utilizzo dello strumento di modifica delle icone di | |
| Gambas | 162 |
| Il controllo TreeView | 163 |
| Il controllo GridView | 171 |
| Il controllo ColumnView | 175 |
| Controlli del layout: HBox, VBox, HPanel e Vpanel ... | 177 |
| HBox e VBox | 177 |

| | |
|---|------------|
| Una guida per principianti a HPanel e Vpanel | 177 |
| Il controllo TabStrip | 182 |
| Capitolo 9 - Lavorare con i file..... | 188 |
| Accesso | 188 |
| Dir | 189 |
| Eof | 190 |
| Esistere | 190 |
| IsDir / Dir? | 191 |
| statistica..... | 191 |
| Temp / Temp \$ | 192 |
| APRIRE e CHIUDERE | 192 |
| INGRESSO DI LINEA | 193 |
| LEGGI, CERCA, SCRIVI e RISCIACQUA | 194 |
| COPY, KILL e RENAME..... | 195 |
| MKDIR, RMDIR | 196 |
| Capitolo 10 - Operazioni matematiche | 215 |
| Precedenza delle operazioni..... | 215 |
| Addominali | 216 |
| Acs / ACos | 216 |
| Acsh / ACosh | 217 |
| Asn / ASin | 217 |
| Asnh / ASinh | 218 |
| Atn / ATan | 218 |
| Atnh / ATanh | 219 |
| Cos | 219 |
| Cosh..... | 220 |
| Deg e Rad..... | 221 |
| Exp | 221 |
| Fix e Frac | 222 |
| Int..... | 223 |
| Log | 223 |

| | |
|---|------------|
| Una guida per principianti a | |
| Log10..... | 224 |
| | |
| Max e Min | 224 |
| Pi..... | 225 |
| Randomizza e Rnd | 225 |
| Il giro | 227 |
| Sgn | 227 |
| Peccato..... | 228 |
| Sinh | 229 |
| Mq..... | 229 |
| Tan | 230 |
| Tanh | 231 |
| Funzioni matematiche derivate | 231 |
| | |
| Capitolo 11 - Concetti orientati agli oggetti..... | 237 |
| Fondamenti di programmazione orientata agli oggetti..... | 238 |
| Oggetti..... | 239 |
| Astrazione dei dati..... | 239 |
| Incapsulamento | 240 |
| Polimorfismo | 240 |
| Eredità | 240 |
| L'approccio di Gambas all'OOP | 241 |
| Classi di Gambas | 241 |
| Programma di esempio: contatti..... | 242 |
| La classe Contact..... | 242 |
| Metodo Contact.GetData | 243 |
| Metodo Contact.PutData | 246 |
| File Form1.class | 247 |
| Costruttore Form1 | 248 |
| Form_Open Subroutine | 250 |
| Aggiunta di controlli a Form1.Form | 250 |
| I ToolButtons | 251 |
| Il pulsante Esci | 251 |
| Aggiunta di etichette e caselle di testo | 252 |
| Subroutine UpdateForm ()..... | 253 |
| Tasti degli strumenti di codifica: primo, precedente, successivo e ultimo..... | 254 |
| Pulsanti degli strumenti di codifica: aggiunta di un record | 256 |

| | |
|--|------------|
| Una guida per principianti a | |
| Pulsanti degli strumenti di codifica: cancellazione dei dati..... | 258 |
| Convalida dell'input dell'utente..... | 258 |
| Aggiunta di una funzione di ricerca | 260 |
| La subroutine DoFind | 262 |
| Di nuovo ToolButtons: Aggiornamento di un record | 263 |
| Di nuovo i pulsanti degli strumenti: eliminazione di un record | 264 |
| Di nuovo ToolButtons: Salvataggio dei dati | 265 |
| Creazione di un eseguibile autonomo | 266 |
| Capitolo 12 - Imparare a disegnare | 267 |
| Proprietà di disegno | 267 |
| BackColor / Background e ForeColor / Foreground..... | 267 |
| Clip..... | 268 |
| FillColor, FillStyle, FillX, FillY | 268 |
| Font..... | 269 |
| Invertire | 269 |
| LineStyle / LineWidth | 270 |
| Trasparente..... | 270 |
| Metodi di disegno..... | 270 |
| Text / TextHeight / TextWidth..... | 272 |
| Disegna primitive: Punto / Retto / Ellisse / Linea | 274 |
| Disegna primitive: poligono e polilinea | 280 |
| Immagine / Immagine / Piastrella | 284 |
| Disegnare con un oggetto Drawing | 292 |
| Capitolo 13 - Gestione degli errori | 298 |
| Concetti generali di gestione degli errori..... | 298 |
| Gestione degli errori..... | 298 |
| BoundaryRelated Errori | 299 |
| Errori di calcolo | 299 |
| Stati iniziali e successivi | 300 |
| Errori di flusso di controllo | 300 |
| Errori nella gestione o interpretazione dei dati | 301 |
| Condizioni di gara e carico | 301 |
| Problemi di piattaforma e hardware..... | 302 |
| Errori di controllo di origine, versione e ID | 303 |
| Errori di test..... | 303 |
| Revisioni del piano di test..... | 303 |

| | |
|--|------------|
| Una guida per principianti a | |
| Gestione degli errori di Gambas | 304 |
| Istruzione TRY ... IF ERROR | 304 |
| Dichiarazioni di cattura e infine | 306 |
| Gestione eventi Gambas | 309 |
| Capitolo 14 - Mouse, tastiera e operazioni sui bit..... | 312 |
| Operazioni con il mouse..... | 312 |
| Operazioni da tastiera | 316 |
| Operazioni sui bit | 318 |
| Capitolo 15 - Gambas e database..... | 326 |
| Classe di connessione..... | 327 |
| Proprietà di connessione | 328 |
| Charset..... | 328 |
| Banche dati..... | 328 |
| Ospite..... | 329 |
| Login..... | 329 |
| Nome..... | 329 |
| Parola d'ordine | 330 |
| Porta..... | 330 |
| Tabelle | 330 |
| genere | 330 |
| Utenti..... | 330 |
| Versione..... | 330 |
| Il concetto di transazione | 331 |
| Metodi di classe di connessione | 332 |
| Aperto chiuso..... | 332 |
| Inizia / Conferma / Ripristina | 333 |
| Trova | 333 |
| Creare | 334 |
| modificare | 334 |
| Exec | 334 |
| Citazione | 335 |
| Oggetti risultato | 335 |
| Classe DB | 336 |
| Banca dati..... | 337 |
| Campo | 337 |
| Indice | 337 |
| tavolo | 337 |

| | |
|---|------------|
| Una guida per principianti a | |
| Utente | 338 |
| | |
| Il programma di esempio del database | 338 |
| | |
| Capitolo 16 - Global Gambas..... | 351 |
| Internazionalizzazione..... | 351 |
| Localizzazione | 351 |
| Set di caratteri universale (UCS)..... | 352 |
| Unicode | 352 |
| UTF8..... | 353 |
| Come tradurre in Gambas | 355 |

Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Tabella delle figure

| | |
|--|----|
| Figura 1 Panoramica generale dell'architettura Gambas | 20 |
| Figura 2 la schermata di apertura di Gambas..... | 21 |
| Figura 3 Creazione guidata del progetto Gambas. | 22 |
| Figura 4 Una finestra di dialogo per selezionare il tipo di progetto Gambas che desideri creare | 23 |
| Figura 5 Finestra di dialogo per la selezione del nome del progetto. | 24 |
| Figura 6 Finestra di dialogo per la selezione della directory del progetto. | 24 |
| Figura 7 Finestra di dialogo di conferma del nuovo progetto | 25 |
| Figura 8 L'IDE di Gambas..... | 26 |
| Figura 9 Menu File di Project Explorer | 27 |
| Figura 10 Menu Progetto | 27 |
| Figura 11 Menu di visualizzazione di Project Explorer. | 27 |
| Figura 12 Menu Strumenti di Project Explorer | 27 |
| Figura 13 Menu e barra degli strumenti di Project Explorer | 28 |
| Figura 14 Una finestra di dialogo Divisione per zero errori | 40 |
| Figura 15 La casella degli strumenti di Gambas | 58 |
| Figura 16 Controlli aggiuntivi per QT | 60 |
| Figura 17 Risultati del codice del primo pulsante..... | 65 |
| Figura 18 Dimostrazione delle capacità dei caratteri. | 67 |
| Figura 19 La linea tratteggiata indica lo stato attivo per un controllo | 72 |
| Figura 20 Il layout per SecondProject Form1.form..... | 72 |
| Figura 21 Un modulo parzialmente costruito con i nostri primi quattro controlli. | 73 |
| Figura 22 Menu Evento..... | 74 |
| Figura 23 Aggiunta di FunBtn al nostro modulo. | 76 |
| Figura 24 Come appare il nostro modulo quando viene rilevato un mouse sul modulo. Nota che il testo è cancellato | 77 |
| Figura 25 La barra di avanzamento quando si fa clic su FunBtn tre volte. | 77 |
| Figura 26 ThirdProject (quasi) risultato finale. | 80 |
| Figura 27 Utilizzo dell'HTML per formattare l'output di TLabel | 82 |
| Figura 28 Output di TLabel modificato utilizzando la formattazione HTML.... | 83 |
| Figura 29 Aggiunta di una descrizione comando per informare l'utente su come mostrare / nascondere un controllo | 85 |
| Figura 30 Il nostro ComboBox..... | 86 |
| Figura 31 Editor delle proprietà dell'elenco di modifica. | 87 |
| Figura 32 Formattazione di un'etichetta di testo con HTML | 88 |
| Figura 33 Pulsanti più e meno per modificare l'elenco ComboBox | 88 |
| Figura 34 Come sarà il nostro ListBox | 90 |
| Figura 35 ListBox Modifica dell'editor delle proprietà dell'elenco. | 90 |

Una guida per principianti a92
Figura 36 Come sarà il frame di esempio che costruiremo.

Una guida per principianti a

| | |
|--|-----|
| Figura 37 Un pannello con pulsanti radio..... | 95 |
| Figura 38 Risultati finali del progetto di menu..... | 97 |
| Figura 39 Editor del menu Gambas quando viene avviato per la prima volta..... | 98 |
| Figura 40 Modifica campi per l'Editor dei menu..... | 99 |
| Figura 41 Creazione del menu del nostro progetto | 101 |
| Figura 42 Un'etichetta di testo formattata che mostra il valore del colore | 104 |
| Figura 43 Selezione di colori e caratteri | 111 |
| Figura 44 Creazione di un nuovo carattere predefinito per il controllo TextLabel1 | 112 |
| Figura 45 Un MessageBox di informazioni | 113 |
| Figura 46 Una voce di menu selezionata | 113 |
| Figura 47 MessageBox di una domanda..... | 115 |
| Figura 48 Un messaggio di errore | 115 |
| Figura 49 Un messaggio di avviso..... | 116 |
| Figura 50 Elimina messaggio con tre pulsanti..... | 117 |
| Figura 51 La finestra di dialogo Apri file | 118 |
| Figura 52 Finestra di dialogo Salva file | 119 |
| Figura 53 La finestra di dialogo Seleziona directory | 119 |
| Figura 54 Scelta dell'esempio Explorer | 149 |
| Figura 55 Layout per il nostro esempio ListView | 159 |
| Figura 56 Creazione di una nuova immagine dell'icona in Gambas | 163 |
| Figura 57 La nostra immagine icona quadrata | 163 |
| Figura 58 Casella degli strumenti dell'editor delle icone | 163 |
| Figura 59 L'immagine dell'icona del nostro cerchio | 163 |
| Figura 60 La finestra del progetto TreeView | 164 |
| Figura 61 Come sarà il nostro GridView | 172 |
| Figura 62 Il nostro esempio ColumnView | 175 |
| Figura 63 Icone del progetto di layout | 178 |
| Figura 64 Modalità di progettazione Form1 che mostra il layout dei nostri controlli..... | 179 |
| Figura 65 Programma di layout all'avvio | 182 |
| Figura 66 Un controllo TabStrip | 183 |
| Figura 67 Scheda Progetto Form1.form Design..... | 183 |
| Figura 68 Layout Tab0 | 184 |
| Figura 69 Layout Tab1 | 184 |
| Figura 70 Layout Tab2 ToolButton con icone | 185 |
| Figura 71 Layout Tab3 con un ComboBox | 186 |
| Figura 72 Il programma FileOps in fase di esecuzione..... | 196 |
| Figura 73 Modalità di progettazione Form2.form | 210 |
| Figura 74 Programma Contatti finiti | 248 |

Una guida per principianti a
Figura 75 Form1 visto in modalità progettazione. 253

Una guida per principianti a

| | |
|--|-----|
| Figura 76 Gestione contatti in esecuzione autonoma sul desktop | 266 |
| Figura 77 layout gfxDemo Form1..... | 271 |
| Figura 78 Risultati del clic sul pulsante Testo..... | 274 |
| Figura 79 Risultati del clic del pulsante InvRect. Notare il minuscolo schermo centrale con mirino nero | 275 |
| La Figura 80 Ellissi mostra il disegno di linee ed ellissi. | 278 |
| Figura 81 Output dopo aver fatto clic sul pulsante FillRect | 280 |
| Figura 82 Utilizzo di Draw.Polygon per disegnare un triangolo | 282 |
| Figura 83 Utilizzo di Draw.Polyline per disegnare linee. | 284 |
| Figura 84 Utilizzo di immagini affiancate con il pulsante TileImg del nostro programma demo. | 292 |
| Figura 85 Caricamento di un file SVG in Gambas | 295 |
| Figura 86 Risultati di errore rilevati con TRY | 305 |
| Figura 87 Schermata di apertura del programma di test CATCH | 307 |
| Figura 88 Errore Div per Zero rilevato da CATCH. | 307 |
| Figura 89 L'etichetta di testo aggiornata con informazioni sull'errore. | 308 |
| Figura 90 Messaggio informativo che indica che l'errore è stato cancellato. | 308 |
| Figura 91 Schermata principale dopo la cancellazione dell'errore | 308 |
| Figura 92 Finestra di dialogo di errore predefinita di Gambas | 309 |
| Figura 93 Il nostro evento è stato generato | 311 |
| Figura 94 Programma MouseOps in esecuzione | 315 |
| Figura 95 Programma KbdOps in esecuzione..... | 318 |
| Figura 96 Il programma BitOps in modalità progettazione | 320 |
| Figura 97 Il nostro programma bitOps in esecuzione | 325 |
| Figura 98 Il modulo FMain in modalità progettazione | 339 |
| Figura 99 FRequest Form in modalità progettazione | 339 |
| Figura 100 Scelta dell'opzione Traduci in Gambas..... | 356 |
| Figura 101 La finestra di dialogo Traduci in Gambas. | 356 |

This program is released under the terms of the GNU General Public License (GPL). It is released to the public domain under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Questa pagina è intenzionalmente vuota.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Ringraziamenti

Prima di tutto, un ringraziamento molto speciale a Benoît Minisini per la creazione di Gambas e per il suo sostegno a questo sforzo per documentare ulteriormente questo meraviglioso linguaggio. Senza l'iniziativa di Benoît, faremmo tutti fatica a trovare uno strumento migliore di quello che esiste oggi sulle piattaforme Linux. Gran parte della documentazione iniziale della lingua Gambas è stata inserita nel Gambas Wiki da Benoît e merita un credito speciale per aver reso disponibili queste informazioni. Poiché era l'unica fonte pubblicata nota di documentazione definitiva esistente prima di questa stesura, gran parte del materiale di riferimento qui è stato raccolto da quella serie iniziale di documentazione. Come per qualsiasi materiale scritto, non vi è alcuna garanzia che questa documentazione sia accurata come quella che potresti trovare nell'ultima versione del prodotto Gambas.

L'autore desidera estendere un ringraziamento speciale a Fabien Bodard per il suo aiuto nel rendere questo lavoro una realtà. Fabien ha modificato instancabilmente il codice, ha esaminato i progetti di esempio presentati nel presente documento e ha fornito una visione approfondita del funzionamento interno di Gambas

- tutto al momento della vendemmia per la sua vigna. Laurent Carlier e Steve Starr meritano entrambi la mia gratitudine per il loro meticoloso lavoro di modifica del codice e per garantire che tutto funzioni come pubblicizzato. Il loro feedback, suggerimenti e correzioni sono stati molto apprezzati. Vorrei anche ringraziare Daniel Campos che è stato un grande vantaggio nell'aiutarmi a capire alcuni dei punti più fini di Gambas. Nigel Gerrard ha anche contribuito al successo di questo progetto fornendo una revisione finale e una modifica del materiale del database. Tra questi programmati della Gambas Hall of Fame, chi altri avrebbe potuto farlo meglio?

Innumerevoli e-mail sono state ricevute da molti membri della comunità Gambas, che hanno fornito incoraggiamento e supporto continuo che mi hanno motivato a completare questo sforzo e renderlo disponibile a tutti. Date le difficoltà di scrivere un libro in primo luogo, è stata una grande soddisfazione ricevere un tale sostegno da perfetti sconosciuti che condividono tutti la visione unica di rendere Gambas un successo. Posso solo sperare che questo libro renda giustizia a questo sforzo e accolga favorevolmente qualsiasi suggerimento per il cambiamento, nonché eventuali complimenti o critiche costruttive. È stato molto divertente imparare questa meravigliosa nuova lingua ed esplorare le possibilità che offre. Dedico questo libro a tutti quegli utenti e sviluppatori di Gambas che hanno fatto affidamento gli uni sugli altri nella comunità di utenti / sviluppatori di Gambas per trasformare questo prodotto in realtà. Visione condivisa, obiettivi comuni e software aperto sono forze potenti che non dovrebbero mai essere sottovalutate. È noto che tali forze cambiano il mondo e continueranno a farlo a lungo dopo che questo libro sarà stato letto e dimenticato.

Una guida per principianti a
John W. Rittinghouse, Ph.D., CISM
ottobre 2005

Prefazione

Negli ultimi tre anni mi sono tuffato a capofitto nell'ambiente di programmazione Linux. Considerando che la mia prima installazione di Linux risale al 1996, non dovrebbe essere considerato un buon primo avvio. In quanto entusiasta sviluppatore Basic, l'ambiente Linux mancava di uno strumento che mi permetesse di programmare facilmente nell'ambiente Linux. Per caso mi sono imbattuto in un piccolo progetto, frutto di oltre due anni di lavoro di un uomo di nome Benoit Minisini. Dopo aver capito come superare le idiosincrasie della compilation, ho scoperto quali sarebbero state le fasi embrionali di uno dei progetti più favolosi che avrei potuto immaginare, Gambas! A quel tempo, Gambas era già implementato con il proprio ambiente di sviluppo integrato (IDE) e un evidenziatore di sintassi.

Oggi, Gambas è arrivato come un prodotto maturo che, nella sua prima versione, consente a un utente di costruire applicazioni grafiche o applicazioni basate su console, supporta la gestione del database, la connessione a Internet o ai server socket, utilizza la compressione dei dati, supporta DCOP con le applicazioni KDE , e altro ancora. Un'applicazione Gambas può essere tradotta direttamente dall'IDE e impacchettata in formato binario per diverse distribuzioni Linux. Ciò che inizialmente mi affascinava ancora è che Gambas è andato oltre l'essere solo un altro linguaggio di programmazione perché supporta tutte le caratteristiche di un prodotto "professionale" pur mantenendo la sua semplicità. Gambas fornisce uno strumento reale per il programmatore principiante e consente a un programmatore alle prime armi di sviluppare applicazioni di alta qualità. Qui, in questo libro, troverete il primo trattamento completo di Gambas. Rende semplice l'approccio al linguaggio Gambas e il lettore può facilmente passare dagli argomenti di livello principiante agli argomenti più avanzati che i programmatori professionisti usano quotidianamente.

Gambas ha e continuerà ad evolversi. La versione 1.0 di Gambas (discussa in questo libro) è la base di un linguaggio che si evolverà per essere ancora più potente. Ad esempio, Gambas versione 2 renderà possibile effettuare chiamate API a una libreria nativa. Permetterà ai programmatore di gestire ancora più tipi di server di database e funzionerà ugualmente bene con le librerie grafiche Qt o GTK. Gambas 2 consentirà ai programmatore di sviluppare i propri componenti dall'ambiente Gambas. Gli sviluppatori potranno creare giochi utilizzando SDL e OpenGL. L'ambito del linguaggio Gambas sta diventando sempre più ampio e la sintassi sta diventando più compatta. Tutte queste idee per il miglioramento

Una guida per principianti a

e il cambiamento non è solo nelle teste di una scuderia di sviluppatori scelti che lavorano al progetto Gambas 2. Provengono da tutti gli utenti di Gambas che forniscono le idee che rendono Gambas 2 ancora migliore di prima. Può già fare molto di più rispetto alla versione 1.0. Consiglio vivamente che, mentre aspetti il rilascio stabile di Gambas 2, inizi il tuo addestramento Gambas con la tua macchina e con quello che troverai in questo libro. Ti preparerà per una meravigliosa esperienza di programmazione in ambiente Linux.

Fabien Bodard e Benoit Minisini

This document is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed further terms or conditions without the express written consent of the author.

Una guida per principianti a

Capitolo 1 - Presentazione di Gambas

Gambas è stato inizialmente creato da Benoît Minisini, un residente della periferia di Parigi. Secondo Benoît, Gambas è un linguaggio di base con estensioni agli oggetti. Il nome stesso è un gioco di parole "Gambas significa quasi base" e, secondo l'autore, Gambas si è evoluto grazie alle sue personali esperienze di programmazione con il prodotto software Microsoft Visual Basic®¹. Piuttosto che lottare con l'orrendo numero di bug e idiosincrasie riscontrate in quel prodotto, ha deciso di creare Gambas.

Gambas è concesso in licenza con GNU Public License² e ha assunto una vita propria grazie alla sua immensa popolarità. Gambas funziona sulla maggior parte delle principali piattaforme Linux e la versione stabile corrente (al momento della stesura di questo documento) è la versione 1.0.9. Minisini chiarisce che Gambas non è compatibile con Visual Basic e non sarà mai reso compatibile. La sintassi e il funzionamento interno di Gambas sono di gran lunga migliori e più facili da usare. Minisini ha dichiarato [sic] di aver "preso da Visual Basic ciò che ha trovato utile; il linguaggio Basic, l'ambiente di sviluppo e la capacità di creare rapidamente [e facilmente] programmi con interfacce utente [grafiche]".³

A Minisini non piaceva lo scarso livello di programmazione comune a molti programmi Visual Basic. Molti credono che questo problema possa essere dovuto all'uso "forzato" di pratiche di programmazione stravaganti imposte agli sviluppatori a causa dell'ampia gamma di bug e delle strane idiosincrasie del linguaggio proprietario VB. Gambas è stato sviluppato per essere il più coerente, logico e affidabile possibile. Poiché è stato sviluppato con un approccio progettato per migliorare lo stile di programmazione e catturare il meglio che il linguaggio di programmazione Basic ha da offrire, l'aggiunta della programmazione basata su oggetti ha permesso a Gambas di diventare un ambiente di programmazione popolare, moderno, stabile e utilizzabile per gli sviluppatori Linux.

Architettura Gambas

Ogni programma scritto con Gambas Basic è composto da una serie di file di progetto. Ogni file all'interno di un progetto descrive una classe. I file di classe vengono inizialmente compilati e successivamente eseguiti dall'interprete Gambas. Questo è molto simile a come funziona Java. Gambas si compone dei seguenti programmi:

1 Il lettore è incoraggiato a visitare <http://gambas.sourceforge.net/index.html> per saperne di più sul progetto Gambas.

2 Per informazioni sulla licenza, visitare <http://www.gnu.org/licenses/licenses.html#GPL>.

Una guida per principianti a

- 3 Vedere l'introduzione del sito Web Wiki di Gambas all'indirizzo <http://gambas.sourceforge.net/index.html> per ulteriori dettagli.

Una guida per principianti a

- ✓ Un compilatore
- ✓ Un interprete
- ✓ Un archiviatore
- ✓ Un componente dell'interfaccia utente grafica
- ✓ Un ambiente di sviluppo

La Figura 1 sotto⁴ è un'illustrazione dell'architettura complessiva di Gambas. In Gambas, un progetto contiene file di classe, moduli, moduli e file di dati. Un progetto Gambas è archiviato in una singola directory. La compilazione di un progetto utilizza un metodo di compilazione incrementale che richiede solo la ricompilazione delle classi modificate. Ogni riferimento esterno di una classe viene risolto dinamicamente in fase di esecuzione. L'archiviatore Gambas trasforma l'intera struttura della directory del progetto in un eseguibile autonomo. L'ambiente di sviluppo di Gambas è stato scritto con Gambas per dimostrare le fantastiche capacità del linguaggio.

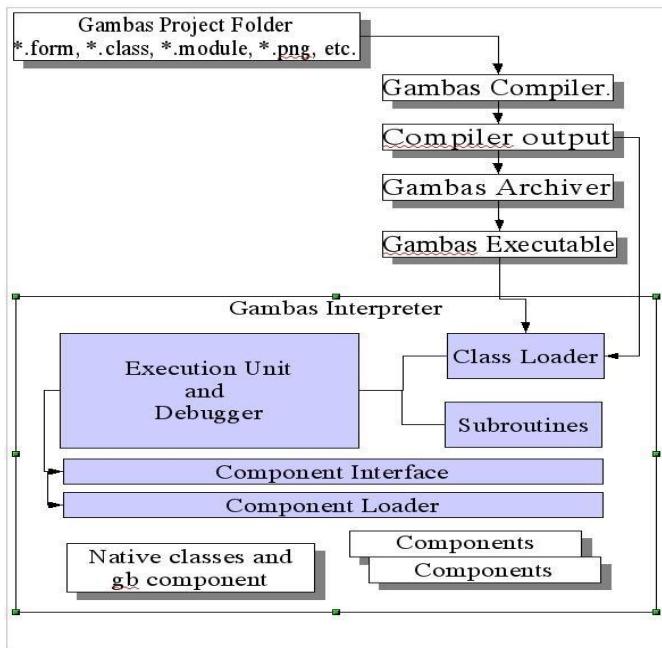


Figura 1 Panoramica generale dell'architettura Gambas.

Alcune altre caratteristiche che distinguono Gambas dagli altri linguaggi includono il fatto che Gambas ha un'architettura a componenti estensibili che consente ai programmati di estendere il linguaggio. Chiunque può scrivere componenti come librerie condivise che aggiungono dinamicamente nuove classi native all'interprete. L'architettura dei componenti è documentata online nell'encyclopedia Gambas Wiki⁵.

⁴ Vedi URL <http://gambas.sourceforge.net/architecture.html>, per la grafica originale.

Una guida per principianti a

5 Copyright (c) 1999-2005 degli autori che contribuiscono. Tutto il materiale su Gambas Wiki è di proprietà del contributore

Una guida per principianti a

Tratteremo i componenti di Gambas in maggiore dettaglio più avanti in questo libro. Per impostazione predefinita, Gambas Interpreter è un programma di solo testo (basato su consolle). L'architettura dei componenti viene utilizzata per la parte dell'interfaccia utente grafica (GUI) del linguaggio. Poiché la GUI è implementata come un componente Gambas, ha la capacità di essere indipendente da qualsiasi toolkit GUI specifico. Con Gambas, puoi scrivere un programma e scegliere quale toolkit, come GTK + 6, Qt7, ecc., Da utilizzare in seguito. L'attuale versione di Gambas implementa l'interfaccia utente grafica con il toolkit Qt. I componenti della GUI derivano direttamente dalla libreria QT. Si consiglia al lettore di consultare anche la documentazione di QT8 per comprendere meglio i controlli della GUI. Secondo Minisini, è previsto un componente GTK + con un'interfaccia quasi identica a quella del componente Qt.

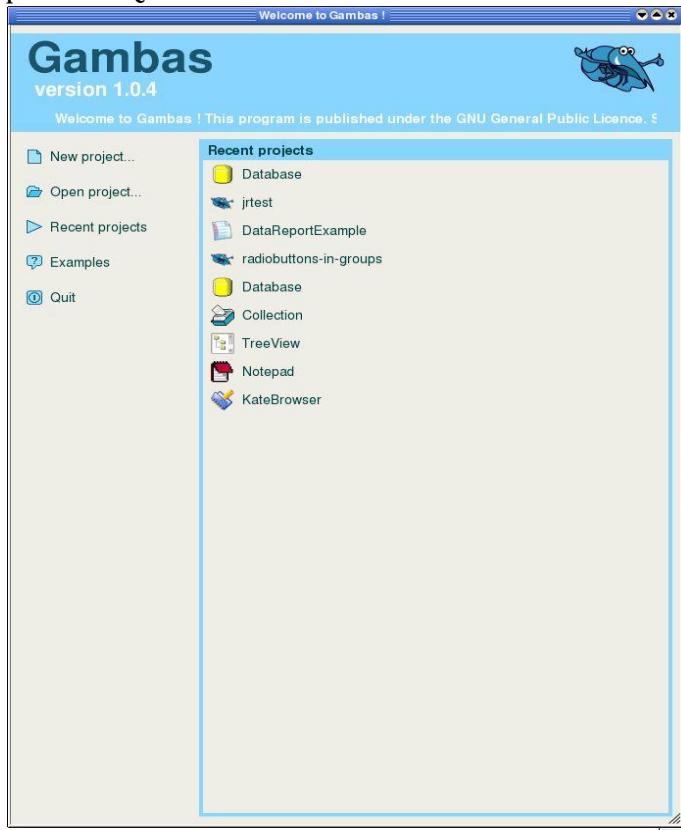


Figura 2 la schermata di apertura di Gambas.

by John W. Rittinghouse, all rights are reserved. It is released to the
Gambas Community under an Open Content License (OCL) and may not be distributed
without the express written consent of the author.

autori.

- 6 Per maggiori informazioni su GTK +, visita <http://www.gtk.org/>.
- 7 Per maggiori informazioni su Qt, visita <http://www.trolltech.com/products/qt/>.
- 8 <http://doc.trolltech.com/3.3/index.html>

Una guida per principianti a

Un'altra caratteristica di Gambas che lo distingue dagli altri linguaggi di programmazione è la capacità di qualsiasi finestra o finestra di dialogo di essere utilizzata come un controllo. Questa funzione non è direttamente supportata in altri linguaggi di programmazione. Inoltre, i progetti Gambas sono facilmente traducibili in quasi tutte le altre lingue. Dimostreremo questa caratteristica più avanti nel libro, quando tratteremo l'Internazionalizzazione (i18n).

L'ambiente di programmazione Gambas

Per ora, facciamo un breve tour di Gambas e ti presentiamo l'ambiente di programmazione Gambas. Tutti gli esempi e il codice scritti per questo libro sono stati sviluppati utilizzando la versione 1.0.9 di Gambas in esecuzione su Linspire® 5.09. Dovrebbero funzionare su qualsiasi piattaforma su cui puoi installare Gambas con successo. Per gli utenti Linspire®, è semplice come fare clic su un pulsante utilizzando la funzione di magazzino ClickNRun® di quel prodotto. Quando fai clic per la prima volta sull'icona del desktop per avviare il programma Gambas, ti verrà presentata una schermata di apertura simile a quella che si trova nella Figura 2 sopra. La schermata di benvenuto consente di creare un nuovo progetto, aprire un progetto esistente, visualizzare e selezionare da progetti recenti, guardare esempi o uscire. Per il nostro rapido tour di Gambas, selezioneremo l'opzione Nuovo progetto e creeremo il nostro primo progetto Gambas.



Figura 3 Creazione guidata del progetto Gambas.

This document is (C) 2005 by the Gambas User Committee. It may be freely distributed, but express written consent of the author.

9 Per maggiori informazioni su Linspire, visita <http://www.linspire.com>.

Una guida per principianti a

Fare semplicemente clic sul pulsante Avanti >> e verrà visualizzata un'altra finestra di dialogo che richiede di scegliere il tipo di progetto che si desidera creare (vedere la Figura 4, di seguito). Questa finestra di dialogo richiede di scegliere tra un progetto GUI o un progetto console.

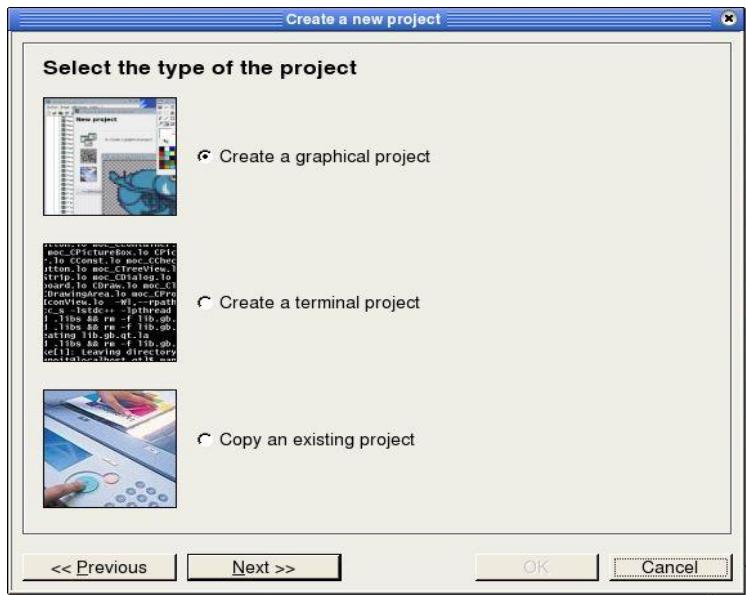


Figura 4 Una finestra di dialogo per selezionare il tipo di progetto Gambas che si desidera creare.

Per il nostro esempio, scegli la prima selezione, "Crea un progetto grafico" e fai clic sul pulsante Avanti >> nella parte inferiore della finestra di dialogo. La finestra di dialogo successiva che vedrai ti richiede di dare un nome al tuo progetto e di scegliere dove si troverà nel tuo file system locale. È mostrato nella Figura 5 di seguito.

Questa sezione della procedura guidata richiede di specificare il nome del progetto poiché verrà archiviato su disco. Ricorda, in Gambas un progetto è interamente contenuto in una directory o cartella. Il nome della cartella che scegli come nome per il progetto è quello che verrà creato sul tuo sistema.

Il titolo del progetto sarà quello che specifichi nel secondo campo di immissione di testo della finestra di dialogo. Inoltre, la sezione Opzioni ti consente di scegliere se il tuo progetto sarà traducibile o meno e se i controlli utilizzati sui tuoi moduli saranno resi pubblicamente accessibili. Discuteremo queste opzioni in modo più dettagliato più avanti in questo libro. Per ora, basta compilare la finestra di dialogo come mostrato nella Figura 5 e fare clic su Avanti >>.

Una guida per principianti a



Figura 5 Finestra di dialogo per la selezione del nome del progetto.

Scegliere una directory appropriata per il proprio file system e fare clic sul pulsante Avanti >> per procedere alla schermata di dialogo finale della procedura guidata, come mostrato nella pagina seguente nella Figura 7. Questa finestra di dialogo è semplicemente una schermata di conferma delle scelte effettuate. La cosa più importante da ricordare di questa schermata è che è l'ultima possibilità che hai per eseguire il backup e apportare modifiche prima che il tuo nuovo progetto venga creato.

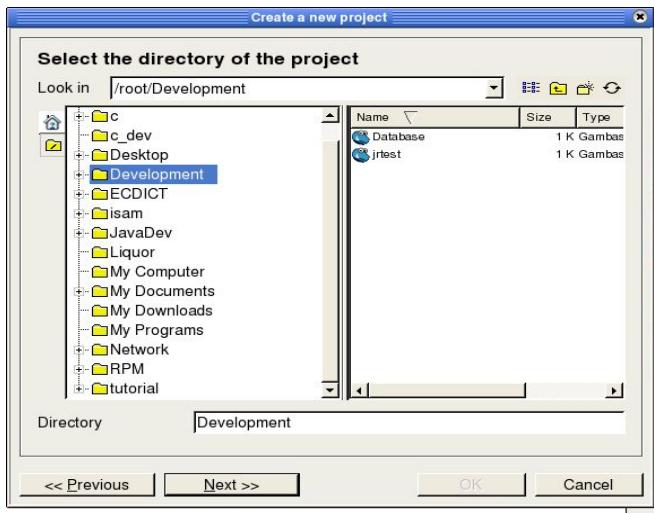


Figura 6 Finestra di dialogo per la selezione della directory del progetto.

This product is (C) 2005 by John W. Rittman. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the written consent of the author.

This product is (C) 2005 by John W. Langhouse, all rights reserved. It is released to the Gambas User Community under an [Open Content License](#). It may not be distributed or modified without the express written consent of the author.

Una guida per principianti a

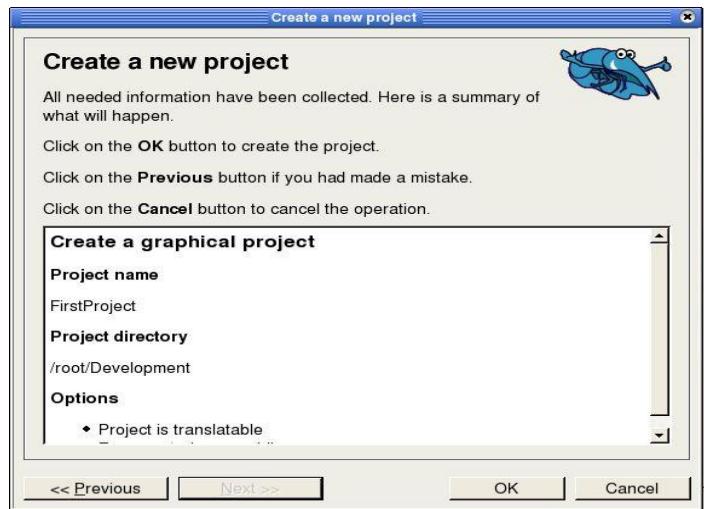


Figura 7 Finestra di dialogo di conferma del nuovo progetto.

Dopo aver fatto clic sul pulsante "OK" per terminare la procedura guidata, Gambas ti presenterà l'ambiente di sviluppo integrato (IDE) di Gambas. L'IDE è costituito da diversi componenti mostrati nella Figura 8 nella pagina successiva.

Componenti IDE di Gambas

Project Explorer è la finestra principale di Gambas. Ti mostra una visualizzazione ad albero dei tipi di file trovati all'interno del tuo progetto (cioè, file di classe, moduli, moduli e altri tipi di file come file di dati e immagini o icone) e Project Explorer ti consente di eseguire la maggior parte della gestione dei progetti Gambas operazioni, come l'apertura e il salvataggio di progetti, la creazione di eseguibili, l'esecuzione del programma o il debug, la visualizzazione o l'occultamento di varie finestre di dialogo di Gambas, ecc. Dal TreeView in Esplora progetti, vedrai gli elementi elencati come segue:

- ✓ Classi
- ✓ Le forme
- ✓ Moduli
- ✓ Dati

Classi elenca i file di classe che hai creato per il tuo progetto. Le classi sono fondamentalmente modelli che possono essere utilizzati per creare oggetti in fase di esecuzione, con codice per definire proprietà, metodi e gestori di eventi per ogni oggetto creato.

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It may not be distributed or sold without the express written consent of the author.

Una guida per principianti a

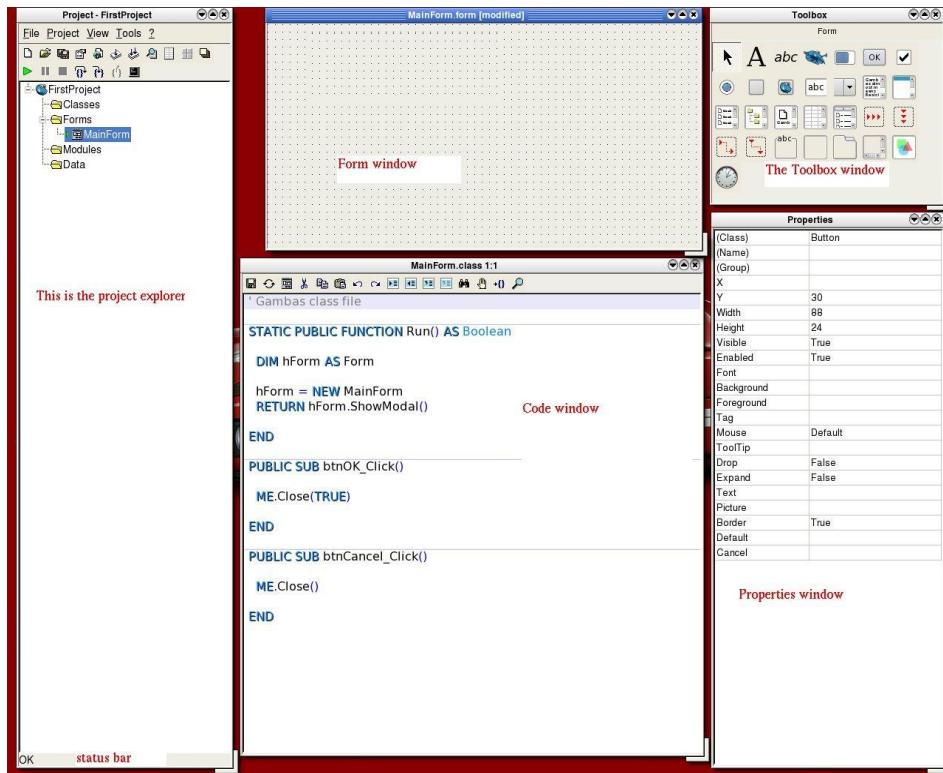


Figura 8 L'IDE di Gambas.

Le forme elenca i vari moduli che crei per il tuo progetto. I moduli sono le finestre con cui l'utente interagisce effettivamente.

Moduli mostra i moduli che hai scritto per il tuo progetto. I moduli sono semplicemente insiemi di subroutine e funzioni da utilizzare ovunque nel programma. A differenza delle classi, non è possibile ricavarne oggetti in fase di esecuzione e non hanno gestori di eventi.

Dati elenca gli altri file nel progetto. Questi possono includere qualsiasi altro tipo di file utilizzato per creare il progetto, come file grafici, icone, bitmap, file di testo o HTML e persino file multimediali. Nella parte inferiore di Project Explorer troverai la barra di stato che viene utilizzata per indicare ciò che Gambas sta attualmente facendo.

Quando inizi a sviluppare un progetto in Gambas, di solito vorrai iniziare con un modulo principale. La maschera principale è dove si verificheranno l'avvio e l'inizializzazione del programma e di solito è la prima cosa che l'utente vede quando esegue (o esegue) l'applicazione che hai creato. Questo è il modulo a cui aggiungere i controlli e specificare quali azioni devono essere intraprese quando l'utente

Una guida per principianti a
interagisce con

Una guida per principianti a

quei controlli. Tale interazione tra l'utente e la GUI viene definita eventi. I controlli si trovano nella finestra ToolBox. È possibile modificare l'aspetto e il comportamento dei controlli impostando le proprietà per ogni controllo. Le proprietà possono essere visualizzate nella finestra delle proprietà.

Ora, diamo un'occhiata ai menu e ai pulsanti che si trovano nella parte superiore di Project Explorer. I menu (vedere le figure da 9 a 12 di seguito) controllano tutte le principali attività di gestione di Gambas. Il menu File ti consentirà di aprire un progetto, salvare un progetto, creare un nuovo progetto, aprire alcuni progetti di esempio di Gambas o uscire dall'uso di Gambas. Il menu Progetto è dove avviene la compilazione del programma. È inoltre possibile creare il programma eseguibile, creare un archivio di origine o creare un pacchetto di installazione. Questo menu ti offre la possibilità di tradurre il tuo programma in un'altra lingua. Infine, puoi impostare le proprietà IDE di Gambas da questo menu.

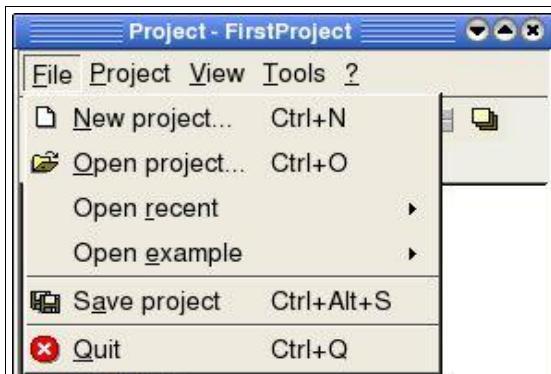


Figura 9 Menu File di Project Explorer.

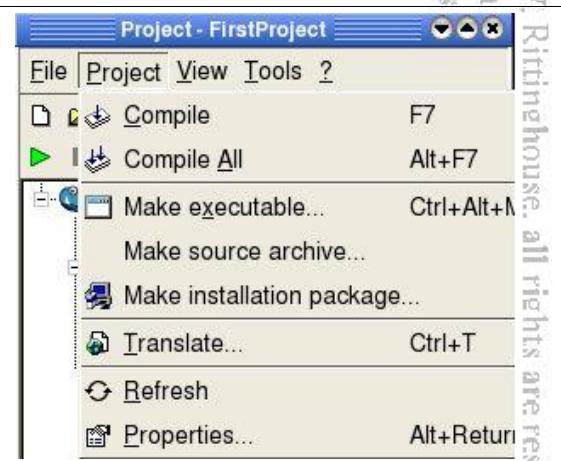


Figura 10 Menu Progetto.



Figura 11 Menu di visualizzazione di Project Explorer.



Figura 12 Menu Strumenti di Project Explorer.

Una guida per principianti a

Il menu Visualizza ti consentirà di visualizzare la finestra delle proprietà o la finestra della casella degli strumenti. È possibile aprire una console o utilizzare l'Editor delle icone per creare icone di programma. L'opzione Gerarchia aprirà una finestra e ti mostrerà la gerarchia di classi del tuo progetto. Infine, da questo menu, puoi chiudere tutte le finestre attualmente aperte nell'IDE. I pulsanti della barra degli strumenti (vedere la Figura 13 di seguito) forniscono l'accesso con un solo clic alle voci di menu più comuni. Passando il cursore del mouse su uno dei pulsanti verrà visualizzata una descrizione comando che ti dirà quale azione di menu eseguirà quel particolare pulsante.

La Figura 13 di seguito mostra il menu File e la barra degli strumenti. Da Project Explorer TreeView puoi fare doppio clic su un modulo e apparirà per la modifica. La modifica dei moduli è semplicemente una questione di selezionare il tipo di controllo che si desidera posizionare sul modulo e quindi utilizzare il mouse per ridimensionarlo (disegnarlo) sul modulo.



Figura 13 Menu e barra degli strumenti di Project Explorer.

Facendo clic con il pulsante destro del mouse sul modulo o su uno dei suoi figli (controlli) verrà visualizzato un menu a comparsa che consentirà di eseguire operazioni sul controllo, modificarne le proprietà o eliminarlo. Il controllo attualmente selezionato è indicato da quattro quadrati neri chiamati maniglie. Facendo clic su una maniglia e utilizzando il mouse per trascinare il controllo dove si desidera, sarà possibile spostare o ridimensionare il controllo. Facendo doppio clic su un controllo verrà visualizzata la finestra dell'editor del codice e verrà visualizzato qualsiasi gestore di eventi esistente per il controllo o verrà visualizzato un gestore di eventi predefinito se non è stato specificato nessuno.

L'editor di codice Gambas ti consente di scrivere codice per gestire gli eventi per i controlli che hai inserito nel modulo. Nella finestra della casella degli strumenti, lo strumento di selezione è l'unico elemento della casella degli strumenti che non è effettivamente un controllo. Lo strumento di selezione ti dà semplicemente il controllo del puntatore del mouse predefinito. Utilizzare questo puntatore per selezionare i controlli ed eseguire operazioni di posizionamento e ridimensionamento sui moduli e sui controlli associati.

Una guida per principianti a

Dal menu File di Project Explorer, scegli l'opzione Esci e salva il progetto.
Quando lo riapriremo, tutto il tuo lavoro verrà salvato. Nella prossima sezione,
inizieremo a coprire gli elementi essenziali che devi comprendere per programmare

Una guida per principianti a

Gambas. Torneremo su questo progetto per sviluppare un programma che utilizza la GUI dopo aver appreso un po 'di più sull'ambiente di codifica Gambas, i concetti di linguaggio primari necessari per utilizzare Gambas e alcune nozioni di base sui tipi di dati e sulle variabili. Questo è tutto trattato nella prossima sezione e costituisce il resto di questa introduzione alla programmazione Gambas. Per ora, prenditi una breve pausa e continueremo quando tornerai.



This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed in any manner other than the terms of the OCL. This document is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The author shall not be liable for any damages resulting from the use of this document.

Capitolo 2 - Gambas Concetti linguistici

In questo capitolo inizieremo ad apprendere i concetti di base necessari per padroneggiare il linguaggio di programmazione Gambas. Gli argomenti trattati in questo capitolo includono l'apprendimento dei tipi di dati, delle costanti e delle variabili di Gambas e come dichiarare e assegnare valori a tali costanti e variabili. Impareremo gli operatori aritmetici di base, gli operatori di confronto e gli operatori di stringa.

Variabili, tipi di dati e costanti di Gambas

Le variabili devono essere definite all'inizio di una classe, metodo o funzione. Le dichiarazioni di variabili possono essere locali a una procedura o una funzione oppure possono essere dichiarate globali a una classe. Una variabile globale è accessibile ovunque nella classe in cui è dichiarata. Il formato di una dichiarazione di variabile globale in Gambas assume la seguente forma generale:

```
[STATICO] (PUBBLICO | PRIVATO) Identificatore [Dichiarazione di matrice] AS  
[NUOVO] Tipo di dati
```

Se viene specificata la parola chiave PUBLIC, è anche accessibile alle altre classi che hanno qualsiasi riferimento a un oggetto di quella classe. Se viene specificata la parola chiave PRIVATE, non è accessibile al di fuori della classe in cui è stata definita. Se viene specificata la parola chiave STATIC, la stessa variabile verrà condivisa con ogni oggetto della classe in cui è dichiarata. Se viene specificata la parola chiave NEW, la variabile viene inizializzata con (ovvero, istanziata con) una nuova istanza della classe utilizzando il tipo di dati specificato. Per le dichiarazioni di variabili locali, il formato è il seguente:

```
[DIM] Identificatore Tipo di dati AS
```

Questo dichiarerà una variabile locale in una procedura o funzione. Questa variabile è accessibile solo alla procedura o alla funzione in cui è dichiarata. Di seguito è riportato un esempio di diverse dichiarazioni locali:

```
DIM iValue AS INTEGER  
DIM stMyName AS STRING  
DIM fMyMatrix [3, 3] COME GALLEGGIANTE  
DIM oMyObject COME OGGETTO
```

This product is covered by the Creative Commons License CC-BY-SA 2005 by John Vininghouse, all rights reserved. It is released to the Gambas (Gambas Community under a Creative Content License (CC)) may not be distributed under any other terms or conditions than those of the author.

Una guida per principianti a

Attualmente ci sono nove tipi di dati di base che un programmatore può utilizzare per scrivere il codice del programma in Gambas10. Questi tipi di dati includono quanto segue:

| | | |
|-----------------|------------------|---------------|
| Booleano | Byte | Corto |
| Numero | Galleggia | Data |
| intero | nte | |
| Corda | Variante | Oggett |
| | | o |

Tipi di dati booleani può contenere un solo valore, TRUE o FALSE. TRUE è definito come 1 e FALSE è definito come 0. La dichiarazione di una variabile booleana occupa 1 byte di memoria e il valore predefinito è FALSE. Un esempio di dichiarazione di una variabile per una variabile booleana è simile a questo:

```
STATICO PRIVATO bGrid AS Booleano
```

Tipicamente, un programmatore userebbe un tipo di dati booleano quando gli unici valori per la variabile sarebbero sì / no, VERO / FALSO, 1/0, ecc. Se i valori usati nel tuo programma potessero essere qualcos'altro, booleano sarebbe una selezione inappropriata per il tipo di dati variabile. Un'altra cosa da notare nella dichiarazione della variabile sopra è il posizionamento della lettera minuscola b in da del nome della variabile.

Una buona convenzione di programmazione incoraggia questa pratica, nota come notazione ungherese¹¹, poiché consente ai programmatori di sapere quale tipo di dati è la variabile semplicemente sapendo che la "b" sta per booleano. Cosa succede quando un programmatore desidera utilizzare un tipo di dati Byte invece di un tipo di dati booleano? In genere, una seconda lettera viene aggiunta alla dichiarazione della variabile, quindi invece di usare 'b' davanti al nome della variabile il programmatore userebbe 'by' come di seguito:

```
STATICO PRIVATO di Qualcosa COME Byte
```

Le lettere "ar" "s", "i", "f", "d", "st", "v" e "o" sono notazioni comunemente usate quando si dichiarano variabili durante la programmazione in Gambas. È una buona pratica di programmazione sforzarsi di aderire a questa tecnica in modo che altri possano prendere il tuo codice e usarlo senza dover cercare in giro per trovare il tipo di dati per ogni variabile incontrata. Alcuni programmatori usano addirittura più della prima lettera

(s) per nominare le loro variabili. Ad esempio, codificheranno IntMyNumber o **ByteQualcosa**.

Una guida per principianti a

- 10 Nella seconda versione di Gambas (Gambas2) Sono pianificati i tipi di dati "Lungo" e "Singolo". Questi tipi di dati sono come i tipi di dati C LONG LONG e FLOAT. Questo è un miglioramento significativo perché Gambas2 fornirà la programmazione direttaaccesso all'API C e supporterà anche piattaforme a 64 bit.
- 11 La notazione ungherese (HN) è una convenzione di denominazione creata da Charles Simonyi di Microsoft. È stato presentato per la prima volta nella sua tesi ed è ampiamente utilizzato in tutto il codice sorgente del sistema operativo Windows, tra gli altri luoghi.

Una guida per principianti a

Il tipo di dati Byte può contenere valori da 0 a 255 e occupa un byte di memoria quando dichiarato. Il valore predefinito quando viene dichiarato un tipo di dati Byte è 0. Se si è certi che i valori non supereranno 255, questo tipo di dati è appropriato per l'uso. Se è possibile che i valori assegnati a questa variabile superino 255, il risultato probabile sarebbe un arresto anomalo del programma in fase di esecuzione. È preferibile utilizzare il tipo di dati Short o un tipo di dati Integer per tali situazioni. Per un tipo di dati Short, i valori possono variare da -32768 a +32767. I tipi di dati brevi occupano due byte di memoria e quando vengono dichiarati il valore predefinito è zero. I tipi di dati interi occupano il doppio della memoria, occupando quattro byte. L'intervallo di valori che puoi utilizzare con il tipo di dati Integer è da

Da -2.147.483.648 a +2.147.483.647. È simile al tipo di dati LONG utilizzato in VB. Esempi di dichiarazioni di variabili del tipo di dati Short e Integer sono:

```
STATICO PUBBLICO sSomeShort AS Short  
STATICO PUBLIC iSomeInteger AS Integer
```

Quando gli interi non funzionano per i tuoi scopi, Gambas ti fornisce il tipo di dati Float. Questo tipo di dati ti consente di utilizzare numeri in virgola mobile per il tuo programma. I tipi di dati Float sono come i tipi di dati Double usati in C e VB. L'intervallo di valori per le variabili del tipo di dati float va da -1.79769313486232E308 a

4.94065645841247E324 per valori negativi e da 4.94065645841247E324 a 1.79769313486232E308 per valori positivi. Le variabili float occupano otto byte di memoria e impostano un valore zero quando dichiarate. Una dichiarazione di esempio per un float sarebbe la seguente:

```
DIM fRadius AS Float
```

L'ultimo tipo di dati numerico che abbiamo in Gambas è il tipo di dati Date. Anche le variabili di data occupano otto byte di memoria. La parte della data della data viene memorizzata in un numero intero di quattro byte e la parte dell'ora viene memorizzata in un numero intero di quattro byte. Viene memorizzato come [Anno, Mese, Giorno] [, Ore, Minuti, Secondi] e viene solitamente utilizzato con le funzioni Data e Ora incorporate di Gambas, di cui parleremo più avanti in questo libro. Il tipo di dati della data viene impostato su un valore NULL quando viene inizializzato. Ecco come dichiarare un tipo di dati Date:

```
DIM ddate AS Date  
DIM dttime AS Date
```

Una guida per principianti a
Stringhe, varianti e oggetti sono i tipi di dati non numerici supportati in
Gambas. Un tipo di dati String è una serie di zero o più caratteri che vengono trattati
come una singola entità. Le stringhe possono contenere dati alfanumerici.
Alfanumerico

Una guida per principianti a

significa che i dati possono contenere qualsiasi combinazione di lettere e numeri interi o caratteri speciali come \$% ^ & *. Le stringhe, se dichiarate, richiedono quattro byte di memoria. Ciò significa che la dimensione massima per una stringa è di 4 byte * 8 bit per byte o 32 bit al quadrato (1.024 byte). Le variabili stringa impostano di default un valore NULL quando dichiarate. Dichiara una variabile stringa proprio come faresti con qualsiasi altra variabile:

```
STATIC PUBLIC stSomeString AS String
```

Il tipo di dati Variant viene utilizzato quando non si conosce il tipo di dati che la variabile riceverà. Ad esempio, se si leggono dati da un file, è possibile leggere un numero intero, una stringa, un carattere singolo, numeri in virgola mobile, ecc. Per garantire che i dati vengano inseriti in una variabile senza causare un arresto anomalo del programma, viene utilizzato il tipo di dati variante. È quindi possibile testare i dati delle varianti utilizzando alcune funzioni incorporate di Gambas per determinare il tipo di dati oppure è possibile convertire i dati nel tipo di dati necessario utilizzando una funzione di conversione. Lo dimostreremo più avanti nel libro. Per ora, è importante solo che tu capisca che esistono tipi di dati varianti e che vengono utilizzati quando non sei sicuro del tipo di dati che la variabile conterrà.

Il tipo di dati Object è un tipo di dati speciale che contiene e fa riferimento a oggetti come controlli e moduli. Più avanti, quando inizieremo a discutere della programmazione OO, tratteremo l'uso dei tipi di dati oggetto in maggiore dettaglio. La tabella mostrata di seguito è presentata come un comodo riferimento:

Tipi di dati Gambas

| Nome | Descrizione | Dimensione della memoria | Predefinito |
|---------------|---|--------------------------|-------------|
| Booleano | Vero o falso | 1 byte | FALSO |
| Byte | 0 ... 255 | 1 byte | 0 |
| Corto | 32768 ... +32767 | 2 byte | 0 |
| Numero intero | 2147483648 ... + 2147483647 | 4 byte | 0 |
| Galleggiante | Simile al doppio tipo di dati in C | 8 byte | 0 |
| Data | Data / ora, ciascuna memorizzata in un numero intero di 4 byte. | 8 byte | NULLO |
| Corda | Un riferimento a una stringa di lunghezza variabile. | 4 byte | NULLO |
| Variante | Può essere costituito da qualsiasi tipo di dati. | 12 byte | NULLO |
| Oggetto | Un riferimento indiretto a un oggetto. | 4 byte | Nullo |

Ora che conosci tutti i diversi tipi di dati supportati da Gambas, inizieremo a esaminare cosa puoi fare con quei tipi di dati. Quando si utilizzano le variabili nei

Una guida per principianti a programmi Gambas, possono essere rappresentate da dati che cambiano (ad esempio, è una variabile) oppure possono essere rappresentate da dati che rimangono

Una guida per principianti a

costante durante tutto il programma. Questo tipo di dati è noto come costante in Gambas. Le costanti Gambas vengono utilizzate per rappresentare un riferimento a un oggetto NULL, una stringa di lunghezza zero, una data NULL o una variante non inizializzata. Esempi di costanti includono i valori NULL, TRUE e FALSE. Per dichiarare una costante in Gambas utilizzare il seguente formato:

```
(PUBBLICO | PRIVATO) CONST Identificatore AS Tipo dati =  
        valore
```

Questo dichiara una costante globale di classe. Questa costante è accessibile ovunque nella classe in cui è dichiarata. Se viene specificata la parola chiave PUBLIC, è accessibile anche alle altre classi che hanno un riferimento a un oggetto di questa classe. I valori costanti devono essere booleani, numeri interi, virgola mobile o tipi di dati stringa. Di seguito sono riportati alcuni esempi di dichiarazioni costanti:

```
CONST PUBBLICO MAX_FILE AS Integer = 30  
PRIVATE CONST MAGIC_HEADER AS String = "# file di modulo Gambas"
```

Le costanti incorporate che useresti in Gambas sono elencate nella tabella seguente:

Costanti di Gambas

| Costante | Esempio |
|-------------------------------------|---------------------------------|
| Il valore VERO. | VERO |
| Il valore FALSE. | FALSO |
| Numeri interi. | 0, 562, 17, 32769 |
| Interi con segno breve esadecimale. | & H100F3, & HF0FF e FFFF |
| Interi con segno esadecimale. | & H1ABF332E e 1CBF302E |
| Interi esadecimali senza segno. | & H80A0 &, & HFCFF & |
| Numero intero binario. | & X1010111101,% 101000011 |
| Numeri in virgola mobile. | 1.1110, 5.3419E + 4 |
| Costanti stringa. | "Ciao, Gambas World!" |
| Costanti stringa da tradurre. | ("Questo è molto, molto bello") |
| Costante NULL / stringa nulla. | NULLO |

Una guida per principianti a

Le costanti stringa possono contenere anche i seguenti caratteri di escape:

| Carattere di fuga | Equivalente ASCII |
|-------------------|-------------------|
| \n | CHR \$ (13) |
| \r | CHR \$ (10) |
| \t | CHR \$ (9) |
| \" | Virgolette doppie |
| \\" | Barra rovesciata |
| \xx | CHR \$ (& Hxx) |

È possibile scrivere una costante di stringa in più parti successive. Ad esempio, "Mio figlio" "è" "sedici" è visto da Gambas come "Mio figlio ha sedici anni".

Assegnazione variabile

Un programmatore può assegnare qualsiasi valore a una variabile in Gambas utilizzando il seguente formato generale:

Variabile = Espressione

Questa istruzione di assegnazione assegna il valore di un'espressione a uno dei seguenti elementi:

- ✓ Una variabile locale
- ✓ Un parametro di funzione
- ✓ Una variabile globale (classe)
- ✓ Un elemento array
- ✓ Una proprietà o una variabile oggetto pubblico Di

seguito sono riportati alcuni esempi di assegnazioni di variabili:

```
iMyVal = 1984  
stMyName = "Orwell"  
fMyNum = 123,45
```

Assegnazione utilizzando l'istruzione WITH

Questa istruzione è più comunemente utilizzata per impostare le proprietà per i controlli. L'espressione che esiste tra la parola chiave WITH e l'istruzione END WITH

This product is (C) 2005 by Jun W. Rittihouse, all rights are reserved. It is released to the public under an Open Content License (OCL) and may not be distributed or any other terms or conditions without the express written consent of the author.

Una guida per principianti a

si usa. L'espressione inizierà con la notazione del punto, cioè, è possibile utilizzare .Text. WITH assegna all'espressione tratteggiata a sinistra del segno di uguale il valore che si trova a destra del segno di uguale. L'espressione deve essere un oggetto. Ecco un esempio di come appare la struttura WITH:

```
CON espressione
    .object = "qualcosa" ;
FINISCI CON
```

Ad esempio, il codice seguente è un codice equivalente a hButton.Text = "Exit"

```
CON hButton
    .Text = "Exit"
END WITH
```

Operatori ed espressioni

Ora che sappiamo come dichiarare variabili e costanti e come assegnare valori a queste variabili e costanti, diamo un'occhiata alle operazioni che possono essere eseguite con esse. Inizieremo con gli operatori di confronto, quindi daremo uno sguardo agli operatori aritmetici e agli operatori di stringa.

Operatori di confronto

Il confronto di due variabili richiede la ricerca di risposte a domande come "x è uguale a y" o "è a minore di b". I seguenti confronti sono supportati in Gambas:

| Operatore | Significato | Esempio |
|-----------|----------------------------|----------------------|
| = | È uguale a | SE a = b ALLORA ... |
| <> | Non è uguale | SE a <> c ALLORA ... |
| < | È meno di | SE a < d ALLORA ... |
| > | È più grande di | SE a > e ALLORA ... |
| <= | È minore o uguale a | SE a <= f ALLORA ... |
| >= | È più grande di O uguale a | SE a >= g ALLORA ... |

Operatori aritmetici

Tutte le operazioni aritmetiche di base sono supportate in Gambas, questi

Una guida per principianti a

gli operatori includono addizione, sottrazione, moltiplicazione e divisione. I simboli standard per queste operazioni sono "+", "", "*" e "/". Ad esempio, Numero + Numero aggiungerà due numeri. Quando un valore o una variabile è preceduto da un segno meno, -222, ad esempio, Gambas calcola il segno opposto di quel numero. Il valore Zero è l'opposto di se stesso. Ora inizieremo a scrivere del codice Gambas utilizzando la funzione dell'applicazione terminale Gambas. La finestra della console mostrerà il nostro output, quindi usiamo Gambas per sperimentare con gli operatori mentre li apprendiamo.

Iniziamo a programmare Gambas

Ora che conosciamo i tipi di dati e le assegnazioni delle variabili, bagniamoci i piedi con Gambas. Avvia Gambas e dal menu File Esplora progetto seleziona Nuovo progetto. Durante la procedura guidata Nuovo progetto, seleziona un progetto terminale e fai clic su Avanti. Assegna un nome a questo progetto TerminalTest e posizionalo in una directory denominata TerminalTest. Non preoccuparti di nessuna delle altre opzioni. Basta fare clic su Avanti >> fino al completamento della procedura guidata. Una volta visualizzato l'IDE con il tuo nuovo progetto, dovremo creare una classe di avvio per eseguire il nostro codice. Da Project Explorer, trova l'elemento TreeView denominato Classes e fai clic con il pulsante destro del mouse. Scegli l'opzione Nuovo ... e prendi il nome predefinito Class1. Apparirà la finestra del codice e all'interno della finestra dovrebbe apparire qualcosa del genere:

```
"File di classe Gambas
STATICO PUBBLICO SUB
Principale ()

FINE
```

Diamo un'occhiata ad alcune parole chiave di Gambas di cui dovresti conoscere un po' di più prima di procedere.

Dichiarazioni END, RETURN e QUIT

La parola chiave END indica la fine di una procedura o di una funzione. Ci sono differenze da VB quando si usa END. In VB, il comando End chiude tutti i moduli e file e termina il programma. In Gambas, il comando END funziona più come End Function di VB combinato con End Sub di VB. Chiude la funzione o il sottoprogramma. Per la funzionalità del comando End di VB, utilizzare il comando QUIT. Termina immediatamente il programma. Tutte le finestre sono chiuse e tutto viene liberato nella memoria nel modo più pulito possibile. In Gambas, quando desideri uscire da una routine, puoi utilizzare il comando RETURN. Di solito,

Una guida per principianti a
RETURN viene utilizzato per restituire un valore alla procedura chiamante. Il
formato di RITORNO è:

Una guida per principianti a

RITORNO [Espressione]

Quando Gambas esegue il comando RETURN, chiude una procedura o una funzione e completa il suo lavoro restituendo il valore di Expression. Ora, inserisci il seguente codice dopo la riga "Gambas class file (nota che i commenti in Gambas iniziano con" [aka il segno di spunta]) e tra le istruzioni STATIC PUBLIC SUB Main () e END. Dopo aver inserito il codice sottostante nella finestra del codice Gambas, fare clic sul verde ➤ dalla barra degli strumenti di Project Explorer per eseguire il programma. Ecco il codice che vuoi provare per primo:

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Inter
    DIM R AS Inter
    N = 3
    R = 6
    STAMPA "====>; N; "|; R; " e "; N; "|; R; FINE
```

Se tutto va bene (e dovrebbe), vedrai ballare il gambero blu (noto anche come la mascotte di Gambas) e la finestra della console risponderà con quanto segue:

```
====> 3 | 6 e 6 | 3
```

Nota che il valore della variabile N è cambiato da un 3 positivo a 3 e il valore di 6 è cambiato in un valore positivo di 6. Non preoccuparti della sintassi dell'istruzione PRINT o dell'uso della parola chiave DIM usata per dichiarare le nostre variabili per adesso. Tratteremo queste parole chiave più avanti nel libro.

Per sottrarre valori, usa il formato Numero Numero e Gambas sottrarrà il secondo numero dal primo.

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Inter
    DIM R AS Inter
    N = 8
    R = 5
    STAMPA "====>; NR;
FINE
```

La console risponderà con quanto segue:

```
==> 3
```

This product is (C) 2005 by John Waddington, all rights are reserved. It is released to the Gambas User Community under a Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Per moltiplicare i numeri, usiamo il formato Numero * Numero e Gambas moltiplicherà i due numeri. Ecco un altro esempio di console da provare:

```
STATICO PUBBLICO SUB Principale ()  
    DIM N AS Intero  
    DIM R AS Intero  
    N = 8  
    R = 5  
    STAMPA "====>"; N * R;  
FINE
```

La console risponderà con quanto segue:

```
==> 40
```

La divisione non è diversa dalla moltiplicazione. Usa il formato Numero / Numero per fare in modo che Gambas divida due numeri. Se il valore del numero a destra della barra è zero, si verificherà una divisione per zero. Prova questo esempio di console:

```
STATICO PUBBLICO SUB Principale ()  
    DIM N AS Intero  
    DIM R AS Intero  
    N = 9  
    R = 3  
    STAMPA "====>"; N / R;  
FINE
```

La console risponderà con quanto segue:

```
==> 3
```

Ora prova a utilizzare \ per dividere invece del carattere /:

```
STATICO PUBBLICO SUB Principale ()  
    DIM N AS Intero  
    DIM R AS Intero  
    N = 9  
    R = 5  
    STAMPA "====>"; N \ R;  
FINE
```

La console risponderà con il quoziente:

```
==> 1
```

This product is (C) 2005 by John W. Patterson, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed in any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Se si utilizza una barra rovesciata per dividere i numeri, ovvero Numero \ Numero Gambas calcola il quoziente dei due numeri. Se il valore del numero a destra della barra rovesciata è zero, si verificherà una divisione per zero. A \ B è l'equivalente di INT (A / B). Per ottenere il resto, possiamo usare la funzione MOD incorporata in questo modo:

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Intero
    DIM R AS Intero
    N = 9
    R = 5
    STAMPA "====>; N \ R; "e il resto è:"; 9 MOD 5; FINE
```

La console risponde con:

```
====> 1 e il resto è: 4
```

Utilizzando Number MOD Number calcola il resto del quoziente dei due numeri. Se il valore del numero a destra dell'operatore MOD è zero, si verificherà una divisione per zero. Infine, possiamo testare l'errore Divisione per zero digitando questo esempio:

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Intero
    DIM R AS Intero

    N = 9
    R = 0
    STAMPA "====>; N / R;
FINE
```

Gambas risponderà con quanto segue:

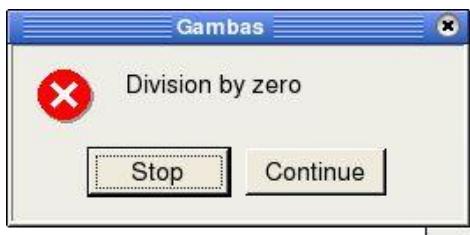


Figura 14 Una finestra di dialogo Divisione per zero errori.

NOTA: fare clic sul pulsante Interrompi quando viene visualizzata questa finestra di dialogo.

Una guida per principianti a

Per elevare un numero a una data potenza (esponente), usa il formato di Numero ^ Potenza e Gambas eleva Numero alla potenza specificata dall'operatore di Potenza. Prova questo:

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Intero
    DIM R AS Intero
    N = 2
    R = 3
    STAMPA "====>"; N ^ R;
FINE
```

La console risponderà con quanto segue:

```
====> 8
```

Gambas ha anche la capacità di supportare operazioni aritmetiche logiche¹². L'utilizzo del formato Number AND Number indica a Gambas di utilizzare l'operatore AND per calcolare l'operatore AND matematico dei valori binari di entrambi i numeri. Prova questo:

```
STATICO PUBBLICO SUB Principale ()
    DIM N AS Intero
    DIM R AS Intero

    N = 0
    R = 0
    PRINT "=>"; N AND R; "è il risultato AND di"; N; "e"; RR
    = 1
    PRINT "=>"; N AND R; "è il risultato AND di"; N; "e"; RN
    = 1
    PRINT "=>"; N AND R; "è il risultato AND di"; N; "e"; R
END
```

La finestra della console risponde con:

```
=> 0 è il risultato AND di 0 e 0
=> 0 è il risultato AND di 0 e 1
=> 1 è il risultato AND di 1 e 1
```

Allo stesso modo, Number OR Number utilizza l'operatore OR e calcola l'OR matematico del valore binario dei due numeri.

¹² In Gambas2 consentirà l'utilizzo di operatori concatenati. Ad esempio, a + = 2 o B / = 4 funzionerà come se stessi programmando ion C or C ++.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
STATICO PUBBLICO SUB Principale ()  
DIM N AS Intero  
DIM R AS Intero  
  
N = 0  
R = 0  
PRINT "=>"; N OR R; "è il risultato OR di"; N; "OR";  
RR = 1  
PRINT "=>"; N OR R; "è il risultato OR di"; N; "OR";  
RN = 1  
PRINT "=>"; N OR R; "è il risultato OR di"; N; "OR"; R  
END
```

La finestra della console risponde con:

```
=> 0 è il risultato OR di 0 OR 0  
=> 1 è il risultato OR di 0 OR 1  
=> 1 è il risultato OR di 1 OR 1
```

Numero XOR Numero utilizza l'operatore XOR e calcola l'OR matematico esclusivo del valore binario dei due numeri.

```
STATICO PUBBLICO SUB Principale ()  
DIM N AS Intero  
DIM R AS Intero  
  
N = 0  
R = 0  
PRINT "=>"; N XOR R; "è il risultato XOR di"; N; "XOR"; RR  
= 1  
PRINT "=>"; N XOR R; "è il risultato XOR di"; N; "XOR"; RN  
= 1  
PRINT "=>"; N XOR R; "è il risultato XOR di"; N; "XOR"; R  
END
```

La finestra della console risponde con:

```
=> 0 è il risultato XOR di 0 XOR 0  
=> 1 è il risultato XOR di 0 XOR 1  
=> 0 è il risultato XOR di 1 XOR 1
```

Inoltre, i seguenti operatori manipolano uno o più valori numerici e restituiscono un valore numerico:

DEC | INC | MI PIACE | NON

Una guida per principianti a

L'operatore DEC viene utilizzato per decrementare un valore di uno. L'operatore INC incrementerà il valore di uno. La variabile può essere qualsiasi destinazione di un'assegnazione, ma deve essere un valore numerico.

```
STATICO PUBBLICO SUB Principale ()  
    DIM N AS Interger  
    DIM R AS Interger  
  
    N = 5  
    R = 5  
    DEC N  
    INC R  
    STAMPA "====>"; N; " | "; R;  
FINE
```

La console risponderà con quanto segue:

```
====> 4 | 6
```

PIACE viene utilizzato per eseguire un confronto booleano di una stringa. Prende il formato di String LIKE Pattern e restituisce TRUE se la String corrisponde al Pattern. Il modello può contenere i seguenti caratteri di corrispondenza del modello:

| | |
|--------|--|
| * | Corrisponde al numero N di qualsiasi tipo di carattere. |
| ? | Trova qualsiasi singolo carattere. |
| [abc] | Trova qualsiasi carattere specificato tra i simboli delle parentesi. |
| [xy] | Trova qualsiasi carattere presente nell'intervallo xy. |
| [^ xy] | Trova qualsiasi carattere che non esiste nell'intervallo xy. |

```
STATICO PUBBLICO SUB Principale ()  
    STAMPA "Rittinghouse" COME "R  
** END
```

La console risponde con:

```
VERO
```

Il carattere speciale \ impedisce al carattere successivo di seguirlo in una stringa
dall'essere interpretato come una parte generica della stringa. Pensa al carattere \ come a un codice di controllo. Prova questo:

```
STATIC PUBLIC SUB Main ()
```

Una guida per principianti a
PRINT "Samson" LIKE "S **"

This product is (C) 2005 by IBM W. R. Inghouse, all rights are reserved. It is released to the Gambas User Community under the terms of the GNU General Public License (GPL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
STAMPA "Gambas" MI PIACE "? [Aa] *"  
STAMPA "Leonard" MI PIACE "G  
[Aa] \\ *" STAMPA "Alfred" MI  
PIACE "G [^ Aa] *"  
FINE
```

La console risponde con:

```
VERO  
VERO  
FALSO  
FALSO
```

Nota: è necessario utilizzare una doppia barra rovesciata, \\ per stampare una barra rovesciata o una sequenza di stringhe speciale contenente barre rovesciate. Altrimenti \\ verrà interpretato dal compilatore come un carattere speciale come '\ n', '\ t', '\ *', ecc. In alternativa, puoi usare questa stringa di pattern:**MI PIACE "G [Aa] [*]"**

L'operatore NOT viene utilizzato per restituire il risultato di un'espressione. Il formato è il seguente:

**Risultato = NON
espressione**

Quando Gambas valuta questa espressione, calcola il NOT logico dell'espressione. Se l'espressione è una stringa o un oggetto, Gambas restituisce TRUE se l'espressione è NULL. Ecco alcuni esempi da provare sulla tua console:

```
STATIC PUBLIC SUB Main  
() PRINT NOT TRUE  
STAMPA NON  
FALSA STAMPA  
NON 11  
STAMPA NON  
"Gambas" STAMPA  
NON ""  
FINE
```

La console risponde con:

```
Falsa  
verit  
à  
12  
Falsa  
verit  
à
```

Una guida per principianti a Operatori di stringa

In Gambas, quando vuoi confrontare o concatenare stringhe, puoi usare

Una guida per principianti a

gli operatori di stringa. Questi operatori consentono di concatenare stringhe e percorsi di file, eseguire operazioni LIKE come spiegato sopra, determinare se le stringhe sono uguali o non uguali, minori, maggiori, minori o uguali o maggiori o uguali tra loro. La seguente tabella di operazioni sulle stringhe viene mostrata per comodità:

Gambas String Operations

| Operazione su stringhe | Risultato da determinare |
|------------------------|--|
| String & String | Concatena due stringhe. |
| String & / String | Concatena due stringhe che contengono nomi di file. Aggiungere un separatore di percorso tra le due stringhe, se necessario. |
| Stringa come modello | Eseguire la corrispondenza dei modelli utilizzando l'operatore LIKE. |
| String = String | Determina se le due stringhe sono uguali. |
| Corda <> String | Determina se le due stringhe sono diverse. |
| String <String | Determina se la prima stringa è inferiore alla seconda |
| Stringa> Stringa | Determina se la prima stringa è maggiore della seconda |
| Corda <= String | Determina se la prima stringa è inferiore o uguale alla seconda |
| Corda > = String | Determina se la prima stringa è maggiore o uguale alla seconda |

Nota: tutti i confronti tra operatori di stringa fanno distinzione tra maiuscole e minuscole.

Ecco alcune cose da provare utilizzando la console Gambas:

```
STATICO PUBBLICO SUB Principale ()
    DIM a AS String
    DIM b AS String
    a = "ham"
    b = "hamburger"
    STAMPA "====>; A & B;
    a = "Gambas"
    b = 1
    STAMPA "====>; A & "è un numero" & B;
FINE
```

La console risponde con:

```
====> hamburger
====> Gambas è il numero 1
```

Una guida per principianti a

Capitolo 3 - Parole chiave e controllo del flusso del programma

In Gambas, il programmatore controlla il programma utilizzando parole chiave ed espressioni condizionali. Gambas è un linguaggio guidato dagli eventi. Il programmatore ha la capacità di dirigere ciò che accade ogni volta che si verifica un evento. Il programmatore utilizza le parole chiave Gambas (che sono parole riservate con una sintassi molto specifica) per creare istruzioni che guidano ciò che il programma farà in una data circostanza. I condizionali sono test di un'espressione o di una variabile. Il test potrebbe essere una valutazione del risultato di un'operazione o un confronto di uguaglianza, ad esempio. La tabella seguente mostra tutte le parole chiave Gambas attualmente supportate:

| | | | | | |
|----------|----------|-----------|-------------|--------|-------|
| ROMPERE | ASTUCCIO | CONTINUA | PREDEFINITO | FARE | ALTRO |
| FINE | FINISCI | FINE | SELEZIONARE | FINE | CON |
| SE | | | SE | | |
| PER | PER | VAI A | CICLO | IL | |
| CIASCUNO | | | CONTINUO | PROSSI | |
| | | | O | MO | |
| SMETTERE | RIPETERE | RITORNO | SELEZIONARE | PASSO | POI |
| E | | | SE | | |
| PER | FINO A | ASPETTARE | WEND | MENTRE | CON |
| | | E | | | |

Il modo migliore per capire cosa significano queste parole chiave e questi condizionali è introdurli gradualmente con esempi. Piuttosto che scorrere l'elenco delle parole chiave in ordine alfabetico, adottiamo l'approccio di introdurle in base al tipo di funzionalità che supportano. Inizieremo con le istruzioni e i comandi più elementari e proseguiremo con quelli più complessi man mano che procediamo. Cominciamo con l'istruzione Gambas PRINT.

L'istruzione PRINT

L'istruzione PRINT stampa le espressioni sullo standard output. Le espressioni vengono convertite in stringhe dalla funzione Str(). PRINT assume il formato di:

```
PRINT Espressione [(; |,) Espressione ...] [(; |,)]
```

Le parentesi utilizzate nella definizione della sintassi sopra indicano parametri opzionali. Se non è presente un punto e virgola o una virgola dopo

Una guida per principianti a l'ultima espressione, viene stampato automaticamente un carattere di nuova riga dopo l'ultima espressione. Se viene utilizzata una virgola al posto di un punto e virgola per separare le espressioni, viene stampato un carattere di tabulazione (codice ASCII 9) tra i valori di output per separare le espressioni. PRINT può anche essere utilizzato per indirizzare l'output a un file. Discuteremo la stampa su file quando arriveremo alla sezione su

Una guida per principianti a

Ingresso e uscita. Quando si utilizza PRINT per scrivere l'output su un file, le espressioni vengono inviate al file di flusso e viene utilizzato questo formato:

```
PRINT # File, Espressione [(; [,) Espressione ...] [,)]
```

Ecco qualcosa da provare a utilizzare PRINT:

```
STATICO PUBBLICO SUB Principale ()
    DIM b AS Interio

    b = 1
    PRINT "====>" & "b è:" & B
    PRINT "====>", "b è:", B
    PRINT "====>; "b è:" & B
FINE
```

La console visualizza il seguente output:

```
====> b è: 1
====> b è: 1
====> b è: 1
```

La dichiarazione IF

L'istruzione IF viene utilizzata per prendere una decisione. IF è una delle strutture più comuni utilizzate dai programmati per effettuare confronti e decisioni in base al risultato di tale confronto. Utilizzando la logica IF, il programma può eseguire confronti utilizzando operatori di cui si è appreso e successivamente prendere decisioni in base al risultato di tali confronti. IF assume la forma generale di:

```
SE Espressione
    ALLORA fai
        qualcosa ...
[ELSE IF Expression THEN
    do qualcos'altro ...]
[ ALTRO
    fare qualcosa di completamente diverso
...] ENDIF
```

Ecco un esempio di IF che puoi provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
```

Una guida per principianti a

```
DIM b AS Interger  
b = 1
```

This product is (C) 2005 by John W. Nutting. All rights are reserved. It is released to the Gambas User Community under an Open Content license (OCL) and may not be distributed under any other terms or conditions. No part of this express written consent of the author.

Una guida per principianti a

```
SE b = 1 ALLORA
    PRINT "====>" & "Gambas è il numero" & B;
ALTRIMENTI SE b <> 1 ALLORA
    PRINT "Non dovrebbe
stamparmi!"; ALTRO
    STAMPA "Sta succedendo qualcosa di
brutto" ENDIF
FINE
```

La console risponde con:

```
====> Gambas è il numero 1
```

L'istruzione SELECT / CASE

L'istruzione SELECT valuta un'espressione, confrontandola con ogni CASE specificato, ed eseguirà il codice racchiuso nell'istruzione CASE corrispondente se l'espressione valutata è TRUE. Se nessuna istruzione CASE corrisponde all'espressione sottoposta a valutazione, viene eseguita l'istruzione DEFAULT o CASE ELSE. L'istruzione SELECT / CASE consente a un programmatore di creare un blocco di codice in grado di valutare molti risultati di espressioni senza dover codificare una quantità eccessiva di istruzioni IF / ELSEIF / ELSE. Il formato dell'istruzione SELECT è:

```
SELEZIONA l'espressione
    [CASE Expression [, Expression ...]
     [CASE Expression [, Expression ...]
     [CASE ELSE | PREDEFINITO ...]
FINE SELEZIONA
```

Ecco un po 'di codice per mostrarti come usare l'istruzione SELECT:

```
STATICO PUBBLICO SUB Principale ()

DIM w AS Intero
w = 1
'INIZIO: è un'ETICHETTA, utilizzata con GOTO spiegato di
seguito. INIZIO:
PRINT "Il valore di w è:" & w

SELEZIONA
CASE w
CASE 1
    INC w
    GOTO START
```

CASO 2

Una guida per principianti a

Una guida per principianti a

```
INC w
GOTO START
CASO 3
    INC w
    GOTO START
CASO ALTRO
    PRINT "La variabile w non ha un gestore per:" &
w END SELECT
PRINT "Valore finale di w È:" & w
END
```

La console dovrebbe rispondere con il seguente output:

```
Il valore di w è: 1
Il valore di w è: 2
Il valore di w è: 3
Il valore di w è: 4
La variabile w non ha alcun gestore
per: 4 Valore finale di w IS: 4
```

Poiché la variabile w è stata incrementata al valore di 4 e c'era CASE per gestire quel valore, viene eseguito il blocco CASE ELSE. CASE ELSE può anche essere scritto come CASE DEFAULT. In ogni caso, è dove il codice viene impostato per impostazione predefinita quando nessun caso soddisfa il valore della variabile controllata.

GOTO ed ETICHETTE

Notare l'uso dell'ETICHETTA denominata START: nell'esempio di codice precedente. I due punti devono essere utilizzati con il nome dell'etichetta e devono seguirlo senza spazi tra l'etichetta e i due punti. Le etichette sono le destinazioni in cui un'istruzione GOTO dirigerà il flusso del programma. Sebbene l'uso di GOTO dovrebbe essere giudizioso, a volte è necessario. Viene utilizzato nell'esempio precedente per dimostrare come utilizzare GOTO e LABELS. Successivamente, quando studieremo le strutture di loop, riscriveremo questo codice per utilizzare un meccanismo di looping di Gambas.

L'istruzione FOR / NEXT

Molte volte durante la scrittura del codice, i programmati trovano la necessità di iterare attraverso un insieme di valori (chiamato looping) per elaborare i dati. L'istruzione FOR è un'istruzione di ciclo comunemente utilizzata nei programmi. Assume la forma generale di:

Una guida per principianti a

FOR Variable = Expression TO Expression [STEP Expression] ... NEXT

Una guida per principianti a

L'istruzione FOR ripete un ciclo mentre ad ogni iterazione del ciclo incrementa una variabile. Notare che la variabile deve essere un tipo di dati numerico (cioè un byte, un numero breve, un intero o un numero in virgola mobile) e deve essere dichiarata come variabile locale. Se l'espressione iniziale valutata dall'istruzione FOR è maggiore dell'espressione TO (per valori STEP positivi) o se l'espressione iniziale è minore dell'espressione TO (per valori STEP negativi) il ciclo non verrà eseguito affatto. La parola chiave STEP consente al programmatore di definire la dimensione dell'intervallo incrementato tra le iterazioni del ciclo. Ecco un esempio:

```
STATICO PUBBLICO SUB Principale ()
DIM I AS Interger
DIM J AS Interger

J = 1
PER I = DA 1 A 21 FASE 3
    PRINT "iterazione LOOP:" & J & ", I è uguale a:" & I
    INC J
IL PROSSIMO
    PRINT "J IS:" & J & "e I is:" & I END
```

Questo codice produce il seguente output:

```
Iterazione LOOP: 1, I è uguale a: 1
iterazione LOOP: 2, I è uguale a: 4
iterazione LOOP: 3, I è uguale a: 7
iterazione LOOP: 4, I è uguale a: 10
iterazione LOOP: 5, I è uguale a: 13
iterazione LOOP: 6, I è uguale a: 16
iterazione LOOP: 7, I è uguale a: 19
J IS: 8 e I è: 22
```

Si noti che una volta che il valore di J supera il test di 21 nel ciclo, il ciclo si interrompe e il flusso del codice si sposta oltre l'istruzione NEXT. Il valore di J rimane invariato all'uscita dal ciclo. Sperimenta con il codice sopra, cambiando il valore STEP da 3 a 1, ad esempio. Prova a modificare l'istruzione FOR in questo modo:

```
PER I = 21 a 1 FASE 1
```

Gambas fornisce anche una struttura di ciclo FOR EACH che consente di iterare i valori di una raccolta, un array, classi enumerabili, ecc., Senza la necessità di mantenere un contatore intero. Tratteremo completamente i dettagli di FOR EACH quando discuteremo delle raccolte più avanti in questo libro. A questo punto della

Una guida per principianti a
tua introduzione a Gambas, sei incoraggiato a giocare con il codice in

Una guida per principianti a

esempi e modificare i valori delle variabili, esercitarsi con l'istruzione PRINT, ecc. Impara a conoscere la console e scoprirai rapidamente che è un ottimo strumento per testare espressioni, segmenti di codice, ecc. La cosa peggiore che può accadere è che il tuo programma esploda e devi riavviare il computer - improbabile, ma effettivamente possibile. Gioca e mettiti comodo con Gambas. È il modo migliore per imparare.

FARE [MENTRE] LOOP

La struttura DO [WHILE] LOOP inizia l'esecuzione di un ciclo che non terminerà fino a quando non viene soddisfatta una condizione o il codice all'interno della struttura del ciclo forza un'uscita. Il codice eseguito nel ciclo è delimitato dalle parole chiave DO e LOOP. Se la parola chiave opzionale WHILE non è specificata, il ciclo verrà eseguito per sempre (un ciclo infinito) o fino a quando una condizione all'interno della struttura del ciclo non forzerà un'uscita. Se viene specificata la parola chiave facoltativa WHILE, il ciclo viene interrotto una volta che il risultato valutato dell'espressione diventa FALSE. In altre parole, mentre i risultati di questa espressione sono TRUE, continua a iterare (eseguire) il ciclo. Se l'espressione è FALSE quando il ciclo viene avviato, il ciclo non verrà eseguito affatto. Ecco il formato del DO LOOP:

```
DO [WHILE Expression]
...
CICLO CONTINUO
```

Il seguente esempio di codice dovrebbe aiutarti a capire meglio come usare DO ... WHILE LOOP:

```
STATICO PUBBLICO SUB Principale ()
DIM a AS Interco
a = 1

FARE MENTRE a
<= 5 SE a = 1
ALLORA
    STAMPA "Hello World, looping" & a & "time".
ALTRO
    STAMPA "Hello World, looping" & a & "times".
FINISCI SE
INC a
LOOP
DEC a
PRINT
STAMPA "Addio mondo, ho ripetuto un totale di:" & a &
"volte". FINE
```

This document is (C) 2006 JPBsoft. All rights reserved. It is released to the User Community under the GNU General Public License (GPL) and may not be distributed, copied or modified without the express written consent of the author.

This product is (C) 2007 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

La console stampa quanto segue:

```
Hello World, looping 1 volta.  
Hello World, looping 2 volte.  
Hello World, looping 3 volte.  
Hello World, loop 4 volte.  
Hello World, loop 5 volte.
```

```
Goodbye World, ho ripetuto un totale di: 5 volte.
```

WHILE [Expression] WEND Loop

Questa struttura di loop inizia un loop delimitato dalle istruzioni WHILE ... WEND. Il ciclo viene ripetuto mentre Expression è TRUE. Se, all'ingresso iniziale, l'espressione è FALSE, il ciclo non viene mai eseguito. Le strutture DO WHILE LOOP e WHILE ... WEND sono equivalenti. Prova questo codice:

```
STATICO PUBBLICO SUB Principale ()  
  
DIM a AS Interger  
  
a = 1  
  
MENTRE a <= 5  
    SE a = 1  
    ALLORA  
        STAMPA "Hello World, WHILE ... WEND looping" & a & "time."  
    ALTRO  
        STAMPA "Hello World, WHILE ... WEND looping" & a & "times".  
    FINISCI SE  
    INC a  
WEND  
DEC a  
PRINT  
STAMPA "Arrivederci, MENTRE ... WEND ha ripetuto un totale di:" & a  
& "volte." FINE
```

La console dovrebbe stampare quanto segue:

```
Hello World, WHILE ... WEND looping 1  
time. Hello World, WHILE ... WEND looping  
2 times. Hello World, WHILE ... WEND in  
loop 3 volte. Hello World, WHILE ... WEND  
looping 4 times. Hello World, WHILE ...  
WEND in loop 5 volte.
```

```
Arrivederci, MENTRE ... WEND ha ripetuto un totale di: 5 volte.
```

Una guida per principianti a

Il ciclo REPEAT UNTIL

Questa struttura del ciclo inizia a eseguire il codice delimitato dalle parole chiave REPEAT e UNTIL. Un ciclo ripetuto verrà sempre eseguito almeno una volta, anche se il valore UNTIL è inizialmente FALSE. Ecco un esempio da provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
    DIM a AS Interger
    a = 1

    RIPETERE
        SE a = 1 ALLORA
            STAMPA "Hello World, REPEAT UNTIL looping" & a & "time."
        ALTRO
            STAMPA "Hello World, REPEAT UNTIL looping" & a & "times".
        FINISCI SE
        AUMENTARE
    a FINO A a>
    5 DIC a
    STAMPA
    PRINT "Goodbye, REPEAT UNTIL looped per un totale di:" & a &
"volte". FINE
```

La console dovrebbe stampare quanto segue:

```
Hello World, REPEAT UNTIL looping 1 volta.
Hello World, REPEAT UNTIL looping 2 volte.
Hello World, REPEAT UNTIL looping 3 volte.
Hello World, REPEAT UNTIL looping 4 volte.
Hello World, REPEAT UNTIL looping 5 volte.
```

```
Arrivederci, REPEAT UNTIL looped per un totale di: 5 volte.
```

Definizione e utilizzo di array in Gambas

Esistono due tipi di array che possono essere utilizzati in Gambas. Il primo tipo di array Gambas funziona come gli array utilizzati nel linguaggio di programmazione Java. L'altro tipo di array utilizzato in Gambas è noto come array nativo. Quando si utilizzano array Javalike, è necessario ricordare che gli array sono oggetti delle seguenti classi: Integer [], String [], Object [], Date [] e Variant []. Tutti questi tipi di array Javalike possono avere una sola dimensione. Li dichiari così:

```
DIM MyArray AS NEW Integer []
```

Una guida per principianti a

Gli array vengono sempre inizializzati come void all'avvio. Sono dinamici e hanno molti metodi utili che si applicano a loro. Quando si utilizzano array monodimensionali, questi sono la scelta giusta. D'altra parte, gli array nativi possono supportare implementazioni multidimensionali o array con oggetti array delle seguenti classi: Integer, String, Object, Date e Variant. Sono dichiarati in questo modo:

```
DIM MyArray [Dim1, Dim2, ...] AS Integer  
DIM MyArray [Dim1, Dim2, ...] AS String  
DIM MyArray [Dim1, Dim2, ...] AS Variant
```

Gli array nativi possono avere fino a otto dimensioni. NON sono oggetti. Sono allocati nello stack se li dichiari locali a una funzione. Sono allocati all'interno dei dati dell'oggetto se li dichiari come globale nell'ambito. Gli array nativi NON sono dinamici. Non possono crescere o ridursi una volta dichiarati. Puoi solo impostare o ottenere dati da un elemento in questi tipi di array. Ecco un esempio di utilizzo di un array nativo tridimensionale in Gambas. In questo esempio, l'array è riempito con valori interi compresi tra 0 e 26. Immettere questo codice nella finestra del codice della console:

```
STATICO PUBBLICO SUB Principale ()  
DIM i AS Intero  
DIM ii AS Intero  
DIM iii AS Intero  
  
DIM narMatrix [3, 3, 3] AS Integer  
  
PER i = da 0 a 2  
    PER ii = da 0 a 2  
        PER iii = da 0 a 2  
            STAMPA i, ii, iii & "=>"; narMatrix [i,  
                ii, iii] = i * 9 + ii * 3 + iii PRINT  
            narMatrix [i, ii, iii]  
        AVAN  
    TI  
    AVANTI  
FINE  
SUCCE  
SSIWA
```

Quando esegui questo codice, il tuo output nella finestra della console dovrebbe essere simile a quello mostrato qui:

```
0 0 0 ==> 0  
0 0 1 ==> 1  
...
```

Una guida per principianti a

2 2 1 ==> 25

2 2 2 ==> 26

Una guida per principianti a

Collezioni

Le raccolte sono gruppi di oggetti implementati con una tabella hash. Gli oggetti di raccolta utilizzano chiavi implementate come tipi di dati stringa. I valori dei dati corrispondenti a qualsiasi chiave di raccolta sono un tipo di dati Variant. Gli oggetti in una raccolta sono enumerabili. NULL viene utilizzato quando nulla è associato a una determinata chiave. Di conseguenza, associare NULL a una chiave ha lo stesso effetto che rimuoverlo dalla raccolta. La dimensione della tabella hash interna cresce dinamicamente man mano che i dati vengono inseriti. Questa classe viene creata utilizzando questo formato generale:

```
Collezione DIM hCollection AS  
hCollection = NEW Collection ([Mode AS Integer])
```

Notare la parola EACH dopo l'istruzione FOR. Il costrutto FOR EACH viene discusso di seguito. Inoltre, entreremo molto più in dettaglio sulle raccolte più avanti in questo libro, quando parliamo di programmazione orientata agli oggetti. Svilupperemo un'applicazione che fa ampio uso delle raccolte. Per ora, tieni presente che gli elementi della raccolta vengono enumerati nell'ordine in cui sono stati inseriti nella raccolta. Tuttavia, se si sostituisce il valore di una chiave già inserita, viene mantenuto l'ordine di inserimento originale.

La dichiarazione FOR EACH

L'istruzione FOR EACH esegue un ciclo enumerando contemporaneamente un oggetto. L'espressione deve essere un riferimento a un oggetto enumerabile come una raccolta o una matrice. L'ordine di enumerazione non è necessariamente prevedibile. Il formato generale della dichiarazione è:

```
PER OGNI espressione ... AVANTI  
FOR EACH Variable IN Expression ... NEXT
```

Questa sintassi deve essere utilizzata quando Expression è un oggetto enumerabile che non è un contenitore. Ad esempio, un oggetto restituito come risultato di una query sul database. Le istruzioni precedenti creano un nuovo oggetto di raccolta. Immettere il codice seguente nella finestra del codice della console ed eseguirlo.

```
STATICO PUBBLICO SUB Principale ()  
  
DIM MyDict AS NEW Collection  
DIM strElement AS String
```

The purpose of this document is to provide a guide for beginners to the VBA programming language. It is intended to be a comprehensive introduction to the language, covering basic syntax, data types, control structures, and common functions. The document is written in Italian and is based on the Microsoft Visual Basic for Applications (VBA) documentation. The code examples are provided in VBA syntax, which is used in Microsoft Office applications like Excel and Word. The document is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike license, and may not be reproduced or distributed without express written consent of the author.

Una guida per principianti a
MyDict ["absolute"] = 3

This product is (C) 2005 by John Willoughby, all rights are reserved. It is released to the Gambas User Community under a Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
MyDict ["basic"] = 1
MyDict ["carpet"] = 2

PER OGNI strElement IN MyDict
    PRINT strElement & "";
FINE
SUCC
ESSI
VA
```

La tua console dovrebbe visualizzare questo output:

```
3 1 2
```

A questo punto, dovresti avere un'idea abbastanza buona della sintassi e della struttura di Gambas. Ci allontaneremo dalle applicazioni basate su console (terminale) e torneremo al nostro primo progetto per iniziare a esplorare Gambas ToolBox e imparare a sviluppare un programma basato su GUI. È ora di fare una pausa e quando torni al capitolo successivo, sarai aggiornato e pronto per programmare. Ah, diamine. Se non vedi l'ora di entrare, vai avanti e gira pagina!



Una guida per principianti a

Capitolo 4 - Presentazione di Gambas ToolBox

Il Gambas ToolBox predefinito è costituito da molti controlli. Al momento della stesura di questo documento utilizzando la versione (1.0.9), sono attualmente supportati i seguenti controlli:

- ✓ Pulsante
- ✓ Etichetta
- ✓ TextLabel
- ✓ Casella di testo
- ✓ TextArea
- ✓ CheckBox
- ✓ ComboBox
- ✓ ListBox
- ✓ RadioButton
- ✓ ToggleButton
- ✓ Telaio
- ✓ Pannello
- ✓ TabStrip
- ✓ ProgressBar
- ✓ Immagine
- ✓ Timer
- ✓ Comporre
- ✓ SpinBox
- ✓ ScrollBar
- ✓ Slider
- ✓ LCDNumber
- ✓ ListView
- ✓ TreeView
- ✓ IconView
- ✓ Vista a griglia
- ✓ ColumnView
- ✓ ScrollView
- ✓ DrawingArea
- ✓ GambasEditor
- ✓ TableView
- ✓ Area di lavoro

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Come puoi vedere, c'è un'ampia selezione di strumenti con cui giocare

Una guida per principianti a

Gambas. La nostra introduzione agli strumenti (che sono anche chiamati controlli) ti insegnereà cosa fa ogni controllo, quali proprietà, metodi ed eventi puoi usare per dirigere il comportamento dell'interfaccia che stai costruendo e come codificare il controllo da creare funziona esattamente come desideri. Lungo la strada, costruiremo diversi semplici progetti per farti acquisire maggiore esperienza nell'utilizzo dell'IDE di Gambas. Tutti i controlli nel ToolBox sono incorporati utilizzando il componente `gb.qt`. Questo componente implementa le classi dell'interfaccia utente grafica. Si basa sulla libreria QT. Di seguito è riportato un elenco delle varie funzionalità fornite dal componente `gb.qt`:

- ✓ Appunti
- ✓ Contenitori
- ✓ Drag and Drop
- ✓ Disegno
- ✓ Caratteri
- ✓ IconView
- ✓ Tastiera e mouse
- ✓ Menu
- ✓ ListView, TreeView e ColumnView
- ✓ Stampa

Tratteremo altri componenti più avanti in questo libro. Nella parte superiore della Figura 15, viene visualizzata la parola Modulo. Questo è il toolkit predefinito per il componente `gb.qt`. Per ora, diamo di nuovo una rapida occhiata alla casella degli strumenti e rivediamo le icone in essa.

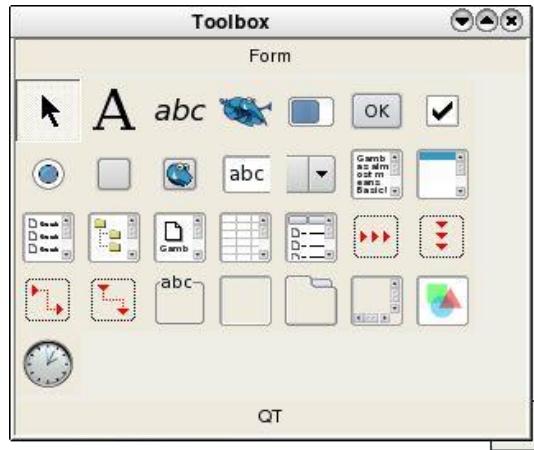


Figura 15 La casella degli strumenti di Gambas.

© 2005 by John W. Rittinghouse. All rights are reserved. It is released to the community under an Open Content License (OCL) and may not be distributed or modified without the express written consent of the author.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Dall'angolo in alto a sinistra della Figura 1, i controlli nella casella degli strumenti (riga 1) sono:

- ✓ Selezionare
- ✓ Etichetta
- ✓ TLabel
- ✓ PictureBox
- ✓ ProgressBar
- ✓ Pulsante
- ✓ CheckBox

Nella seconda riga troverai:

- ✓ RadioButton
- ✓ ToggleButton
- ✓ ToolButton
- ✓ Casella di testo
- ✓ ComboBox
- ✓ TextArea
- ✓ ListBox

Nella terza riga abbiamo:

- ✓ ListView
- ✓ TreeView
- ✓ IconView
- ✓ Vista a griglia
- ✓ ColumnView
- ✓ HBox
- ✓ VBox

Viene visualizzata la quarta riga di controlli:

- ✓ HPanel
- ✓ VPanel
- ✓ Telaio
- ✓ Pannello
- ✓ TabStrip
- ✓ ScrollView
- ✓ DrawingArea
- ✓ Timer

Una guida per principianti a

Nella parte inferiore della finestra della casella degli strumenti, viene visualizzata la parola QT. Se fai clic su questo, vedrai apparire i controlli specifici dell'interfaccia utente QT, simili alla Figura 2 di seguito:

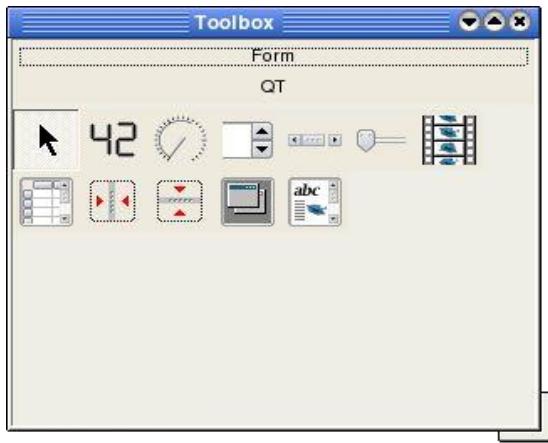


Figura 16 Controlli aggiuntivi per QT.

I controlli QTSpecific che si trovano nella riga superiore della Figura 2 sono:

- ✓ Selettore
- ✓ LCDNumber
- ✓ Comporre
- ✓ SpinBox
- ✓ ScrollBar
- ✓ Slider
- ✓ MovieBox

La seconda riga di controlli include questi strumenti:

- ✓ TableView
- ✓ HSplit
- ✓ VSplit
- ✓ Area di lavoro
- ✓ TextView

L'approccio generale alla programmazione di Gambas consiste nel progettare il layout dei moduli, posizionare i controlli che compongono l'interfaccia per l'utente sul modulo, determinare e gestire i vari eventi che possono verificarsi con ciascun controllo utilizzando una combinazione di metodi Gambas incorporati e il proprio codice e dirigere le operazioni del programma utilizzando i dati di input / output. Se sembra semplice, lo è

Una guida per principianti a

perché una volta che hai compreso i meccanismi di tutto ciò, è semplice. Una volta che ti sarai abituato a programmare applicazioni guidate da eventi in Gambas, diventerà una seconda natura per te.

Il primo gruppo di controlli che apprenderemo è di natura orientata al testo e serve per visualizzare o raccogliere dati di testo o per generare un evento per il quale è possibile scrivere codice, come premere un pulsante. Questi controlli sono:

- ✓ Pulsante
- ✓ Etichetta
- ✓ TextLabel
- ✓ Casella di testo
- ✓ InputBox
- ✓ TextArea

Il controllo dei pulsanti

Questo oggetto classe eredita i suoi attributi dalla classe Control, così come tutti i controlli nel Gambas ToolBox. L'utilizzo di questo controllo implementa un pulsante in un form. Un pulsante può visualizzare testo, un'immagine o entrambi. Hai la possibilità di impostare un pulsante all'interno di una finestra come pulsante predefinito. Quindi, quando l'utente preme il tasto RETURN, attiverà automaticamente quel pulsante. Inoltre, è possibile impostare un pulsante all'interno di una finestra come pulsante Annulla. Premendo il tasto ESC si attiverà automaticamente il pulsante Annulla. Questa classe è creabile, il che significa che puoi scrivere codice per generare dinamicamente un pulsante su un modulo in fase di esecuzione. Per dichiarare un oggetto pulsante, viene utilizzato il formato seguente:

```
DIM hButton Pulsante AS  
hButton = pulsante NUOVO (contenitore AS principale)
```

Il contenitore, nella maggior parte dei casi, sarà il modulo in cui posizionare il pulsante. Potrebbe essere qualsiasi oggetto contenitore, tuttavia. Questo codice creerà un nuovo controllo pulsante. Notare che il nome della variabile è preceduto dalla lettera minuscola h. Questa è una convenzione standard che i programmati usano per fare riferimento all'handle (gli handle sono in realtà riferimenti a qualcosa) di un oggetto. Poiché Gambas supporta la programmazione orientata agli oggetti, introdurremo gradualmente i concetti di OO (Object Oriented) man mano che li incontriamo. OO sarà trattato in modo molto più dettagliato nel capitolo 11 di questo libro. In genere, gli oggetti vengono creati da una classe assegnando una copia dell'oggetto a una variabile, in questo caso hButton. Questo processo, noto

Una guida per principianti a come creazione di un'istanza di un oggetto, si verifica quando a hButton viene assegnato un valore utilizzando NEW

Una guida per principianti a

parola chiave. Ciò che la riga sotto significa è che hButton verrà istanziato come un oggetto NEW Button e Parent è la proprietà (di sola lettura) che leggerà un metodo (pensa ai metodi come funzioni che impostano o ottengono un valore) implementato all'interno della classe Container. La classe Container viene utilizzata perché è la classe padre di ogni controllo che può contenere altri controlli. Ora, questa riga di codice dovrebbe avere perfettamente senso per te:

```
hButton = pulsante NUOVO (contenitore AS principale)
```

Proprietà di controllo comuni

Le proprietà dei pulsanti sono attributi che è possibile impostare o leggere utilizzando la finestra delle proprietà o utilizzando il codice nel programma. Ad esempio, la riga di codice seguente imposterà la proprietà Text di un pulsante:

```
hButton.Text = "OK"
```

Nota il formato della dichiarazione sopra. Control.Property = Expression è la convenzione standard utilizzata per impostare o ottenere informazioni sugli attributi da un controllo. Molte delle proprietà dei controlli sono comuni a tutti i controlli, quindi ci dedicheremo ora a spiegare qui tutte le proprietà dei pulsanti. Più avanti nel libro, spiegheremo solo le proprietà che sono univoche per un dato controllo quando le incontriamo.

Colore di sfondo è definito come PROPRIETÀ BackColor AS Integer

Questo valore intero rappresenta il colore utilizzato per lo sfondo del controllo. È sinonimo della proprietà Background. Tieni presente che PROPRIETÀ è un tipo di dati predefinito utilizzato internamente in Gambas. È possibile utilizzare le costanti predefinite di Gambas per il colore per impostare valori di colore comuni:

```
NeroBluCianoBlu scuro  
DarkCyanDarkGrayDarkGreenDarkMagenta  
DarkRedDarkYellow  
DefaultGray  
Verde Grigio Chiaro Magenta Arancione  
Rosa Rosso Trasparente Viola Bianco  
Giallo
```

Per impostare la proprietà BackColor per hButton su rosso, dovrresti usare questo codice:

```
hButton.BackColor = Color.Red
```

Una guida per principianti a

In alternativa, se conosci i valori RGB (rosso, verde, blu) o HSV (tonalità, saturazione, valore) per un colore specifico, Gambas fornisce un mezzo per convertire quei valori in un valore intero che può essere passato a BackColor (o altra proprietà correlata al colore). La classe Color fornisce due metodi, RGB e HSV che puoi utilizzare. Ecco come vengono definite queste funzioni:

```
FUNZIONE STATICA RGB (R AS Integer, G AS Integer, B AS Integer [, Alpha AS Integer] ) AS Integer
```

La funzione RGB restituisce un valore di colore dai suoi componenti rosso, verde e blu. Per la funzione HSV, usa questo:

```
FUNZIONE STATICA HSV (Hue AS Integer, Sat AS Integer, Val AS Integer) AS Integer
```

HSV restituisce un valore di colore dai suoi componenti di tonalità, saturazione e valore. Per utilizzare una di queste funzioni per impostare il colore di sfondo del pulsante, ecco cosa potresti codificare:

```
hButton.BackColor = Color.RGB (255,255,255)
```

Questo imposterebbe il colore di sfondo del pulsante su bianco. Ciò è particolarmente utile quando si tenta di impostare colori i cui valori non rientrano nei valori di costanti di colore predefiniti forniti con Gambas.

Confine Questa proprietà è un altro attributo comune trovato su molti controlli. Questa classe è statica. Le costanti utilizzate dalla proprietà Border includono quanto segue: Etched, None, Plain, Raised e Sunken. Per impostare la proprietà border per il nostro pulsante, potremmo usare questo codice:

```
hButton.Border = Border.Plain
```

Annulla è definito come PROPERTY Cancel AS Boolean e viene utilizzato per stabilire se il pulsante viene attivato o meno quando viene premuto il tasto ESC. Il codice seguente attiverà questo pulsante se l'utente preme il tasto ESC sulla tastiera.

```
hButton.Cancel = TRUE
```

Se la proprietà Annulla viene utilizzata nel tuo programma, ti consentirebbe di gestire l'evento e di uscire con garbo dal modulo proprio come se l'utente avesse fatto clic su un pulsante Esci, Esci o Annulla.

This product is (C) 2004 John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Didascalia è definito come PROPERTY Caption AS String e la proprietà Caption è sinonimo della proprietà Text. Il codice seguente

```
hButton.Caption = "OK"
```

equivale a

```
hButton.Text = "OK"
```

e può essere utilizzato in modo intercambiabile. Proviamo un po 'di codice. Avvia Gambas e apri il progetto che abbiamo creato chiamato FirstProject. Dal TreeView, fare doppio clic con il cursore del mouse sul modulo MainForm e il modulo vuoto si aprirà. Fai doppio clic in un punto qualsiasi del modulo vuoto e vedrai la finestra del codice. Dovrebbe sembrare come questo:

```
"File di classe Gambas
PUBLIC SUB Form_Open()
()

FINE
```

Ora, inserisci questo codice tra PUBLIC SUB Form_Open () e END dichiarazioni così assomiglia a questo:

```
PUBLIC SUB Form_Open ()
    DIM hButton Pulsante AS
    hButton = NUOVO pulsante (ME) COME "hButton"

    hButton.X = 215
    hButton.Y = 60
    hButton.Width = 200
    hButton.Height = 40
    hButton.Enabled = TRUE
    hButton.Text = "Fai clic o premi ESC per uscire"
    hButton.Border = TRUE
    hButton.Default = TRUE
    hButton.Cancel = TRUE
    hButton.Show

FINE
```

Dopo l'istruzione END di PUBLIC SUB Form_Open (), aggiungi questa routine:

```
PUBLIC SUB hButton_Click
    () MainForm.Close
FINE
```

Una guida per principianti a

Dopo aver inserito il codice sopra, fai clic sul pulsante verde di esecuzione del programma e dovresti vedere qualcosa di simile al seguente:



Figura 17 Risultati del codice del primo pulsante.

Fare semplicemente clic sul pulsante o premere il tasto ESC per uscire dal programma. Congratulazioni! Hai appena creato il tuo primo programma basato su GUI in Gambas! Sì, è un po 'brutto ma ricorda, l'hai fatto nel modo più duro, usando il codice invece di sfruttare Gambas ToolBox. Ricorda, un buon programmatore proverà sempre a utilizzare costanti predefinite se sono disponibili invece di valori numerici di hardcoding. Ad esempio, i codici dei tasti e gli stili del mouse hanno gli stessi nomi di quelli definiti come costanti, ma i loro valori numerici sono completamente diversi. Il componente gb.gtk anticipata non funzionerà con il codice gb.qt se non si utilizzano costanti predefinite, quindi durante la scrittura del codice è imperativo utilizzare queste costanti invece dei valori numerici. In caso contrario, il codice non sarà portatile. Un buon codice non dovrebbe mai utilizzare valori numerici per queste attività. Adesso,

Cursore è definito come PROPERTY Cursor AS Cursor ed è possibile utilizzare la proprietà Cursor per assegnare un cursore personalizzato a un controllo. Questa classe implementa un cursore del mouse personalizzato. Questa classe è creabile. Ecco come dichiarare una variabile oggetto cursore e istanziare un oggetto cursore:

```
DIM hCursor AS Cursor  
hCursor = NEW Cursor (MyPic AS Picture [, X AS Integer, Y AS Integer])
```

Il formato del codice sopra può essere utilizzato per creare un nuovo cursore da un oggetto Immagine e impostare un punto attivo opzionale. Se non è specificato, l'hotspot predefinito è l'angolo superiore sinistro dell'immagine. Il cursore ha due proprietà, X e Y, che possono essere utilizzate per impostare o ottenere la posizione corrente del mouse. Ecco un esempio che mostra come utilizzare un oggetto Cursor nel tuo programma:

Una guida per principianti a

```
DIM hPict AS Picture
DIM hCursor AS Cursor
hPict = Immagine
["torric.png"] hCursor =
NUOVO Cursore (hPict)
ME.Cursor = hCursor
```

Predefinito è definito come PROPERTY Default AS Boolean e si utilizza la proprietà Default per indicare se il pulsante viene attivato o meno ogni volta che si preme il tasto RETURN. Utilizzando il codice seguente, costringerebbe l'utente a fare effettivamente clic su un pulsante anziché premere il tasto INVIO per attivarlo. In genere, quando si crea una finestra di dialogo di tipo OK / Annulla, si desidera impostare uno dei pulsanti su un'azione predefinita utilizzando questa proprietà. Pensa di premere semplicemente il tasto INVIO in una procedura guidata per accettare le impostazioni predefinite e andare avanti.

```
hButton.Default = FALSE
```

Design è definito come PROPERTY Design AS Boolean e indica che il controllo è in modalità progettazione. Ciò significa che quando la proprietà Design è impostata su TRUE, i controlli in modalità progettazione si limitano a disegnare se stessi e non reagiranno a nessun evento di input. Questa proprietà viene utilizzata dall'ambiente di sviluppo Gambas per visualizzare i controlli nell'editor del modulo. Ciò che significa veramente è che non devi preoccuparti di questa proprietà.

Far cadere è definito come PROPERTY Drop AS Boolean e viene utilizzato per verificare se un controllo accetta il rilascio da un'operazione di trascinamento. La proprietà Drop è TRUE o FALSE e puoi controllare il valore prima di provare a rilasciare qualcosa sul controllo usando un codice come questo:

```
IF hButton.Drop = FALSE THEN
    hButton.ToolTip = "Nessun rilascio
    consentito"
ALTRO
    hButton.ToolTip = "Rilascia qualcosa
qui" ENDIF
```

Abilitato è definito come PROPERTY Enabled AS Boolean e indica che il controllo è abilitato. Per disabilitare un controllo, è sufficiente impostare la sua proprietà Enabled su FALSE.

```
IF Expression = FALSE THEN
    hButton.Enabled = TRUE
ALTRO
```

Una guida per principianti a

```
hButton.Enabled = FALSE  
ENDIF
```

Una guida per principianti a

La proprietà Expand è definita come PROPERTY Expand AS Boolean e indica che il controllo è in grado di espandersi se è incluso in un contenitore che riorganizza dinamicamente il suo contenuto.

Font è definito come PROPERTY Font AS Font e restituisce o imposta il carattere utilizzato per disegnare il testo nel controllo. Per impostare gli attributi della proprietà Font, è possibile utilizzare codice come questo:

```
hButton.Font.Name = "Lucida"  
hButton.Font.Bold = TRUE  
hButton.Font.Italic = FALSE  
hButton.Font.Size = "10"  
hButton.Font.StrikeOut = FALSE  
hButton.Font.Underline = FALSE
```

L'output del codice precedente sarebbe il seguente:



Figura 18 Dimostrazione delle capacità dei caratteri.

ForeColor è definito come PROPERTY ForeColor AS Integer e questo valore intero rappresenta il colore utilizzato per il controllo in primo piano. È sinonimo della proprietà Primo piano. È possibile utilizzare le costanti predefinite di GambaSAS (OCL) per impostare il colore per il colore per impostare il valore del colore:

```
NeroBluCianoBlu scuro  
DarkCyanDarkGrayDarkGreenDarkMagenta  
DarkRedDarkYellow  
DefaultGray  
Verde Grigio Chiaro Magenta Arancione  
Rosa Rosso Trasparente Viola Bianco  
Giallo
```

Per impostare la proprietà ForeColor per hButton su rosso, dovresti usare questo codice:

```
hButton.ForeColor = Color.Red
```

Come con la proprietà BackColor, se conosci i valori RGB o HSV per un colore specifico, puoi passarli alla proprietà ForeColor con questo codice:

```
hButton.ForeColor = Color.RGB (255,255,255)
```

Una guida per principianti a

Ciò imposterebbe il colore di primo piano del pulsante su bianco.

La proprietà X è definita come PROPERTY X AS Integer e viene utilizzata per impostare o restituire la posizione x del controllo. X è uguale alla proprietà Left. La proprietà Y è definita come PROPRIETÀ Y AS Integer e viene utilizzata per impostare o restituire la posizione y del controllo (angolo in alto a sinistra).

La proprietà Height è definita come PROPERTY Height AS Integer e viene utilizzata per impostare o restituire l'altezza del controllo. È sinonimo della proprietà H e può essere utilizzato in modo intercambiabile. Questo valore descrive quanto è alto il controllo dalla posizione x che si sposta verso il basso Altezza pixel. La proprietà Width è definita come PROPERTY Width AS Integer e viene utilizzata per impostare o restituire la larghezza del controllo. È sinonimo della proprietà W e può essere utilizzato in modo intercambiabile. Questo valore descrive la larghezza del controllo dalla posizione x ai pixel di larghezza a destra.

La proprietà Handle è definita come PROPERTY READ Handle AS Integer e viene utilizzata per restituire l'handle di finestra X11 interno del controllo. È un attributo di sola lettura. È sinonimo di ID (ID PROPRIETÀ READ AS Integer).

La proprietà Mouse è definita come PROPERTY Mouse AS Integer e viene utilizzata per impostare o restituire l'aspetto del cursore a un'immagine predefinita quando si trova all'interno dei limiti del controllo. I valori predefiniti per la proprietà Mouse sono mostrati nella tabella seguente:

| Forma del cursore | Valore |
|-------------------|--------|
| Predefinito | 1 |
| Freccia | 0 |
| Attraversare | 2 |
| Aspettare | 3 |
| Testo | 4 |
| TagliaS | 5 |
| Taglia E. | 6 |
| TagliaNESW | 7 |
| TagliaNWSE | 8 |
| SizeAll | 9 |
| Vuoto | 10 |
| SplitV | 11 |
| SplitH | 12 |
| Indicando | 13 |

Una guida per principianti a

La proprietà Next è definita come PROPERTY READ Next AS Control e viene utilizzata per restituire il controllo successivo con lo stesso genitore all'interno di un Container. È utile durante l'iterazione di una serie di controlli per impostare lo stato attivo. Discuteremo questa proprietà in modo più dettagliato quando discuteremo del metodo SetFocus (). Per ottenere il controllo precedente, si utilizzerebbe la proprietà Previous, definita come PROPERTY READ Previous AS Control. Questa proprietà restituisce il controllo precedente con lo stesso genitore.

La proprietà Parent è definita come PROPERTY READ Parent AS Control e viene utilizzata per restituire il contenitore che contiene il controllo. È una proprietà di sola lettura.

La proprietà Picture è definita come PROPERTY Picture AS Picture e restituisce o imposta l'immagine visualizzata su un controllo (ad esempio, un pulsante). Discuteremo la classe Picture più avanti in questo capitolo.

La proprietà ScreenX è definita come PROPERTY READ ScreenX AS Integer e restituisce la posizione del bordo sinistro del controllo nelle coordinate dello schermo. Di solito, viene utilizzato insieme a ScreenY.

La proprietà ScreenY è definita come PROPERTY READ ScreenX AS Integer e restituisce la posizione del bordo superiore del controllo nelle coordinate dello schermo.

La proprietà Tag è definita come PROPERTY Tag AS Variant e viene utilizzata per restituire o impostare il tag di controllo. Questa proprietà è destinata all'uso da parte del programmatore e non viene mai utilizzata dal componente. Può contenere qualsiasi valore Variant.

La proprietà Text è definita come PROPERTY Text AS String e restituisce o imposta il testo visualizzato nel pulsante. Viene quasi sempre utilizzato e il suo valore può essere modificato dinamicamente. Abbiamo già utilizzato questa proprietà nell'esempio sopra:

```
hButton.Text = "Fai clic o premi ESC per uscire"
```

La proprietà ToolTip è definita come PROPERTY ToolTip AS String e restituisce o imposta il tooltip (un tooltip è la piccola casella di testo che si apre su un controllo quando il mouse vi passa sopra per alcuni secondi).

```
hButton.ToolTip = "Cliccami!"
```

Una guida per principianti a

La proprietà Top è definita come PROPERTY Top AS Integer e restituisce o imposta la posizione del bordo superiore del controllo rispetto al suo genitore.

La proprietà Value è definita come PROPERTY Value AS Boolean e viene utilizzata per attivare il controllo (ovvero, fare clic sul pulsante). Ciò viene eseguito se si imposta questa proprietà su TRUE. La lettura di questa proprietà restituirà sempre un valore FALSE.

La proprietà Visible è definita come PROPERTY Visible AS Boolean e indica se il controllo è visibile o meno. L'impostazione di questa proprietà su FALSE la farà scomparire dalla visualizzazione. Al contrario, impostarlo su TRUE lo fa apparire.

La proprietà Window è definita come PROPERTY READ Window AS Window e restituisce la finestra di primo livello che contiene il controllo.

A questo punto, abbiamo coperto tutte le proprietà per il controllo del pulsante e dovresti avere una buona comprensione di cosa vengono utilizzate e come usarle. Ricorda che la maggior parte di queste proprietà sono comuni a molti controlli. Successivamente, tratteremo i metodi (cose che puoi fare con il pulsante) e ti forniremo alcuni esempi di come usarli.

Metodi dei pulsanti

I metodi sono le routine incorporate fornite per un controllo che ti consentono di fare cose come mostrarlo, nasconderlo, spostarlo, ecc. Ecco i metodi supportati dal controllo Button: Elimina, Trascina, Afferra, Nascondi, Abbassa, Sposta, Alza, Aggiorna, Ridimensiona, Imposta messa a fuoco e Mostra. Di solito, un metodo viene chiamato in risposta a qualche evento. Discuteremo gli eventi nella prossima sezione. Diamo uno sguardo ai metodi usati con Button:

Il metodo Delete è definito come SUB Delete () e quando viene richiamato distrugge il controllo. È importante sapere che un controllo distrutto diventa un oggetto non valido. Ciò significa che dopo averlo distrutto, assicurarsi che nessun altro codice vi faccia riferimento successivamente.

Il metodo Drag è definito come SUB Drag [Data AS Variant [, Format AS String]] e quando invocato avvia un processo di drag & drop. I dati sono i dati da trascinare. Può essere una stringa o un'immagine. Se Data è una stringa di testo, è possibile specificare in Formato il TIPO MIME del testo trascinato. Ad esempio,

Una guida per principianti a
TIPO MIME "text / html".

Una guida per principianti a

Il metodo Grab è definito come FUNCTION Grab () AS Picture e acquisisce un'immagine del controllo e la restituisce.

Il metodo Hide e il metodo Show sono definiti come SUB Hide () e SUB Show () e semplicemente nascondono o mostrano il controllo. Ecco come verrebbero utilizzati:

```
IF Expression = TRUE THEN
    hButton.Hide
ALTRO
    hButton.Show
ENDIF
```

Il metodo Lower e il metodo Raise sono definiti come SUB Lower () e SUB Raise (). Lower invia il controllo allo sfondo del suo genitore. Raise porta il controllo in primo piano rispetto al suo genitore. Come esempio di come funzionano questi metodi, torniamo al nostro progetto, FirstProject. Modificare il codice nella routine hButton_Click () per leggere come segue:

```
PUBLIC SUB hButton_Click
    () Button1.Hide
    Label1.Text = "Ouch!"
    WAIT 2.0
    Button1.Show
    Label1.Text = "Meglio
adesso!" FINE
```

Una volta inserito il codice, esegui il programma e guarda cosa succede. Quando fai clic sul pulsante, scomparirà e Label1.Text cambierà. Dopo due secondi, riapparirà e cambierà di nuovo l'etichetta. Inoltre, tieni presente che abbiamo inserito un nuovo comando da provare: ATTENDI. WAIT accetta un valore in virgola mobile come parametro che rappresenta i secondi o le frazioni di secondo espresse come valori decimali. I numeri inferiori a 1 vengono elaborati come millisecondi. WAIT 1.5 sospenderebbe l'esecuzione per 1.500 millisecondi. Se si specifica WAIT senza un parametro, verrà aggiornata l'interfaccia e non consentirà all'utente di interagire con l'applicazione.

Il metodo Move è definito come SUB Move (X AS Integer, Y AS Integer [L Width AS Integer, Height AS Integer]) e viene utilizzato per spostare e / o ridimensionare il controllo.

```
hButton.Move (hButton.X 50, hButton.Y 20)
```

Una guida per principianti a

Il metodo Refresh è definito come SUB Refresh ([X AS Integer, Y AS Integer, Width AS Integer, Height AS Integer]) e viene utilizzato per ridisegnare il controllo o, in alcuni casi, solo una parte di esso.

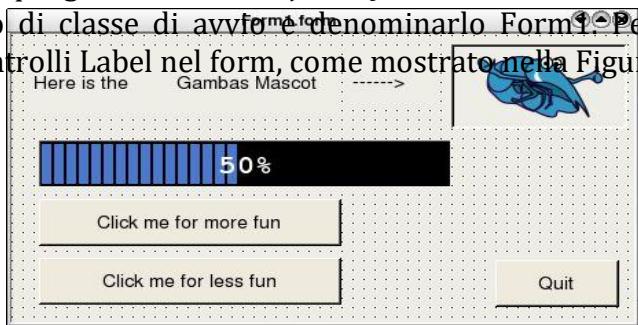
Il metodo Resize è definito come SUB Resize (Width AS Integer, Height AS Integer) e viene utilizzato per ridimensionare il controllo.

Il metodo SetFocus è definito come SUB SetFocus () e si chiama questo metodo per dare lo stato attivo al controllo. Lo stato attivo è più o meno identico al punto in cui sarebbe posizionato il cursore del testo. Quando un controllo è attivo, di solito viene indicato visivamente in qualche modo. Ad esempio, un pulsante con lo stato attivo può avere una piccola linea tratteggiata attorno ad esso per indicare dove si trova il controllo dello stato attivo nel modulo:



Figura 19 La linea tratteggiata indica lo stato attivo per un controllo.

Abbiamo coperto tutti i metodi disponibili per la classe Button (controllo) e ora inizieremo a discutere gli eventi per quella classe. Come cambio di ritmo, proviamo a conoscere gli eventi con un approccio più pratico. Possiamo farlo costruendo un nuovo programma e imparando cosa fanno gli eventi e come usarli. Chiudi Gambas (una piccola particolarità di Gambas è la mancanza di una selezione del menu di progetto ravvicinata) e riavvia per avviare la procedura guidata Nuovo progetto. Seleziona un progetto di interfaccia utente grafica e segui la procedura guidata proprio come abbiamo fatto con il progetto FirstProject. Assegna un nome a questo progetto SecondProject. Quando viene visualizzato l'IDE, creare un nuovo modulo di classe di avvio e denominarlo Form1.gambas. Per questo progetto, inseriremo tre controlli Label nel form, come mostrato nella Figura 20 di seguito.



Una guida per principianti a
Figura 20 Il layout per SecondProject Form1.form.

Una guida per principianti a

La didascalia di testo della prima etichetta viene impostata con la proprietà Label.Text. L'impostazione dovrebbe essere "Here is the" e la seconda è "Gambas Mascot" e la terza è "

> "Che punta a un controllo PictureBox (dove vive il gambero). Assegnare un nome alle etichette Label1, Label2 e Label3 rispettivamente. Quindi, denominare il controllo PictureBox PictureBox1.

Per ottenere la grafica della mascotte di Gambas, è possibile visitare il sito web ufficiale di Gambas13 e fare clic con il tasto destro sulla mascotte. Scegli di salvare l'immagine e di usarla come "gambas mascot.png" sul tuo computer. Spostare il file nella directory dei dati nella cartella SecondProject. (Un'altra particolarità di Gambas è che le finestre di dialogo del file explorer non sembrano essere dinamiche. Una volta inizializzate, le modifiche apportate al di fuori dell'ambiente Gambas non vengono aggiornate dinamicamente.) Dopo averlo fatto, puoi impostare la proprietà Picture per PictureBox1 su il file denominato "gambas mascot.png". A questo punto, la parte superiore del modulo dovrebbe essere simile a questa:

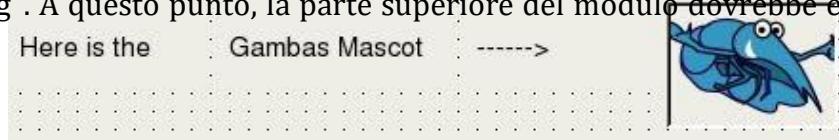


Figura 21 Un modulo parzialmente costruito con i nostri primi quattro controlli.

Ora, seleziona il controllo ProgressBar dalla casella degli strumenti e posizionalo nel form in modo che assomigli a quello mostrato nella Figura 20 sopra. Infine, abbiamo tre pulsanti da aggiungere al modulo. Assegna a questi pulsanti il nome FunBtn, NoFunBtn e QuitBtn. Tutti i controlli necessari per il nostro esempio sono stati aggiunti al modulo a questo punto. Cerca di far sembrare il layout il più vicino possibile all'immagine mostrata nella Figura 3 sopra. A questo punto, potresti chiederti cosa porterà a termine tutto questo? Ricorda, impareremo come utilizzare gli eventi con questi controlli. Abbiamo posizionato i controlli dove vogliamo e ora il passo successivo è decidere quali eventi gestiremo.

Per questo esercizio, gestiremo gli eventi del mouse quando il mouse si sposta sulle etichette. Se il mouse si sposta fuori dalla finestra principale (Form1.form) o di nuovo su di essa, gestiremo quegli eventi. Gestiremo anche i clic dei pulsanti e il tasto ESC se premuto.

Iniziamo impostando ProgressBar1.Value su zero all'avvio del programma. Questo viene gestito con la routine di inizializzazione quando si apre il modulo in

Una guida per principianti a
fase di esecuzione del programma, come mostrato di seguito:

13 Il sito ufficiale di Gambas è <http://gambas.sourceforge.net/index.html> .

Una guida per principianti a

```
"File di classe Gambas

PUBLIC SUB Form_Open ()
    ProgressBar1.Value = 0,50
    ProgressBar1.BackColor = Nero
    ProgressBar1.ForeColor = Giallo
FINE
```

This product is sold "as is". W. Ritterhouse, all rights reserved. It is released to the Gambas User under an Open Content License (OCL) and may not be distributed or modified in any other way without the express written consent of the author.

Quando il mouse si sposta sul form di avvio (il nostro programma Gambas sul desktop), vogliamo impostare le proprietà di testo delle tre etichette in alto su valori vuoti in modo che sembri che il testo scompaia. Ciò richiede la scelta di un gestore di eventi. Per scegliere un evento dall'IDE Gambas, sposta il mouse sul modulo (ma non direttamente su un controllo) e fai clic una volta. Quindi, fai clic con il pulsante destro del mouse e dovresti vedere un menu a comparsa, simile a quello mostrato nella Figura 22 sotto. Scegli la voce Selezione evento dal popup e seleziona Inserisci evento. Aggiungi questo codice, come mostrato di seguito:

```
PUBLIC SUB Form_Enter()
    Label1.Text = ""
    Label2.Text = ""
    Label3.Text = ""
FINE
```

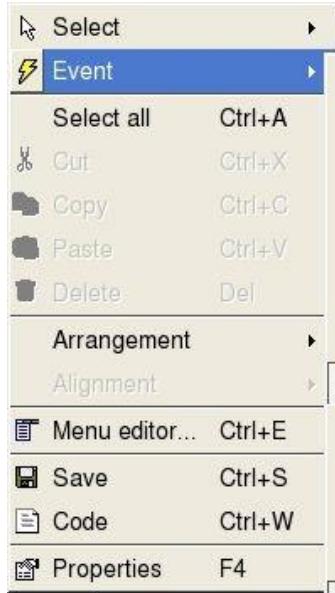


Figura 22 Menu Evento.

Ripeti questo processo per scegliere un altro evento, l'evento Leave, in modo che quando il mouse si sposta fuori dalla finestra del programma possiamo rilevarlo e fare qualcosa al riguardo. Aggiungi questo codice:

This product is released to the public domain under an Open Content License (OCL) and may not be distributed under any other conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB Form_Leave ()  
    ProgressBar1.Value = 0.50  
    Label1.Text = "Quella"  
    Label2.Text = "non era una  
vera" Label3.Text = "roba  
divertente?"  
FINE
```

La prossima cosa che vogliamo fare è gestire la situazione (evento) che si verifica quando il mouse si sposta su una delle nostre etichette. In questo caso, cambieremo la proprietà Label.Text dell'etichetta interessata per indicare che è stato rilevato un mouse sul controllo. Fai un singolo clic con il cursore del mouse sulla prima etichetta e quando vengono visualizzate le maniglie, fai clic con il pulsante destro del mouse e scegli l'evento Inserisci quando appare nel sottomenu Evento. Nella finestra del codice, trova la subroutine Label1_Enter () e inserisci questo codice:

```
PUBLIC SUB Label1_Enter ()  
    Label1.Text = "Ecco il"  
FINE
```

Ripetere la stessa procedura per Label2.Text e Label3.Text proprietà codice dovrebbe assomigliare a questo:

```
PUBLIC SUB Label2_Enter ()  
    Label2.Text = "Gambas Mascot"  
FINE  
  
PUBLIC SUB Label3_Enter  
    () Label3.Text = ">"  
FINE
```

Vogliamo anche rilevare e rispondere agli eventi quando il mouse si sposta anche sulla nostra immagine della mascotte. Fare clic una volta sul controllo PictureBox1 e scegliere Enter Event. Vai alla finestra del codice e rendi il codice simile a questo:

```
PUBLIC SUB PictureBox1_Enter  
    () Label1.Text = ""  
    Label2.Text = ""  
    Label3.Text = ""  
  
    PictureBox1.Border = Border.Plan  
END
```

Ripeti il processo, scegliendo l'evento Lascia e assicurati che il tuo codice sia digitato

Una guida per principianti a
nell'editor di codice come segue:

```
PUBLIC SUB PictureBox1_Leave ()
```

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the Gambas UserCommunity under an Open Content License (CC) and may not be distributed under any terms or conditions without the express written consent of the author.

Una guida per principianti a

```
Label1.Text = "Quella" Label2.Text  
= "non era una vera" Label3.Text =  
"roba divertente?"  
PictureBox1.Border = Border.Sunken  
FINE
```

Il prossimo controllo di cui vogliamo occuparci è il pulsante Esci. Fare doppio clic su di esso e la finestra del codice visualizza la subroutine `QuitBtn_Click()`, che risponde a un evento Click. Aggiungere la riga `Form1.Close`, come illustrato di seguito.

```
PUBLIC SUB QuitBtn_Click  
() Form1.Close  
FINE
```

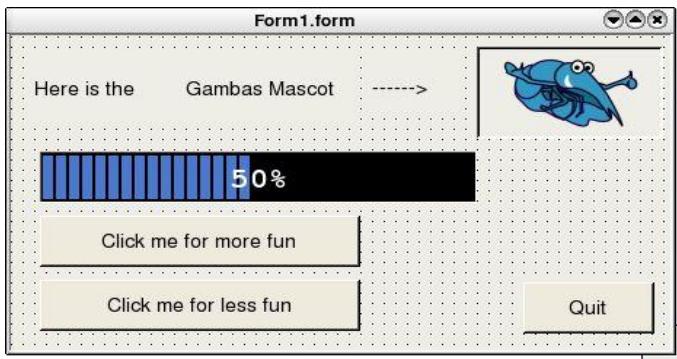


Figura 23 Aggiunta di FunBtn al nostro modulo.

Ora dobbiamo scrivere il codice per `ProgressBar`. Per il nostro esempio, lo incrementeremo quando si fa clic su `FunBtn` e lo diminuiremo quando si fa clic su `NoFunBtn`. Fai doppio clic sul controllo `FunBtn` e aggiungi questo codice:

```
PUBLIC SUB FunBtn_Click ()  
ProgressBar1.Value = ProgressBar1.Value + 0,01  
IF ProgressBar1.Value > 0,99 THEN  
    ProgressBar1.Value = 0.0  
ENDIF  
FINE
```

Ora, fai doppio clic su `NoFunBtn` e aggiungi questo:

```
PUBLIC SUB NoFunBtn_Click ()  
ProgressBar1.Value = ProgressBar1.Value - 0,01  
IF ProgressBar1.Value < 0,01 THEN  
    ProgressBar1.Value = 0.0  
ENDIF  
FINE
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Questo è quasi tutto ciò che resta da fare! Ora salva il progetto e fai clic sul pulsante Esegui. Dovresti vedere questo all'avvio del programma:

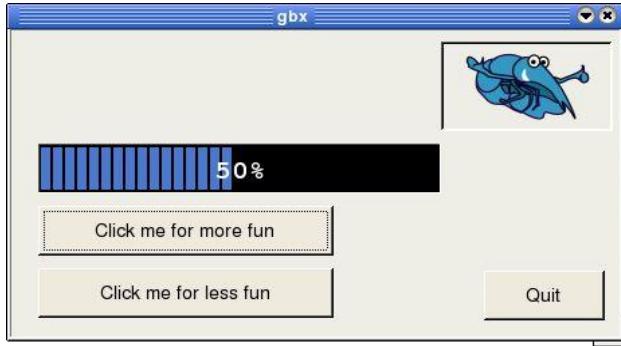


Figura 24 Come appare il nostro modulo quando viene rilevato un mouse sul modulo. Nota che il testo è cancellato.

Ora, fai clic sul pulsante "Fai clic su di me per divertirti di più" e la barra di avanzamento cambia. Fare clic tre volte e dovresti vedere questo:

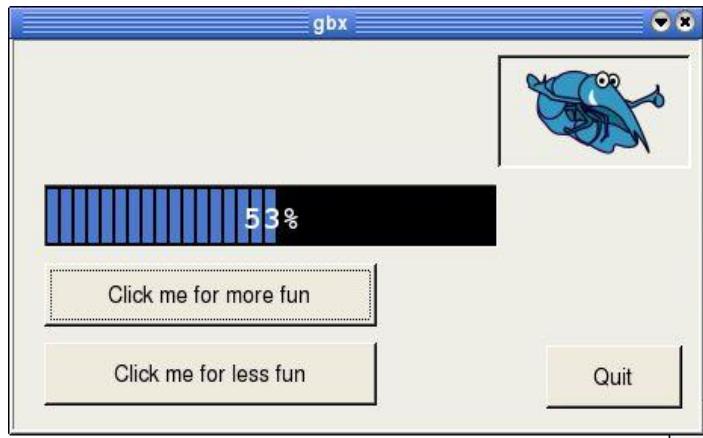


Figura 25 La barra di avanzamento quando si fa clic su FunBtn tre volte.

Se muovi il mouse sull'immagine della mascotte, dovresti vedere il testo dell'etichetta apparire e scomparire mentre muovi il mouse dentro e fuori dall'immagine. Infine, puoi premere ESC o fare clic sul pulsante Esci per uscire dal programma. L'ultimo evento per il quale scriveremo codice in questo esempio è l'evento doubleclick. Ciò richiede di scegliere ancora una volta un gestore di eventi.

Dall'IDE Gambas, sposta il mouse su FunBtn e fai clic una volta. Quindi, fai clic con il pulsante destro del mouse e vedrai il menu popup che ti consente di farlo

This product is (C) 2005 by John W. Ritter. All rights reserved. This software is released to the public domain under the terms of the GIMP General Public License (GPL). It may be freely copied and distributed for non-commercial purposes without prior permission or notice. The software is provided "as is", without any express written warranty. The author is not responsible for any damages resulting from the use of this software.

Una guida per principianti a

scegli Eventi. Dagli eventisottomenu, scegli l'evento Doppio clic. Aggiungi questo codice, come mostrato di seguito:

```
PUBLIC SUB FunBtn_DblClick ()  
    ProgressBar1.Value = ProgressBar1.Value + 0.1  
    IF ProgressBar1.Value > 0.90 THEN  
        ProgressBar1.Value = 0.0  
    ENDIF  
END
```

Ripeti questo processo per NoFunBtn e aggiungi questo codice:

```
PUBLIC SUB NoFunBtn_DblClick ()  
    ProgressBar1.Value = ProgressBar1.Value - 0.1  
    IF ProgressBar1.Value < 0.1 THEN  
        ProgressBar1.Value = 0.0  
    ENDIF  
END
```

Ora salva il progetto e fai clic sul pulsante Esegui. Questa volta, quando fai clic sulla barra di avanzamento funziona come prima, ma se fai doppio clic, aumenterà il valore di 10 unità anziché di una. Una volta uscito dal programma, salva il progetto ed esci da Gambas. Successivamente, esamineremo gli altri eventi che possono essere utilizzati con il controllo Button.

Eventi pulsante

Gli eventi Button che abbiamo utilizzato nel nostro esempio precedente includono gli eventi Click, DblClick, Enter e Leave. Riserveremo la nostra discussione sugli eventi rimanenti, vale a dire Drag, DragMove, Drop, LostFocus e GotFocus per quando ci occuperemo di operazioni di trascinamento della selezione. Gli eventi Menu, KeyPress e KeyRelease verranno trattati quando parliamo di operazioni con menu e tastiera. Infine, quando trattiamo le operazioni del mouse, discuteremo degli eventi MouseDown, MouseMove, MouseUp e MouseWheel.

La classe delle immagini

In precedenza nel capitolo, quando abbiamo utilizzato il controllo Immagine per posizionare la nostra mascotte sul modulo nel nostro secondo esempio, abbiamo menzionato la classe Immagine e abbiamo detto che l'avremmo coperta in seguito. Bene, ora è il momento di parlare della Picture Class. Questa classe rappresenta un'immagine e il suo contenuto è archiviato nel server di visualizzazione, non nella memoria di processo come sarebbe normalmente archiviata un'immagine. Anche se

Una guida per principianti a
XWindows

Una guida per principianti a

non gestisce (ancora) la trasparenza, ad ogni oggetto immagine può essere assegnata una maschera. Questo può essere impostato in modo esplicito nel momento in cui viene creata l'istanza dell'immagine, oppure può essere impostato in modo implicito durante il caricamento di un file immagine con attributi di trasparenza, come un file PNG. Quando si disegna su un'immagine con una maschera, vengono modificate sia l'immagine che la maschera. Questa classe è creabile. È definito come:

```
DIM hPicture AS Picture
```

e il processo di istanziazione utilizza questo formato:

```
hPicture = NEW Picture ([Width AS Integer, Height AS Integer,  
Transparent AS Boolean])
```

Il codice precedente creerà un nuovo oggetto immagine. Se la larghezza e l'altezza non sono specificate, la nuova immagine è vuota (nulla). È possibile specificare se l'immagine ha una maschera con il parametro Trasparente. La classe immagine si comporta come un array. Per esempio:

```
DIM hPicture AS Picture  
hPicture = Immagine [Path AS String]
```

restituirà un oggetto Picture dalla cache delle immagini interna. Se l'immagine non è presente nella cache, viene caricata automaticamente dal file specificato. Per inserire un'immagine nella cache, usa questa chiamata:

```
Immagine [Path AS String] = hPicture
```

Le proprietà per questo controllo includono Profondità, Altezza, Immagine, Trasparente e Larghezza. I metodi forniti includono Clear, Copy, Fill, Flush, Load, Resize e Save. Tratteremo l'uso di questi metodi e proprietà negli esempi successivi di questo libro. A questo punto, dovresti avere una solida conoscenza di come utilizzare l'ambiente di programmazione Gambas e i suoi vari strumenti. Nei prossimi capitoli presenteremo più controlli con esempi più complessi poiché il modo migliore per imparare è farlo.

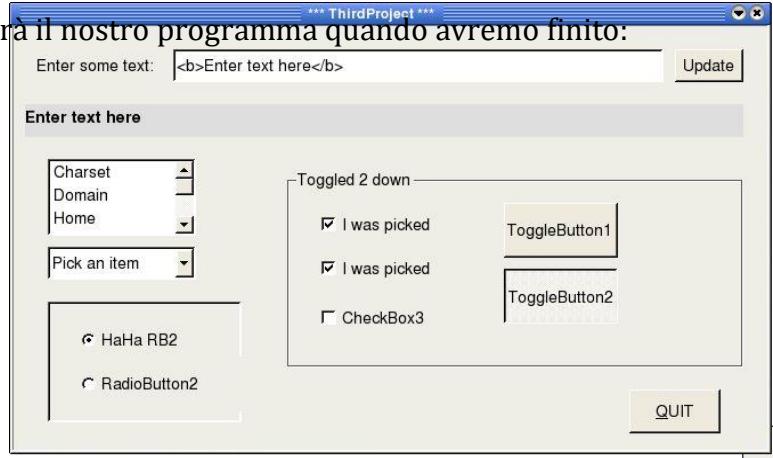
Una guida per principianti a

Capitolo 5 - Controlli per la raccolta dell'input

Nella nostra introduzione ai controlli, abbiamo appreso i tipi più elementari di controlli ed eventi, ovvero immagini, etichette, pulsanti e una barra di avanzamento. Abbiamo anche imparato come rilevare quando il mouse si trova su un controllo e quali azioni possiamo intraprendere quando si sposta o si spegne dal controllo. Questi tipi di controlli presentano tutti dati o rispondono a eventi come il mouse o il clic di un pulsante, ma non consentono agli utenti di scegliere tra alternative come selezionare elementi da un elenco o digitare il loro nome, ecc. In questo capitolo, noi aggiungerà al nostro arsenale di conoscenze Gambas imparando come svolgere questi compiti e rispondere a un numero ancora maggiore di eventi. Costruiremo un programma che ti introdurrà ai seguenti controlli e ti insegnerrà come usarli:

- ✓ TLabel
- ✓ Casella di testo
- ✓ ComboBox
- ✓ ListBox
- ✓ ToggleButton
- ✓ Pannello
- ✓ Telaio
- ✓ Casella di controllo
- ✓ RadioButton

Ecco come sarà il nostro programma quando avremo finito:



This product is copyrighted by W. Rittinghouse, all rights are reserved. It is released to the Gambas User community under the Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Figura 26 ThirdProject (quasi) risultato finale.

Una guida per principianti a

In questo esercizio impareremo a inserire titoli nella finestra del nostro programma, a utilizzare una casella di input per raccogliere input in formato HTML dall'utente e visualizzare il risultato in un nuovo tipo di etichetta, `TextLabel`. Impareremo anche come raggruppare e organizzare i nostri controlli nel modulo, utilizzando pannelli e frame, e posizionare controlli come `RadioButtons` e `Checkboxes` all'interno di quei controlli organizzativi. Infine, impareremo come utilizzare `ListBox` e `ComboBox` per consentire all'utente di selezionare una delle tante scelte. Lungo il percorso, impareremo un po' di più sulla codifica degli eventi e cercheremo di renderlo interessante. Cominciamo creando un nuovo progetto con Gambas. Avvia Gambas e scegli di creare un nuovo progetto utilizzando un'interfaccia utente grafica. Esegui la procedura guidata rendendo il progetto traducibile e i controlli del modulo pubblici. Seleziona il nome `ThirdProject` e mettilo in una directory a tua scelta. Quando l'IDE si avvia, crea un nuovo modulo di classe di avvio. Fare doppio clic sul modulo vuoto e verrà visualizzata la finestra dell'editor del codice. Dovresti vedere qualcosa del genere:

```
'File di classe Gambas

PUBLIC SUB Form_Open
()

FIN
E
```

Ora inizieremo mettendo il titolo del nostro programma all'inizio del file finestra quando il programma viene eseguito. Per fare ciò, imposteremo la proprietà `Caption` del form sul testo che vogliamo usare come titolo. Immettere questa riga nella subroutine `Form_Open()`:

```
ME.Caption = "*** ThirdProject ***"
```

Questo si prende cura del testo per il titolo della finestra. Ora, se guardiamo la nostra immagine del risultato finale, dobbiamo ancora aggiungere i nostri controlli. Inizieremo dall'alto, aggiungendo i controlli che già conosciamo: `Label`, `TextBox` e `Button`. Posizionare ogni controllo nel modulo come mostrato nella Figura 26 sopra. Assegnare un nome ai controlli `Label1`, `TextBox1` e `UpdateBtn`. Questi controlli sono stati usati nell'ultimo capitolo, quindi dovresti sentirti a tuo agio nell'usarli nei tuoi programmi. Successivamente, aggiungeremo il pulsante `Esci` nell'angolo in basso a destra del modulo. Assegna un nome a questo controllo `QuitBtn`. Ora aggiungiamo il codice per questi controlli. Fare doppio clic con il mouse su `QuitBtn` e aggiungere questo codice alla subroutine `QuitBtn_Click()`:

```
PUBLIC SUB QuitBtn_Click
```

Una guida per principianti a

```
( ) Form1.Close  
FINE
```

Una guida per principianti a

Per il pulsante di aggiornamento, trasferiremo il testo digitato dall'utente nel controllo TextBox in TextLabel, riflettendo qualsiasi formattazione HTML immessa. Ecco il codice necessario per questo:

```
PUBLIC SUB UpdateBtn_Click ()  
    TextLabel1.Text = TextBox1.Text  
FINE
```

Da questo punto in poi, impareremo a utilizzare nuovi controlli e funzionalità di Gambas. Abbiamo già scritto il codice per assegnare i dati di input a TextLabel, quindi prendiamoci il tempo per imparare tutto ciò che possiamo al riguardo. Prima di iniziare, è probabilmente una buona idea andare avanti e fare clic sul pulsante Salva sull'IDE per salvare il lavoro.

TextLabel

Questa classe è creabile ed eredita i suoi attributi dalla classe Control. Implementa un controllo in grado di visualizzare un semplice testo HTML. Puoi utilizzare entità HTML come < e > per visualizzare caratteri come <e>. Per l'HTML che richiede l'uso delle virgolette, evita la "" perché non funzionerà. Ad esempio, inserendo:

```
<div align = "center"> <b> Non funzionerà. </b>
```

La figura seguente mostra un esempio di come funziona:

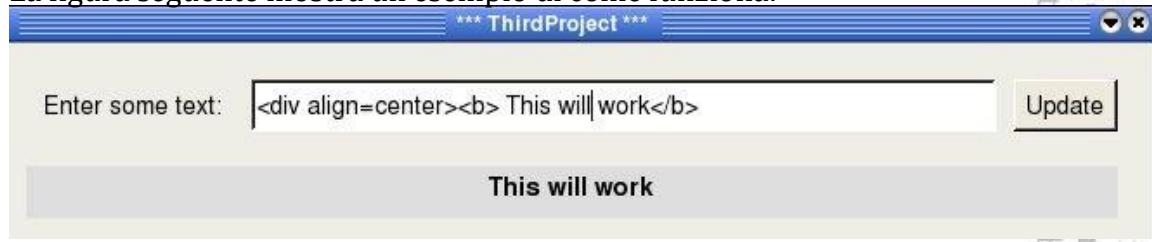


Figura 27 Utilizzo dell'HTML per formattare l'output di TextLabel.

TextLabel è definito come DIM hTextLabel AS TextLabel e lo dichiari in questo modo:

```
hTextLabel = NEW TextLabel (Parent AS Container)
```

Quello che vogliamo fare ora è codificare il pulsante di aggiornamento per sfruttare queste fantastiche funzionalità HTML. A tale scopo, è necessario modificare il codice immesso in precedenza per l'evento clic del pulsante di aggiornamento.

Una guida per principianti a
Fare doppio clic sul file

Una guida per principianti a

UpdateBtn controlla e modifica il codice in questo modo:

```
PUBLIC SUB UpdateBtn_Click ()  
    TextLabel1.Text = "<div align = center> <b>" & TextBox1.Text &  
    "</b>" END
```

Invece di assegnare semplicemente il testo di input dal TextBox alla proprietà TextLabel1.Text, lo faremo sembrare stravagante. Ora, tutto ciò che l'utente digita verrà automaticamente centrato in grassetto e posizionato su TextLabel1. Se salvi il progetto e lo esegui in questo momento, i risultati dovrebbero essere come quelli della Figura 28 di seguito:

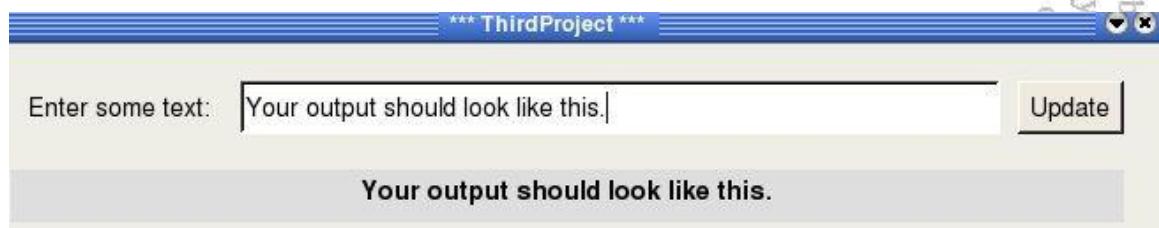


Figura 28 Output di TextLabel modificato utilizzando la formattazione HTML.

Casella di testo

Il nostro prossimo compito è saperne di più sul controllo Textbox che abbiamo appena usato. Sai già che viene utilizzato per consentire all'utente di immettere una riga di input di testo e restituire tale input al programma come stringa. TextBox eredita i suoi attributi dalla classe Control e questa classe implementa un controllo di modifica del testo a riga singola. Questa classe è creabile ed è dichiarata come:

```
DIM hTextBox AS TextBox
```

Per creare e istanziare un nuovo controllo TextBox nel codice, puoi utilizzare questo formato:

```
hTextBox = NEW TextBox (Parent AS Container)
```

Proviamo qualcos'altro adesso. Supponiamo di voler semplificare la nostra applicazione in modo che l'utente non debba fare clic sul pulsante di aggiornamento quando inserisce il testo. Inoltre, sarebbe bello che il campo di testo si pulisse da solo se il mouse è entrato nell'area di controllo e reinserisse il prompt predefinito quando il mouse viene allontanato dal controllo. Vorremmo preservare tutto ciò che è stato digitato nel campo dall'utente quando il mouse lascia il controllo, quindi abbiamo

Una guida per principianti a

per ricordarsi di assegnare il valore che si trova attualmente nel campo TextBox1.Text al campo TextLabel1.Text. Vogliamo continuare a utilizzare le capacità di formattazione HTML di TextLabel.

Dovremo modificare il nostro codice. Innanzitutto, dobbiamo creare i gestori di eventi per il mouse che entra nel controllo e ne esce. Fare clic una volta sul controllo TextBox1 nel modulo e fare clic con il pulsante destro del mouse per selezionare l'elemento Evento. Dal sottomenu Evento scegliere Invio. Ripeti questo processo e scegli l'evento Lascia. Ora vai all'editor di codice, trova la prima subroutine (TextBox1_Enter ()) e inserisci il seguente codice tra la prima e l'ultima riga:

```
TextBox1.Clear
```

Questo è tutto per questa routine. Chiamiamo semplicemente il metodo incorporato per cancellare il TextBox e abbiamo finito. Ora, dobbiamo ancora andare alla subroutine TextBox1_Leave () e inserire questo codice:

```
TextLabel1.Text = "<div align = center> <b>" & TextBox1.Text & "</b>"  
TextBox1.Text = "<Inserisci testo qui>"
```

La prima riga di codice utilizzerà HTML per formattare e centrare la stringa TextBox1.Text e impostarla su un carattere in grassetto. La riga di codice successiva assegna la stringa di prompt predefinita che abbiamo inizialmente creato al TextBox1.Text, quindi quando il mouse esce abbiamo salvato tutto ciò che l'utente ha digitato e ripristinato come appariva prima che il mouse entrasse nel controllo. Salva il tuo lavoro ed esegui il programma per vedere come funziona.

Non ci sono realmente metodi TextBoxunique da discutere, ma mentre impariamo a conoscere gli eventi, giochiamo un po 'con il codice e impariamo i metodi Mostra e Nascondi e come possono essere usati in un programma. Fare clic una volta sul modulo (fare attenzione a non fare clic su alcun controllo nel modulo) e fare clic con il pulsante destro del mouse per selezionare l'elemento Evento. Dal sottomenu Evento scegliere MouseDown. Ripeti questo processo e scegli l'evento DblClick. Ora vai all'editor di codice, trova la subroutine Form_MouseDown () e inserisci il seguente codice tra la prima e l'ultima riga:

```
UpdateBtn.Hide
```

Questo è tutto ciò che dobbiamo fare per nascondere il controllo UpdateBtn. Se l'utente sceglie di non utilizzare il pulsante ora, è sufficiente fare clic sul modulo e questo nasconderà il pulsante. Come lo riportiamo indietro? Doubleclick farà bene il

Una guida per principianti a
trucco. Dal

Una guida per principianti a
editor di codice, trova la subroutine Form_DblClick () e inserisci questo codice:

```
UpdateBtn.Show
```

In questo momento, potresti chiederti "Come lo saprebbe l'utente?" Buon punto. Faglielo sapere con una proprietà ToolTip. Fare clic sul controllo UpdateBtn. Ora vai alla finestra delle proprietà e trova ToolTip. Sul lato destro di quella voce c'è un campo di input e con una casella grigia con tre punti (...). Ciò significa che si aprirà un'altra finestra di dialogo quando si fa clic. In questo caso, si aprirà una finestra di modifica in modo da poter formattare la descrizione comando utilizzando HTML. Fare clic e quando viene visualizzata la finestra di dialogo di modifica, immettere questo testo (o codice, come si desidera):

Fare clic su (o fare doppio clic) nel modulo per nascondermi (o mostrarmi) .

Ora salva il tuo lavoro ed esegui il programma per vedere come funziona. Passa il mouse sul pulsante Aggiorna e apparirà la descrizione comando ben formattata. Dovrebbe assomigliare alla figura sotto. Splendido, no?

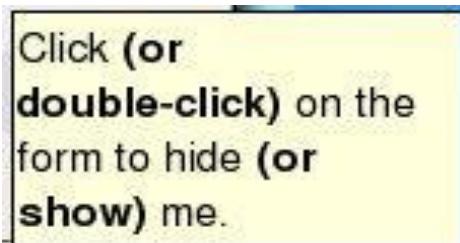


Figura 29 Aggiunta di una descrizione comando per informare l'utente su come mostrare / nascondere un controllo.

Ci sono un paio di eventi TextBoxunique di cui potremmo discutere a questo punto, vale a dire KeyPress e KeyRelease, ma li terremo a bada fino a più avanti nel libro, quando inizieremo a lavorare direttamente con la tastiera. Pronto per andare avanti? In seguito aggiungeremo alcuni controlli di selezione al nostro programma. Il ComboBox e il ListBox consentono agli utenti di scegliere tra diverse scelte. Inizieremo prima con il ComboBox.

Combo box

Il controllo ComboBox eredita i suoi attributi dalla classe Control. Implementa una casella di testo combinata con una casella di riepilogo popup.

Una guida per principianti a
Questa classe è creabile e viene dichiarata utilizzando il formato seguente:

Una guida per principianti a

```
DIM hComboBox AS ComboBox
```

Per istanziare la variabile che abbiamo appena dichiarato, usa questo formato:

```
hComboBox = NEW ComboBox (Parent AS Container)
```

Ciò creerà un nuovo controllo ComboBox. Questa classe si comporta come un array di sola lettura. In altre parole, dichiarare la variabile, istanziarla e recuperare un valore direttamente dal codice funzionerebbe in questo modo:

```
DIM hComboBox AS ComboBox
DIM hComboBoxItem AS .ComboBoxItem
Indice DIM come numero intero

hComboBoxItem = hComboBox [Indice]
```

La riga di codice sopra restituirà un elemento della casella combinata dal suo indice intero. Notare che il .ComboBoxItem rappresenta un elemento nella casella di riepilogo popup della casella combinata. Questa classe è virtuale. Non è possibile utilizzarlo come tipo di dati e non è creabile. Ha una singola proprietà, Text, che restituisce un valore stringa che rappresenta l'elemento ComboBox rappresentato da Index. Creeremo ora un ComboBox nel nostro modulo. Ecco come apparirà il nostro controllo:



Figura 30 Il nostro ComboBox.

OK, sono d'accordo che non è niente di molto stravagante e probabilmente si colloca un po 'in basso nella lista della creatività, ma porterà a termine il lavoro e ti insegnereà ciò che devi sapere per usare i ComboBox. Per costruire il nostro nuovo controllo, andremo al Gambas ToolBox e selezioneremo il controllo ComboBox. Posizionalo sul modulo (fai riferimento alla nostra figura iniziale all'inizio di questo capitolo per vedere dove dovrebbe andare) e cerca di avvicinarlo il più possibile a ciò che vedi in quell'immagine. Una volta che hai finito, fai clic su di esso una volta se non vedi le maniglie e vai

This production is covered by the Creative Commons License (OCL) and may not be distributed or reproduced without express written consent of the author.
Gambas User Community under a Creative Commons License (OCL) and may not be distributed or reproduced without express written consent of the author.

Una guida per principianti a
alla finestra Proprietà. Trova la proprietà List. La casella di input a destra di
questo abiliterà un file

Una guida per principianti a

finestra di dialogo del selettore se fai clic al suo interno. Fare clic sul pulsante di dialogo del selettore (ricorda, con i tre punti?) E verrà visualizzata la finestra di dialogo Modifica proprietà elenco:

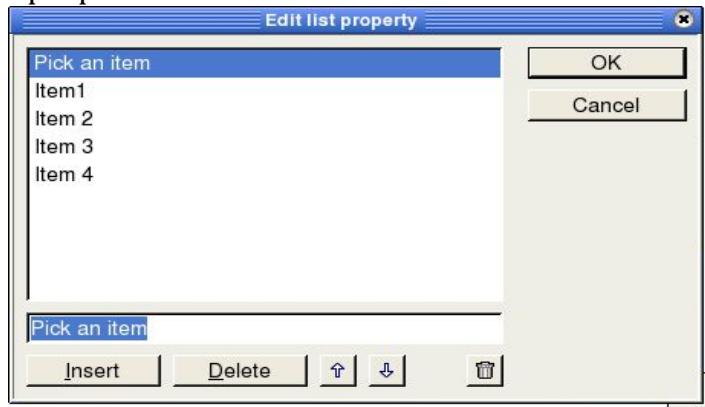


Figura 31 Editor delle proprietà dell'elenco di modifica.

Per inserire un elemento nell'elenco, digita l'elemento nella casella di testo nella parte inferiore dell'editor. Il primo elemento nell'elenco è il valore predefinito. Viene spesso utilizzato per impostare una selezione predefinita (come Nessuno o Scegli qualcosa, ecc.) Nel nostro caso, digita Scegli un elemento e fai clic su Inserisci. Aggiungere Item1, Item2, Item3 e Item 4 all'elenco. Assicurati che appaiano nell'ordine mostrato nella Figura 31 sopra in modo che appaia corretto quando viene eseguito. Una volta impostato come desideri, fai clic sul pulsante OK e l'editor si chiuderà e salverà il tuo lavoro. Salva il tuo progetto ed esegui il programma per vedere come appare sul tuo modulo. Quando hai finito di guardare il lavoro che abbiamo fatto finora, esci dal programma e continueremo aggiungendo del codice per elaborare le selezioni di ComboBox.

Se l'utente seleziona qualcosa dal ComboBox, l'elemento cambierà e questa modifica genera un evento che puoi gestire. Dal modulo, seleziona il controllo ComboBox e fai clic con il pulsante destro del mouse. Scegli Evento e seleziona Modifica dal sottomenu. Ora vai all'editor del codice e trova la subroutine PUBLIC SUB ComboBox1_Change (). Aggiungi il codice in modo che la subroutine abbia questo aspetto:

```
PUBLIC SUB ComboBox1_Change ()  
    Risultato DIM AS String  
  
    risultato = Trim (ComboBox1.Text)  
    TextLabel1.Text = risultato & "è stato selezionato dal  
    ComboBox" END
```

This product is (C) 2005 by John W. Ritter. All rights reserved. It may not be reproduced in whole or in part without the express written consent of the author.

Una guida per principianti a

Salva il tuo progetto ed esegui il programma per vedere cosa succede. Notare che quando si seleziona un elemento il valore `TextLabel1.Text` viene aggiornato ma non ha

Una guida per principianti a

qualsiasi formattazione di fantasia di cui è capace. Bene, questo è facile da risolvere. Cambiamo il codice in questo modo:

```
PUBLIC SUB ComboBox1_Change ()  
    Risultato DIM AS String  
    result = "<div align = center> <b>" & Trim (ComboBox1.Text) & "</b>"  
    TextLabel1.Text = result & " è stato selezionato."  
FINE
```

Modificare il codice ed eseguire nuovamente il programma. Dovresti vedere qualcosa di simile a questo nel controllo TextLabel1:



Figura 32 Formattazione di un'etichetta di testo con HTML.

Quindi i veri programmati là fuori stanno già chiedendo come farlo con il codice, eh? Pensavo che non me lo avresti mai chiesto! Seriamente, non è difficile. Modifichiamo leggermente il nostro programma per consentire l'aggiunta o la rimozione dinamicamente di elementi dal ComboBox. Per fare ciò, avremo bisogno di un pulsante che indicherà l'aggiunta di un elemento e un altro pulsante per rimuovere un elemento. Aggiungi due pulsanti, uno chiamato PlusBtn e l'altro chiamato MinusBtn al modulo, come mostrato di seguito:



Figura 33 Pulsanti più e meno per modificare l'elenco ComboBox.

Fai doppio clic su PlusBtn e inserisci questo codice:

```
PUBLIC SUB PlusBtn_Click  
    () DIM Item AS String  
    DIM NumItems AS Integer  
  
    ComboBox1.Refresh  
    NumItems = ComboBox1.Count  
    IF NumItems = 0 THEN  
        Item = "Scegli un  
oggetto" ALTRIMENTI  
        Item = "Item" & Str (NumItems)  
    ENDIF  
    ComboBox1.Refresh  
    ComboBox1.Add (Item,  
    NumItems)
```

Una guida per principianti a

FINE

Una guida per principianti a

Nel codice sopra, dichiariamo due variabili, Item e NumItems. Item è una stringa e NumItems un Integer. Useremo NumItems per determinare quanti elementi sono già nell'elenco. Questa operazione viene eseguita utilizzando la proprietà .Count. Se non ci sono elementi nell'elenco, imposteremo il nostro prompt predefinito di "Scegli un articolo" come Articolo zero.

Gli array ComboBox sono a base zero, il che significa che il conteggio inizia da zero, non da uno. Se il valore NumItems è zero, creeremo una stringa predefinita. Altrimenti, concateneremo il numero dell'elemento alla parola Item in modo che venga aggiunto allo slot successivo aperto nell'elenco. Chiamiamo il metodo Refresh per forzare il ComboBox ad aggiornare la sua proprietà count, quindi se l'utente prova a selezionarlo di nuovo sarà quello corrente. L'ultima riga di codice utilizza il metodo .Add per aggiungere l'elemento all'elenco di elementi ComboBox. Ora, fai doppio clic su MinusBtn e aggiungi questo codice:

```
PUBLIC SUB MinusBtn_Click
()
DIM NumItems AS Integer

'L'array della casella combinata
inizia da zero ComboBox1.Refresh
NumItems = ComboBox1.Count
IF NumItems > 0 THEN
    DEC NumItems
    IF NumItems <> 0 THEN
        ComboBox1.Remove
        (NumItems)
    FINISCI SE
    ComboBox1.Refresh
ENDIF
FINE
```

La rimozione di elementi dall'elenco è molto più semplice. Abbiamo solo bisogno di una variabile intera per determinare il numero di elementi nell'elenco. Ancora una volta, ciò viene eseguito utilizzando la proprietà .Count. Nel nostro esempio, Count restituirà un valore di 5 la prima volta che viene letto. Se non ci sono elementi nell'elenco, non faremo nulla. Poiché gli elementi ComboBox sono archiviati in una matrice a base zero, è necessario diminuire il conteggio di uno se non è già zero. Ora salva il tuo progetto ed esegui il programma. Prova a rimuovere tutti gli elementi nell'elenco e a reinserirli. Fantastico, eh? Altre cose interessanti arriveranno con ListBox, di cui parleremo in seguito.

ListBox

Una guida per principianti a

Il controllo ListBox eredita anche i suoi attributi dalla classe Control. Il ListBox implementa un elenco di elementi di testo selezionabili. Questa classe è creabile. Il codice seguente dichiarerà un ListBox e lo creerà:

This product is (C) 2005 W. Rittinghouse, all rights reserved. This is released to the
Gambas User Community under the terms of the GNU General Public License.
This software can be redistributed and/or modified without the express
written permission of the author.

Una guida per principianti a

```
DIM hListBox AS ListBox  
hListBox = NEW ListBox (Parent AS Container)
```

Come il ComboBox, questa classe si comporta come un array di sola lettura.

```
Indice DIM come valore  
intero DIM hListBox AS  
ListBox  
DIM hListBoxItem AS .ListBoxItem  
hListBoxItem = hListBox [Indice]
```

La riga di codice precedente restituirà un elemento ListBox dal suo indice intero. Notare che .ListBoxItem rappresenta un elemento nella casella di riepilogo popup ListBox. Questa classe è virtuale. Non è possibile utilizzarla come tipo di dati e non è creabile. Ha una singola proprietà, Text, che restituisce un valore stringa che rappresenta l'elemento ListBox rappresentato da Index. Creeremo ora un ListBox nel nostro modulo. Ecco come apparirà il nostro nuovo controllo:



Figura 34 Come sarà il nostro ListBox.

Per costruire il nostro nuovo controllo, andremo al Gambas ToolBox e selezioneremo il controllo ListBox. Mettilo sul modulo (fai riferimento alla nostra immagine iniziale all'inizio di questo capitolo per vedere dove dovrebbe andare) e cerca di avvicinarlo il più possibile a ciò che vedi in quell'immagine. Una volta che hai finito, fai clic su di esso una volta se non vedi le maniglie e vai alla finestra Proprietà. Trova la proprietà List. La casella di immissione a destra di questa abiliterà una finestra di dialogo di selezione se si fa clic al suo interno. Fare clic sul pulsante di dialogo del selettore (ricorda, con i tre punti?) E verrà visualizzata la finestra di dialogo Modifica proprietà elenco:

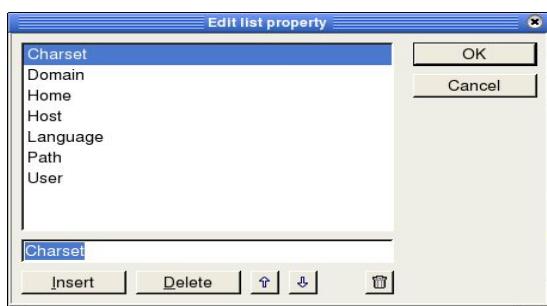


Figura 35 ListBox Modifica dell'editor delle proprietà dell'elenco.

Una guida per principianti a

Questo è esattamente lo stesso editor che abbiamo usato per creare il controllo ComboBox1. Questa volta aggiungeremo elementi all'elenco e impareremo anche la classe System. La classe System fa parte della libreria dei componenti Gambas e fornisce il supporto per tutte le classi incluse nell'interprete per impostazione predefinita. La classe System è statica e fornisce informazioni di sola lettura sull'ambiente del sistema operativo. Le proprietà di questa classe che puoi leggere includono set di caratteri, dominio, home, host, lingua, percorso e utente. Utilizza l'editor e inserisci i seguenti elementi nell'elenco:

- Charset
- Domain
- Home
- Host
- Language
- Path
- Utente

Successivamente, fare doppio clic sul controllo ListBox1 nel modulo. Questo ti porterà all'editor del codice e dovresti essere nella subroutine PUBLIC SUB ListBox1_Click (). Il codice seguente ti mostrerà come accedere a questi valori. Inseriscilo come segue:

```
PUBLIC SUB ListBox1_Click ()  
  
IF Trim (ListBox1.Text) = "System.Charset" THEN  
    TextLabel1.Text = System.Charset  
ELSE IF Trim (ListBox1.Text) = "Domain" THEN  
    TextLabel1.Text = System.Domain  
ELSE IF Trim (ListBox1.Text) = "Home"  
    THEN TextLabel1.Text = System.Home  
ELSE IF Trim (ListBox1.Text) = "Host"  
    THEN TextLabel1.Text = System.Host  
ELSE IF Trim (ListBox1.Text) = "Lingua" THEN  
    TextLabel1.Text = System.Language  
ELSE IF Trim (ListBox1.Text) = "Path"  
    THEN TextLabel1.Text = System.Path  
ELSE IF Trim (ListBox1.Text) = "Utente"  
    THEN TextLabel1.Text = System.User  
ENDIF  
END
```

Ora salva il tuo progetto ed esegui il programma. Prova a selezionare tutti gli elementi nell'elenco e guarda quali informazioni vengono restituite. Liste e ComboBox sono molto

Una guida per principianti a

facile da implementare e utilizzare. Un grande vantaggio dell'utilizzo di questi tipi di componenti del selettori è che elimina virtualmente la possibilità che un utente digit i dati in modo errato. Nella prossima sezione, impareremo come organizzare i controlli sul nostro modulo per gestire meglio la presentazione ai controlli all'utente. Inoltre, impareremo perché i frame e i pannelli sono una buona cosa da considerare all'inizio del processo di sviluppo.

Telaio

La classe Frame eredita i suoi attributi dalla classe Container. Questo controllo è un contenitore con un bordo inciso e un'etichetta. Questa classe è creabile. Dichiara e istanziare un Frame, utilizzare questo formato:

```
DIM hFrame AS Frame  
hFrame = NEW Frame (Parent AS Container)
```

La cosa grandiosa di una cornice è che funziona come una finestra all'interno di una finestra. Qualsiasi controllo che metti nella cornice diventa una parte della cornice, per così dire. Ciò significa che se si inseriscono controlli o pulsanti CheckBox o qualsiasi altra cosa nel frame e si decide di spostare il frame, tutti si muovono con esso. Possono essere riorganizzati all'interno della cornice, ma se la cornice si muove, si nasconde, ecc., lo fanno anche loro. Ecco come saranno il nostro frame e i controlli che aggiungeremo ad esso:

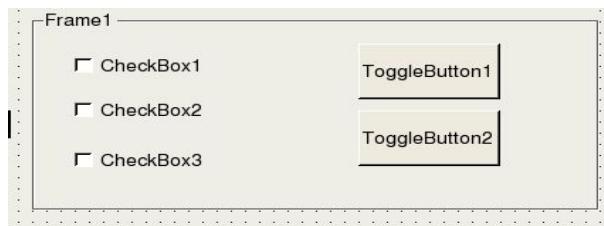


Figura 36 Come sarà il frame di esempio che costruiremo.

Per continuare lo sviluppo del nostro progetto, aggiungeremo prima un controllo Frame al form (chiamiamolo Frame1) e lo posizioneremo in modo simile a quello che si trova nella Figura 24 all'inizio di questo capitolo. Successivamente, aggiungeremo due controlli ToggleButton (denominati ToggleButton1 e ToggleButton2) e tre controlli CheckBox (denominati CheckBox1, CheckBox2 e CheckBox3). Disporli in modo che assomiglino alla Figura 33 sopra. Una volta che hai posizionato tutto correttamente nel modulo, andiamo al passaggio successivo e scriviamo il codice per gli eventi a cui risponderemo nel nostro programma.

Una guida per principianti a

Interruttore

Il controllo ToggleButton eredita i suoi attributi dalla classe Control e implementa un pulsante di attivazione / disattivazione. Ciò significa che si attiva o si abbassa. Questa classe è creabile. Per dichiarare e creare un'istanza di questo controllo, utilizzare il formato:

```
DIM hToggleButton AS ToggleButton  
hToggleButton = NUOVO ToggleButton (Parent AS Container)
```

Fare doppio clic sul controllo ToggleButton1 e aggiungere questo codice al file **PUBLIC SUB ToggleButton1_Click ()** sottoprogramma:

```
PUBLIC SUB ToggleButton1_Click ()  
    IF ToggleButton1.Value = TRUE THEN  
        Frame1.Text = "Abbassato di 1"  
    ALTRO  
        Frame1.Text = "Attivato 1  
    su"  
    ENDIF  
FINE
```

Ripeti il passaggio precedente per il controllo ToggleButton2 e inserisci questo codice:

```
PUBLIC SUB ToggleButton2_Click  
() DIM risultato AS String  
    IF ToggleButton2.Value = TRUE THEN  
        Frame1.Text = "Abbassato 2  
    volte"  
    ALTRO  
        Frame1.Text = "2 in su"  
    ENDIF  
FINE
```

Quello che abbiamo fatto con il codice sopra è controllare la proprietà ToggleButton.Value per vedere se è TRUE, indicando che il pulsante è stato cliccato. Se FALSE, un altro clic lo ha portato in posizione sollevata. Indipendentemente dalla posizione in cui si trova, vogliamo che Frame1.Text visualizzi lo stato dell'ultimo pulsante su cui si è fatto clic. Ora, passiamo a conoscere i controlli CheckBox.

Casella di controllo

La classe CheckBox eredita i suoi attributi dalla classe Control. Questa classe implementa un controllo casella di controllo ed è creabile. Dichiara e crea un'istanza della variabile in questo modo:

Una guida per principianti a

DIM hCheckBox AS CheckBox

Una guida per principianti a

```
hCheckBox = NEW CheckBox (Parent AS Container)
```

Quando si fa clic su un controllo CheckBox nel nostro programma, vogliamo rilevare l'evento clic e rispondere immediatamente. In questo caso, cambieremo la proprietà Checkbox.Text ogni volta che CheckBox.Value viene controllato e risulta essere TRUE. Questo mostrerà che abbiamo rilevato l'evento clic e abbiamo risposto al valore TRUE o FALSE (selezionato o deselectato). Se la casella è deselectata, dovremo riportarla allo stato "Normale". Per CheckBox1, fare doppio clic sul controllo e immettere questo codice nell'editor di codice per la subroutine PUBLIC SUB CheckBox1_Click ():

```
PUBLIC SUB CheckBox1_Click
() DIM outline1 AS String
DIM outline2 AS String
DIM outline3 AS String

IF CheckBox1.Value = TRUE THEN
    Checkbox1.Text = "Sono stato scelto"
ALTRO
    Checkbox1.Text = "Checkbox1"
ENDIF
FINE
```

Ripeti questo processo per CheckBox2 e CheckBox3:

```
PUBLIC SUB CheckBox2_Click ()
IF CheckBox2.Value = TRUE THEN
    Checkbox2.Text = "Sono stato scelto"
ALTRO
    Checkbox2.Text = "Checkbox2"
ENDIF
FINE

PUBLIC SUB CheckBox3_Click ()
IF CheckBox3.Value = TRUE THEN
    Checkbox3.Text = "Sono stato
scelto" ALTRIMENTI
    Checkbox3.Text = "Checkbox3"
ENDIF
FINE
```

Ora salva il tuo progetto ed esegui il programma. Una volta che sei accertato che funzionino come previsto, termina il programma e noi continueremo il nostro progetto aggiungendo gli ultimi due controlli che tratteremo in questo capitolo, il Pannello e il RadioButton.

This document is released to the public domain by the author, John W. Rittinghouse, 2005 by John W. Rittinghouse, all rights are reserved. It is released to the public domain under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

This product is (C) 2002 by John W. Rittinghouse, all rights reserved. It is based on the Gaius User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Pannello

La classe Panel eredita i suoi attributi dalla classe Container. Questa classe implementa un controllo Panel con un bordo modificabile. Questa classe è creabile. Dichiara e crea un'istanza di un pannello come questo:

```
Pannello DIM hPanel AS  
hPanel = NEW Panel (Parent AS Container)
```

Nel nostro programma useremo il Pannello per raggruppare i nostri RadioButton al suo interno. Ecco come apparirà questo pannello quando avremo finito:



Figura 37 Un pannello con pulsanti radio.

Dovremo prima aggiungere un controllo Panel al form (chiamarlo Panel1) e posizionarlo in modo che assomigli al Panel che si trova all'inizio di questo capitolo. Successivamente, aggiungeremo due controlli RadioButton (denominati RadioButton1 e RadioButton2). Disporli in modo che assomiglino alla Figura 37 sopra.

RadioButton

La classe RadioButton eredita i suoi attributi dalla classe Control e viene utilizzata per implementare un controllo pulsante di opzione. I controlli RadioButton che condividono lo stesso genitore (in questo caso, il contenitore del pannello) si escludono a vicenda. È possibile selezionare un solo RadioButton alla volta. Questa classe è creabile. Dichiara e crea un'istanza di un RadioButton in questo modo:

```
DIM hRadioButton AS RadioButton  
hRadioButton = NUOVO RadioButton (Parent AS Container)
```

Selezionare il controllo RadioButton dalla casella degli strumenti e inserirlo nel form. Assegna un nome al primo RadioButton RadioButton1 e ripeti questo processo per un altro, denominato RadioButton2. Dopo aver posizionato tutto

Una guida per principianti a
correttamente nel modulo, andiamo al passaggio successivo e scriviamo il codice.
Fai doppio clic sul primo RadioButton e inserisci questo codice:

This product is (C) 2005 by [V. Rittinghouse](#), all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB RadioButton1_Click ()  
    RadioButton1.Text = "HaHa RB2"  
    RadioButton2.Text = "RadioButton2"  
FINE
```

Ripeti questo processo per il secondo:

```
PUBLIC SUB RadioButton2_Click ()  
    RadioButton2.Text = "HaHa RB1"  
    RadioButton1.Text = "RadioButton1"  
FINE
```

Ora salva il tuo progetto ed esegui il programma. Una volta che ti sarai accortato che tutto ciò che abbiamo fatto in questo capitolo funzionerà come previsto, prenditi una meritata pausa e inizieremo da capo con il prossimo capitolo.

Capitolo 6 - Menu, moduli, finestre di dialogo e finestre di messaggio

Quasi tutti i programmi basati su GUI che incontrerai utilizzeranno una combinazione di menu, MessageBox e finestre di dialogo standard per comunicare con l'utente. Gambas non è diverso. Inoltre, non esiste alcun linguaggio di programmazione che fornisca tutto ciò che un programmatore utilizzerebbe come componente predefinito. Questo è ciò per cui vengono utilizzati i moduli: scrivere ciò di cui hai bisogno al di fuori dell'ambito di ciò che Gambas fornisce. Ad esempio, se hai bisogno di una funzione che restituisca il nome di un colore in base a ciò che viene scelto in una finestra di dialogo di selezione del colore standard, dovrà scriverne uno, e lo faremo proprio nel nostro prossimo progetto di esempio. In questo capitolo impareremo come usare ciascuno di questi controlli e costruiremo un altro programma di esempio per aiutarti a imparare a padroneggiare Gambas. Ecco uno screenshot della versione finale di come sarà il nostro progetto:

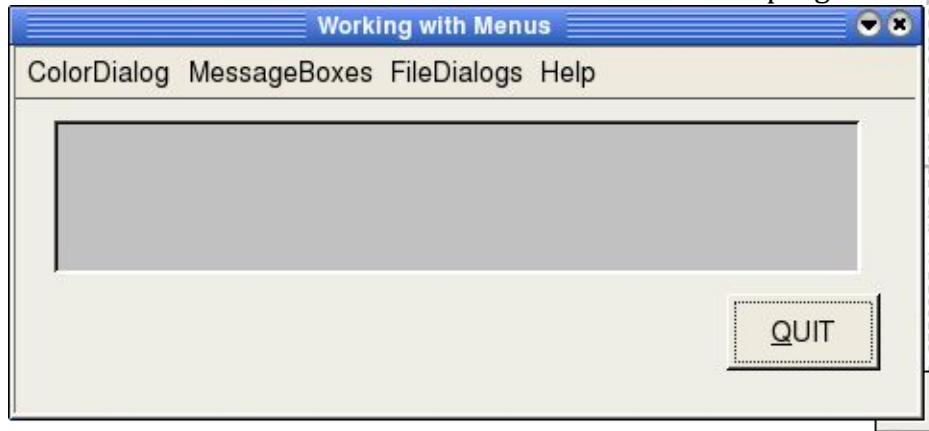


Figura 38 Risultati finali del progetto di menu.

Creeremo quattro menu. Ciascun menu verrà utilizzato per dimostrare i controlli o le finestre di dialogo standard che apprenderemo in questo capitolo. Per prima cosa costruiremo il menu, come mostrato sopra, quindi ti mostreremo come utilizzare i colori standard e le finestre di dialogo dei file, MessageBox e, come bonus aggiuntivo, ti mostreremo come creare un modulo in Gambas e usarlo. Per iniziare il nostro progetto, apri Gambas e crea un nuovo progetto di interfaccia utente grafica. Chiamalo MenuProject e imposta tutti i controlli traducibili e pubblici. Una volta arrivato all'IDE, crea un form, Form1 e rendilo una classe di avvio. Aggiungi un pulsante Esci al modulo, come mostrato sopra e inserisci questo codice da eseguire quando un utente fa clic sul pulsante Esci:

This product is (C) 2001 by John W. Rittichier, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB QuitBtn_Click  
    () Form1.Close  
FINE
```

Fare doppio clic sul modulo ovunque al di fuori di un limite di controllo e si otterrà la subroutine PUBLIC SUB Form_Open () da visualizzare nell'editor del codice. Inserisci questo codice in quella subroutine vuota:

```
ME.Caption = "Lavorare con i menu"
```

Successivamente, aggiungi un TLabel, denominato TLabel1 alla finestra, come mostrato nella Figura 38 sopra. Questo si occupa dei preliminari per il nostro progetto. Vediamo ora come costruire il menu e utilizzare le finestre di dialogo standard fornite da Gambas.

L'editor del menu Gambas

Gambas ha un editor di menu incorporato. È possibile accedervi dalla finestra del modulo premendo CTRLE o facendo clic con il pulsante destro del mouse sul modulo e selezionando l'opzione Editor di menu dal popup che appare. In ogni caso, apparirà l'Editor dei menu e vedrai qualcosa del genere:

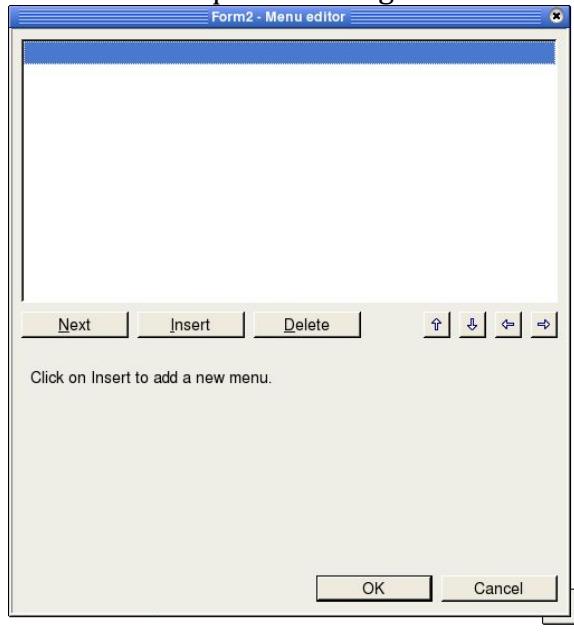


Figura 39 Editor del menu Gambas quando viene avviato per la prima volta.

Una guida per principianti a

Quando avvii per la prima volta l'Editor di menu, la finestra dell'editor sarà, ovviamente, vuota. Non è uno strumento intuitivamente ovvio da usare, quindi dedichiamo un minuto a spiegare l'Editor di menu. Innanzitutto, una terminologia comune. Il testo visualizzato orizzontalmente nella parte superiore di un modulo viene definito menu. Un menu può contenere MenuItems (elencati verticalmente sotto il menu). MenuItems può contenere subMenus, che sono fondamentalmente nuovi menu nidificati che appartengono a un MenuItem e appariranno e visualizzeranno un altro elenco di MenuItems. I menu possono contenere diversi livelli di sottomenu nidificati.

Per inserire un menu dovrà utilizzare il pulsante Inserisci. Il pulsante Avanti sposterà la posizione del cursore del menu corrente all'elemento successivo nella finestra. Si sposta solo verso il basso e non è presente alcun pulsante precedente. Se raggiungi la fine dell'elenco, si fermerà e l'unica cosa che puoi fare è fare clic con il mouse su una voce di menu o inserirne una nuova. Il pulsante Elimina rimuoverà la voce di menu sotto il cursore del menu.

I primi due pulsanti freccia ↑ e ↓ (su e giù) a destra del pulsante Elimina sposteranno la voce sotto il cursore del menu in alto o in basso nell'elenco delle voci di menu. Se fai clic sui pulsanti freccia ← o → (annulla rientro e rientro), funzionerà in modo molto simile al rientro in una struttura. L'elemento rientrato diventerà un MenuItem o un subMenuItem a seconda del livello di rientro. Gli elementi rientrati con tre ... prima del nome indicano MenuItems. Più di tre punti che precedono il nome indicano che si tratta di un subMenuItem della voce al livello di rientro precedente. Una volta creato uno o due menu, sarà facile da capire. Dopo aver fatto clic sul pulsante di inserimento, vedrai apparire diversi campi nella parte inferiore dell'Editor dei menu, ovvero questi:

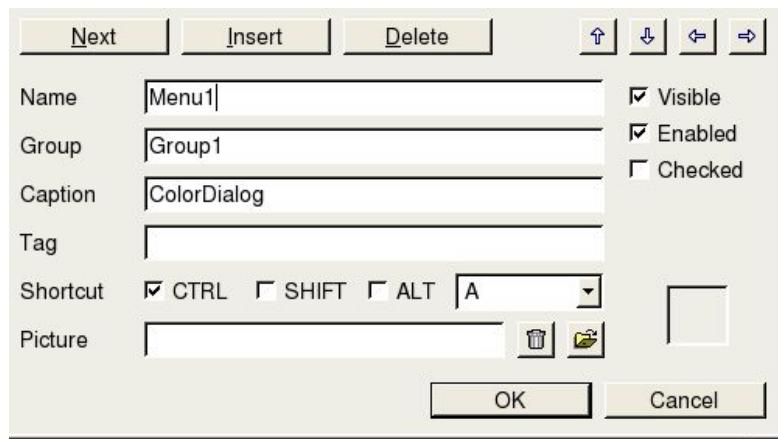


Figura 40 Modifica campi per l'Editor dei menu.

Ritchie, S. (2001). *Windows Context Menus*. Retrieved from <http://www.angelfire.com/richie/contextmenu.html> and may not be distributed without the express written consent of the author.

Una guida per principianti a

Il campo Nome avrebbe probabilmente dovuto essere chiamato Nome variabile perché è per questo che viene utilizzato questo campo. Questo è il nome della variabile che il codice nel file di classe utilizzerà per fare riferimento a questo particolare oggetto.

Il campo Gruppo fa riferimento a un gruppo di controllo. I gruppi di controllo vengono utilizzati nell'IDE Gambas per creare e gestire gruppi di controlli come un'unica entità. Quando selezioni un controllo in un modulo, nella finestra di dialogo Proprietà, dovresti notare una proprietà Gruppo che non corrisponde a nessuna proprietà documentata di quel controllo. Questa speciale pseudoproprietà consente di assegnare gruppi di controllo. In sostanza, quando quel controllo viene creato in fase di esecuzione, verrà trattato come se lo avessi creato nel codice in questo modo:

```
myControl = NEW ColumnView (ME) AS "myGroup"
```

e i suoi gestori di eventi devono essere denominati in questo modo:

```
PUBLIC SUB myGroup_Click ()
```

Puoi fare riferimento al controllo che ha generato l'evento con la parola chiave LAST, oppure puoi assegnare un tag a ciascun controllo per differenziarli utilizzando il valore del tag.

Didascalia è il testo che l'utente finale vedrà quando il programma viene eseguito. Il tag è riservato ai programmati per memorizzare i dati ed è una proprietà comune per quasi tutti i controlli. Tratteremo in seguito come utilizzare la proprietà Tag. Per ora, devi solo sapere che può contenere **QUALSIASI** tipo di dati Variant.

Le caselle di controllo scorciatoia consentono di definire combinazioni di tasti di scelta rapida da tastiera come ControlA per richiamare automaticamente l'evento clic per quella voce di menu. Ricorda, le voci di menu in pratica rispondono solo a eventi di clic o scorciatoie da tastiera. Il campo Immagine consente di specificare il nome file di un'icona che verrà visualizzata con la voce di menu. A destra di questo campo viene visualizzata una finestra di anteprima (ovvero una finestra di immagine).

I CheckBox a destra dei campi di input, Visible, Enabled e Checked si riferiscono allo stato della voce di menu. Per impostazione predefinita, tutte le voci di menu vengono create per essere abilitate e visibili. È possibile specificare se l'elemento è selezionato o meno. Nel codice, puoi vedere se la voce è selezionata o meno guardando la proprietà Controllata della voce di menu. Restituirà un valore

Una guida per principianti a VERO o FALSO che indica lo stato dell'articolo. Ecco un esempio di come attivare o disattivare un segno di spunta quando viene selezionata una voce di menu:

This product is released under the GNU General Public License (GPL) Version 2.0. W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without express written consent of the author.

Una guida per principianti a

```
IF Menu2Item1.Checked THEN  
    Menu2Item1.Checked = FALSE  
ALTRO  
    Menu2Item1.Checked = TRUE  
ENDIF
```

L'istruzione IF verificherà la proprietà Checked dell'oggetto Menu2Item1 e se il valore booleano è TRUE lo imposterà su FALSE (alternandolo). Se il valore booleano è NOT TRUE, viene eseguita la clausola ELSE, impostando il valore .Checked su TRUE. Questa logica può essere utilizzata per alternare qualsiasi menu o subMenuItem.

NOTA: una parola di cautela: utilizzare questa funzione con parsimonia per una buona progettazione della GUI. Molti utenti si sentono sopraffatti quando vengono presentati con troppi gadget in un'interfaccia.

Menu di costruzione

Ora creiamo il nostro menu. Quando hai finito, apparirà qualcosa del genere dall'editor del menu:

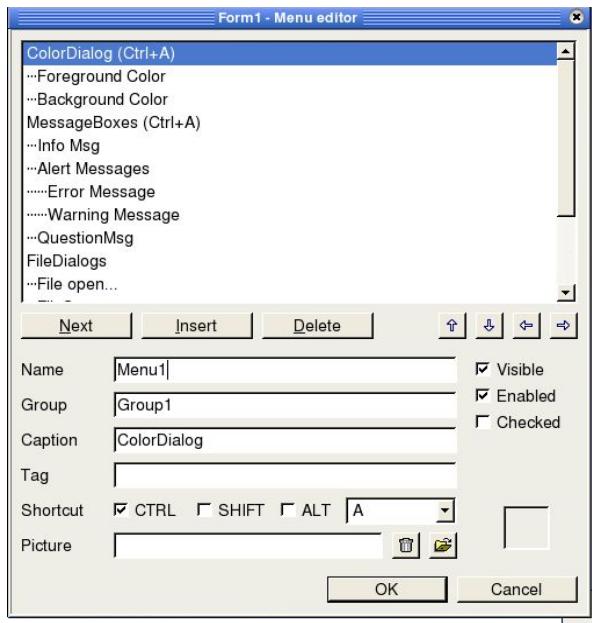


Figura 41 Creazione del menu del nostro progetto.

Per iniziare, fai clic sul pulsante Inserisci. Innanzitutto, inseriremo i menu di primo livello, ColorDialog, MessageBoxes, FileDialogs e Aiutare rendendo facile, usa il

Una guida per principianti a
file

Una guida per principianti a
tabella sottostante per compilare i dati del campo di input:

| Menu principale | Nome (variabile) | Gruppo | Didascalia |
|------------------------|-------------------------|---------------|-------------------|
| ColorDialog | Menu1 | | ColorDialog |
| MessageBoxes | Menu2 | | MessageBoxes |
| FileDialogs | Menu3 | | FileDialogs |
| Aiuto | Menu4 | | Aiuto |

Ora, mettiamo le voci di menu nel primo menu. Queste voci di menu apparterranno al menu ColorDialog:

| ColorDialog | Nome (variabile) | Gruppo | Didascalia |
|-----------------------|-------------------------|---------------|-----------------------|
| Colore di primo piano | MenuItem1 | | Colore di primo piano |
| Colore di sfondo | MenuItem2 | | Colore di sfondo |
| Font | MenuItem3 | | Font |

Dopo aver inserito questi elementi con l'editor, assicurati che siano posizionati sotto la voce ColorDialog e rientrati di un livello. Ora, sposta il cursore del menu sulla seconda voce del menu principale, MessageBoxes e inseriremo le voci di menu per questo menu.

| MessageBoxes | (Variabile) Nome | Gruppo | Didascalia |
|---------------------------|--------------------------|---------------|---------------------------|
| Messaggio informativo | Menu2Item1 | | Messaggio informativo |
| Messaggi di avviso | Menu2Item2 | | Messaggi di avviso |
| Messaggio di errore | subMenuItem1 | | Messaggio di errore |
| Messaggio di avvertimento | subMenuItem2 | | Messaggio di avvertimento |
| Cancella il messaggio | subMenuItem3 | | Cancella il messaggio |
| Messaggio di domanda | Menu2Item3 | | Messaggio di domanda |

Verificare che le voci Errore, Avviso ed Elimina siano rientrate DUE livelli sotto la voce di menu per i messaggi di avviso. Ora, è necessario aggiungere l'ultima voce di menu per il menu Guida.

Una guida per principianti a

| Aiuto | Nome (variabile) | Gruppo | Didascalia |
|-------|------------------|--------|------------|
| Di | Menu4 | | Di |

Una guida per principianti a

È tutto quello che c'è da fare! Fare clic sul pulsante OK per uscire dall'editor del menu e quando torni al modulo, verrà visualizzato un menu. Se non assomiglia al nostro esempio, torna all'editor del menu e regola le voci finché non lo fa. Dopo essere completamente soddisfatto del formato e dell'aspetto del menu, il passaggio successivo consiste nel codificare le azioni per ciascun evento di clic del menu. Tuttavia, prima di farlo, dobbiamo conoscere le finestre di dialogo standard e i MessageBox forniti in Gambas.

Dialoghi

La classe Dialog viene utilizzata per implementare tutte le finestre di dialogo standard di Gambas. Questa classe contiene metodi statici utilizzati per chiamare le finestre di dialogo standard. Questa classe è statica. I metodi di dialogo supportati (cioè le finestre di dialogo standard) sono: OpenFile, SaveFile, SelectColor, SelectDirectory e SelectFont. Le proprietà supportate da questa classe sono: Color, Filter, Font, Path e Title.

Il colore è un valore intero restituito dalla finestra di dialogo SelezioneColore che rappresenta il colore selezionato. È interessante notare che Gambas ha 32 costanti di colore predefinite ma la finestra di dialogo SelectColor standard è composta da 48 colori. Ciò significa che non è possibile utilizzare le costanti predefinite come Red, Black, ecc., Con alcun senso di garanzia in una dichiarazione CASE o altrove. Quando si sceglie un colore, come si determina se si tratta di una costante predefinita o di un altro colore al di fuori della gamma dei 32 predefiniti in Gambas? Scrivi un modulo per determinarlo, ovviamente! Torneremo su questo argomento dopo aver spiegato come utilizzare la finestra di dialogo SelectColor.

Quando si lavora con le finestre di dialogo relative al file, vengono restituite le proprietà Filter e Path. Filter è definito come STATIC PROPERTY Filter AS String [] e restituisce o imposta i filtri (cioè, categorizza in base all'estensione del file, come tutti i file .doc o .png) utilizzati nelle finestre di dialogo dei file standard. Questa proprietà restituisce o riceve una matrice di stringhe. Ogni elemento nella matrice rappresenta un filtro da utilizzare quando viene effettuata la chiamata alla finestra di dialogo del file. Ogni stringa di filtro deve seguire la seguente sintassi:

- ✓ Il nome del filtro.
- ✓ Uno spazio.
- ✓ Una parentesi di apertura.
- ✓ Uno o più modelli di file, separati da punto e virgola.
- ✓ Una parentesi di chiusura.

Di seguito è riportato un esempio che dimostra anche l'uso della proprietà Titolo

Una guida per principianti a
finestra di dialogo: Here is an example that also shows the use of the Dialog Title
property:

Una guida per principianti a

```
Dialog.Title = "Scegli un file"
Dialog.Filter = ["Immagini (*.png; *.Jpg; *.Jpeg)", "Tutti i file (*.*)"]
Dialog.OpenFile
```

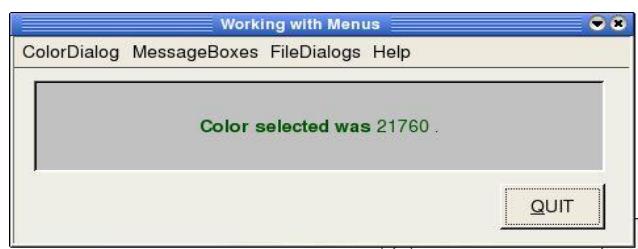
Le chiamate alla finestra di dialogo relative al file (SelectDirectory, OpenFileDialog e SaveFileDialog) restituiscono tutte la proprietà Path, che è una stringa che contiene il percorso del file selezionato. Nel caso della chiamata alla finestra di dialogo SelectDirectory, la stringa viene troncata a livello di directory. Ora, iniziamo a codificare la nostra prima voce di menu, Colore di primo piano. Dalla finestra del modulo, scegli la voce Colore di primo piano dal menu che abbiamo appena creato e verrai portato alla finestra del codice in una subroutine chiamata PUBLIC SUB Menu1Item1_Click () dove vorrai inserire questo codice:

```
PUBLIC SUB Menu1Item1_Click ()
    Dialog.Title = "Scegli un colore di primo
    piano" Dialog.SelectColor
    TextLabel1.ForeColor = Dialog.Color
    TextLabel1.Text = "<b> Il colore selezionato era </b>" &
    ColorName & "."
    FINE
```

Per prima cosa, impostiamo il titolo della finestra di dialogo su "Scegli un colore di primo piano" in modo che l'utente abbia un'idea di cosa viene utilizzata la finestra di dialogo. Chiamiamo la finestra di dialogo standard con Dialog.SelectColor. Una volta che l'utente ha scelto un colore o cancellato dalla finestra di dialogo, inseriremo il nome del colore restituito nel controllo TextLabel1.Text che abbiamo creato all'inizio del nostro progetto. Scegliamo TextLabel perché ci permette di formattare il nostro output in questo modo:

```
TextLabel1.Text = "<b> Il colore selezionato era </b>" & Str $
    (Dialog.Color) & "."
```

La riga di codice sopra verrà impostata in grassetto la parte "Colore selezionato era" della stringa, concatenare il valore Colore alla stringa dopo aver convertito il numero intero in una stringa utilizzando la funzione di conversione Str \$, quindi concatenare la punteggiatura finale alla stringa (vogliamo che abbia un aspetto professionale, non è vero?). Ora esegui il codice e dovresti vedere qualcosa del genere:



Una guida per principianti a

Figura 42 Un'etichetta di testo formattata che mostra il valore del colore.

Una guida per principianti a

Bene, 21760 non è un'informazione molto utile. Questo numero è la rappresentazione interna di quel colore verastro che ho selezionato. Dobbiamo sviluppare un metodo per sapere se il colore è rosso, blu, verde, cioè una costante di colore con nome. Questo è ciò a cui stavamo alludendo nella sezione precedente sul problema con i colori. Ecco dove entrano in gioco i moduli.

Moduli

Creeremo un modulo per restituire il nome di un colore se è una costante predefinita. In caso contrario, vogliamo una stringa che ci dica che non è una costante di colore predefinita. Una volta definito il nostro modulo, torneremo e modificheremo il nostro codice per utilizzare il modulo. Dalla finestra del progetto, trova i moduli nella visualizzazione ad albero e seleziona un nuovo modulo, denominato Module1. Quando viene visualizzata la finestra di dialogo, lasciare vuote le caselle di controllo e prendere il nome predefinito e fare clic su OK. Ti verrà presentata una nuova finestra di codice intitolata Module1.module. Dovrebbe iniziare con un commento come questo:

```
'File del modulo Gambas
```

Ora scriviamo tutto il codice per la nostra funzione, che chiameremo SetColorName. Innanzitutto, dobbiamo dichiarare la funzione e identificare quali parametri di input e output elaborerà. Ecco come lo facciamo:

```
FUNZIONE PUBBLICA SetColorName (iVal AS Integer) AS String
```

Dichiariamo una funzione pubblica che accetta un parametro intero e restituisce una stringa. Questo modulo restituirà il valore intero restituito dalla finestra di dialogo SelectColor e lo confronterà con le 32 costanti di colore predefinite utilizzate in Gambas. Se corrisponde a una costante di colore, imposteremo una stringa temporanea al nome di quel colore. In caso contrario, creeremo una stringa che identifica il valore costante e lo restituiremo al codice chiamante. Dovremo creare una variabile stringa temporanea per farlo, quindi aggiungi questa dichiarazione di variabile come prima riga di codice all'interno della funzione:

```
rval AS String
```

Ora dobbiamo aggiungere la nostra istruzione CASE SELECT. Utilizzerà il parametro intero iVal passato alla funzione dalla subroutine chiamante. Ecco l'elenco completo da inserire:

Una guida per principianti a

```
'File del modulo Gambas
FUNZIONE PUBBLICA SetColorName (iVal AS Integer) AS String
```

Una guida per principianti a

```
rval AS String
SELEZIONA iVal
CASE Color.Black
    rval = "Nero"
CASE Color.Blue
    rval = "Blue"
CASE Color.Cyan
    rval = "Cyan"
CASE Color.DarkBlue
    rval = "DarkBlue"
CASE Color.DarkCyan
    rval = "DarkCyan"
CASE Color.DarkGray
    rval = "DarkGray"
CASE Color.DarkGreen
    rval = "DarkGreen"
CASE Color.DarkMagenta
    rval = "DarkMagenta"
CASE Color.DarkRed
    rval = "DarkRed"
CASE Color.DarkYellow
    rval = "DarkYellow"
CASE Color.Gray
    rval =
    "Grigio"
CASE Color.Green
    rval = "Green"
CASE Color.LightGray
    rval = "LightGray"
CASE Color.Magenta
    rval = "Magenta"
CASE Color.Orange
    rval = "Orange"
CASE Color.Pink
    rval = "Pink"
CASE Color.Red
    rval = "Red"
CASE Color.Transparent
    rval = "Trasparente"
CASE Color.Violet
    rval = "Viola"
CASE Color.White
    rval = "White"
CASE Color.Yellow
    rval = "Yellow"
PREDEFINITO
    rval = Str $ (ival) & "e non è una costante di colore
predefinita" END SELECT
RETURN rval
END
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Dopo aver completato questo codice, salvare questo file di modulo e tornare alla finestra del codice per Form1. Dobbiamo modificare il nostro codice nella subroutine MenuItem1_Click. Ecco la nuova subroutine:

```
PUBLIC SUB MenuItem1_Click ()  
  
    Dialog.Title = "Scegli un colore di primo  
    piano" Dialog.SelectColor  
    ColorName = Module1.SetColorName (Dialog.Color)  
    TextLabel1.ForeColor = Dialog.Color  
    TextLabel1.Text = "<b> Il colore selezionato era </b>" &  
    ColorName & ".." FINE
```

Abbiamo aggiunto la riga ColorName = ... ma non abbiamo ancora dichiarato ColorName. Questa variabile deve essere globale perché stiamo chiamando un modulo per restituire il valore ColorName. All'inizio del file di codice Form1.class, aggiungi questo codice appena prima di PUBLIC SUB Form_Open ():

```
"File di classe Gambas  
'dichiara una variabile globale da utilizzare con il nostro modulo  
SetColorName ColorName AS String
```

Ora, quando il programma viene eseguito, la variabile sarà nota. Nel codice sottostante, la chiamata a:

```
ColorName = Module1SetColorName (Dialog.Color)
```

e restituirà la stringa alla nostra variabile ColorName. Vogliamo impostare il colore del testo in primo piano sul colore appena scelto:

```
TextLabel1.ForeColor = Dialog.Color
```

Ora, prenderemo la stringa e la visualizzeremo (formattata, ovviamente) assegnandola alla proprietà TextLabel1.Text.

```
TextLabel1.Text = "<b> Il colore selezionato era </b>" & ColorName &  
    ".."
```

La versione finale della nostra routine MenuItem1_Click dovrebbe essere simile a questa:

```
PUBLIC SUB MenuItem1_Click ()  
    Dialog.Title = "Scegli un colore di primo  
    piano" Dialog.SelectColor
```

Una guida per principianti a

```
ColorName = Module1.SetColorName (Dialog.Color)
TextLabel1.ForeColor = Dialog.Color
```

This product is (C) 2005 by John W. Eaton, all rights reserved. It is released to the
Gambas User Community under the Gnu General Public License (GPL) and may not be distributed
under any other terms or conditions.

Una guida per principianti a

```
TextLabel1.Text = "<b> Il colore selezionato era </b>" &  
ColorName & "." FINE
```

Vorremo fare la stessa cosa per il colore di sfondo. La seconda voce di menu nel menu ColorDialog è Colore di sfondo, quindi facciamo clic su quella dall'IDE e codifica quanto segue:

```
PUBLIC SUB MenuItem2_Click ()  
    Dialog.Title = "Scegli un colore di  
    sfondo" Dialog.SelectColor  
    ColorName = Module1.SetColorName (Dialog.Color)  
    TextLabel1.BackColor = Dialog.Color  
    TextLabel1.Text = "<b> Il colore selezionato era </b>" &  
    ColorName & "." FINE
```

L'ultima opzione nel nostro primo menu è cambiare il carattere. La finestra di dialogo SelectFont viene utilizzata per eseguire questa operazione. Ancora una volta, vai al menu del modulo e scegli l'opzione Carattere nel menu ColorDialog del nostro progetto per ottenere l'evento di clic impostato nella finestra del codice. I caratteri vengono gestiti in Gambas con la classe Font. Questa classe rappresenta un tipo di carattere utilizzato per disegnare o visualizzare il testo nei controlli. Questa classe è creabile. Per dichiarare una variabile di carattere, utilizza questo formato:

```
DIM hFont AS Font  
hFont = NEW Font ([Font AS String])
```

Il codice precedente crea un nuovo oggetto carattere da una descrizione del carattere. Questa classe si comporta come un array di sola lettura. Questo codice crea un nuovo oggetto carattere da una descrizione del carattere e lo restituisce:

```
DIM hFont AS Font  
hFont = Font [Font AS String]
```

Le proprietà che puoi usare con Font includono:

- ✓ Salita
- ✓ Grassetto
- ✓ Discesa
- ✓ Fisso
- ✓ Corsivo
- ✓ Nome
- ✓ Risoluzione
- ✓ Dimensione
- ✓ StrikeOut

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Gambas User Community License (OCL) and may not be redistributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

- ✓ Stili
- ✓ Sottolineato

Di seguito è riportato un esempio di come utilizzare il codice per impostare le proprietà del carattere:

```
Form1.Font.Name = "Utopia"  
Form1.Font.Bold = VERO  
Form1.Font.Italic = VERO  
Form1.Font.Size = "14"  
Form1.Font.StrikeOut = FALSO  
Form1.Font.Underline = VERO
```

La classe Font ha tre metodi che puoi chiamare: Height, ToString e Width. L'altezza è una funzione che restituisce l'altezza del testo visualizzato con il carattere. È dichiarato come:

FUNZIONE Altezza (testo come stringa) come numero intero

Accordare restituisce il nome completo di un carattere come stringa di descrizione. Questa stringa è una concatenazione di tutte le proprietà del carattere separate da virgolette. È dichiarato come:

FUNZIONE ToString () AS String

Un esempio di chiamata a ToString potrebbe essere:

PRINT Application.Font.ToString ()

Larghezza è una funzione proprio come Altezza ma restituisce la larghezza del testo visualizzato con il carattere:

FUNZIONE Larghezza (testo come stringa) come numero intero

Ora, continuiamo a inserire il nostro codice per il nostro programma MenuProject nella routine PUBLIC SUB Menu1Item3_Click (). Innanzitutto, dichiareremo due variabili locali, fontdata e oldfontdata. Vogliamo ottenere il nuovo font dalla chiamata alla finestra di dialogo SelectFont, ma vogliamo anche conservare i vecchi dati del font nel caso ne avessimo bisogno. Successivamente, dichiariamo due variabili stringa, attr e sel (attributo e selezione) da utilizzare per impostare la stringa dell'attributo del carattere; la selezione è una stringa di lavoro che costruiremo dinamicamente, in base alle varie selezioni che l'utente può effettuare nella finestra di dialogo.

Una guida per principianti a

```
PUBLIC SUB MenuItem3_Click  
() DIM fontdata AS font
```

Una guida per principianti a

```
DIM oldfontdata AS Font  
DIM attr AS String  
DIM sel AS String
```

A questo punto, potresti chiederti se non puoi semplicemente usare la proprietà ToString per farlo. Sì, ma non impareresti tanto. Abbi pazienza e codifica. La seguente riga di codice cancella semplicemente la nostra stringa di attr:

```
attr = ""
```

Ora, assegniamo la proprietà TextLabel1.font corrente alla variabile oldfontdata:

```
oldfontdata = TextLabel1.Font
```

Quindi, impostiamo il titolo per la finestra di dialogo e chiamiamo la finestra di dialogo SelectFont con queste due linee:

```
Dialog.Title = "Scegli un  
carattere ..." Dialog.SelectFont
```

Quando l'utente seleziona un font dalla finestra di dialogo SelectFont, vogliamo assegnare il font selezionato alla variabile fontdata:

```
fontdata = Dialog.Font
```

A questo punto, controlleremo per vedere quali attributi l'utente ha scelto e costruiremo dinamicamente una stringa di output. Avremmo potuto costruire questa istruzione IF THEN ELSE per controllare ogni proprietà supportata dalla classe Font, ma non siamo interessati a tutte per questo esercizio.

```
IF fontdata.Italic THEN  
    sel = "Corsivo"  
    attr = attr & sel  
ENDIF  
IF fontdata.Bold THEN  
    sel = "Bold"  
    attr = attr & sel  
ENDIF  
IF fontdata.Strikeout THEN  
    sel = "Barrato"  
    attr = attr & sel  
ENDIF  
IF fontdata.Underline THEN  
    sel = "Sottolineato"  
    attr = attr & sel  
ENDIF
```

Una guida per principianti a

Successivamente, imposteremo il carattere nel controllo TextLabel1 in modo che sia ciò che l'utente ha selezionato e visualizzeremo la stringa di dati del carattere creata dinamicamente all'utente:

```
TextLabel1.Font = fontdata
TextLabel1.Text = "Font:" & fontdata.Name & "," & Str (Round (fontdata.Size)) &
attr
```

Aspetteremo cinque secondi per dare un'occhiata al risultato, quindi daremo loro la possibilità di scegliere se vogliono o meno mantenere queste impostazioni o ripristinare le impostazioni precedenti.

ATTENDERE 5.0

```
SELEZIONA Message.Question ("Keep the new font?", "Yes", "No", "Don't
know") CASO 1
    TextLabel1.Text = "Questo è ora il carattere
predefinito." CASO 2
    TextLabel1.Font = oldfontdata
    TextLabel1.Text = "Ripristinato l'impostazione del
carattere precedente." CASO 3
    TextLabel1.Font = oldfontdata
    TextLabel1.Text = "Non è stata apportata alcuna modifica
al carattere predefinito." FINE SELEZIONA
FINE
```

Si noti che abbiamo saltato un po 'la pistola e abbiamo introdotto l'uso dei MessageBox senza spiegarli. Questo è il nostro prossimo compito, ma volevo darti un'anteprima. Salva il tuo progetto ed eseguiolo in questo momento. I risultati dovrebbero essere simili a questo, a seconda dei colori e del carattere selezionati:



Figura 43 Selezione di colori e caratteri.

E se decidi di mantenerlo come predefinito, vedrai qualcosa del genere:

This product is licensed under the terms of the GNU General Public License version 2, or at your option, any later version. It is released to the public domain under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

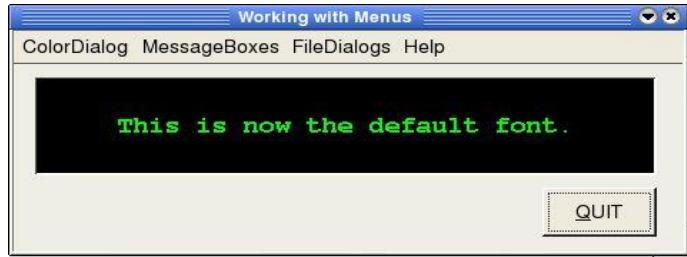


Figura 44 Creazione di un nuovo carattere predefinito per il controllo `TextLabel1`.

A questo punto del tuo viaggio di apprendimento, sai come fare un bel po' a Gambas. Dobbiamo continuare questo viaggio espandendo la nostra capacità di gestire input e output dell'utente. I MessageBox sono un altro prezioso strumento nel nostro arsenale di Gambas. Scopriamo di più su come utilizzarli.

MessageBoxes

I controlli MessageBox sono un mezzo molto pratico per comunicare informazioni all'utente o ottenere risposte di tipo VERO / FALSO, SÌ / NO. In generale, i MessageBox rientrano in una delle quattro categorie principali: query e / o conferma, informazioni, avviso e notifiche di errore / avviso. La classe Message viene utilizzata per visualizzare questi MessageBox. Questa classe è statica e può essere utilizzata come funzione. È dichiarato come

FUNZIONE STATICÀ Message (Message AS String [, Button AS String]) AS Integer

I metodi supportati dalla classe Message sono Info, Question, Delete, Error e Warning. Hai già visto il metodo Message.Question utilizzato già utilizzato. Ogni metodo ha uno scopo specifico e deve essere utilizzato in modo appropriato nel codice. Cominciamo con il metodo MessageBox più semplice, il metodo Information.

Messaggi di informazione

Il metodo Info è dichiarato come:

FUNZIONE STATICÀ Info (Messaggio AS String [, Button AS String]) AS Integer

e viene utilizzato per visualizzare un MessageBox di informazioni, con un solo pulsante. Poiché l'utente non deve prendere una decisione, deve solo riconoscere il messaggio. Quindi, è necessario un solo pulsante. Ecco cosa produrrà il nostro codice:

This product is (C) 2015 John W. Rittinger, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without express written consent of the author.

Una guida per principianti a

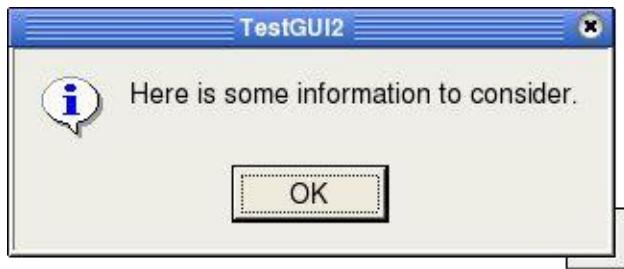


Figura 45 Un MessageBox di informazioni.

Per continuare a costruire il nostro programma di esempio, dal form vai al menu MessageBoxes e clicca sulla voce di menu Info Message. Ti porterà alla finestra dell'editor del codice e ti troverai nella subroutine Menu2Item1_Clicked(). Ecco qui:

```
PUBLIC SUB Menu2Item1_Click()
    Message.Info ("Ecco alcune informazioni da
considerare.") END
```

Abbastanza semplice, eh? E se volessimo cambiare il valore della proprietà selezionata da TRUE a FALSE o viceversa ogni volta che questa voce di menu è stata selezionata. Modificare il codice sopra come segue:

```
PUBLIC SUB Menu2Item1_Click
() IF Menu2Item1.Checked
THEN
    Menu2Item1.Checked = FALSE
ELSE
    Menu2Item1.Checked = TRUE
ENDIF
Message.Info ("Ecco alcune informazioni da
considerare.") END
```



Figura 46 Una voce
di menu selezionata.

La Figura 43 sopra mostra la voce di menu selezionata. Selezionandolo di nuovo verrà deselezionato. Beh, non è stato troppo difficile, vero? Vedi, Gambas rende le cose facili. Ora passiamo alla parte successiva.

Una guida per principianti a

Interroga / Conferma messaggi

Domanda ed Elimina rientrano in questa categoria. La domanda è definita come:

```
FUNZIONE STATICÀ Domanda (Messaggio AS String [, Button1 AS String,  
Button2 AS String, Button3 AS String]) AS Integer
```

Invocarlo visualizza una domanda MessageBox con un massimo di tre pulsanti. Viene restituito l'indice del pulsante cliccato dall'utente. Torniamo al nostro codice precedente ed esaminiamolo in dettaglio:

```
SELEZIONA Message.Question ("Keep the new font?", "Yes", "No", "Don't  
know") CASO 1  
    TextLabel1.Text = "Questo è ora il carattere  
predefinito." CASO 2  
    TextLabel1.Font = oldfontdata  
    TextLabel1.Text = "Ripristinato l'impostazione del  
carattere precedente." CASO 3  
    TextLabel1.Font = oldfontdata  
    TextLabel1.Text = "Non è stata apportata alcuna modifica  
al carattere predefinito." FINE SELEZIONA
```

Poiché viene restituito l'indice del pulsante su cui l'utente fa clic, è più semplice incorporare la chiamata a Message.Question in un'istruzione SELECT / CASE. SELECT accetta il valore di ritorno intero e, poiché sappiamo che può essere restituito solo come CASE 1, 2 o 3, non sarà necessario creare una sezione DEFAULT. Potremmo, ovviamente, ma non è necessario. A seconda del valore restituito, eseguiremo un'azione Sì, un'azione No o un'azione Non so perché è ciò che abbiamo specificato nella nostra chiamata al dialogo (nell'istruzione SELECT sopra). Scriviamo ora il nuovo codice per la nostra voce di menu MessageBoxes Menu Question. Sceglilo dalla finestra del modulo e inserisci questo codice:

```
PUBLIC SUB Menu2Item3_Click ()  
  
    SELEZIONA Message.Question ("Ti è piaciuto questo?", "Sì", "No", "Non  
so") CASO 1  
        TextLabel1.Text = "Mi è  
piaciuto." CASO 2  
        TextLabel1.Text = "Non mi è  
piaciuto." CASO 3  
        TextLabel1.Text = "Non lo  
sapevo." FINE SELEZIONA  
FINE
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Cambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Quando esegui il programma, ecco cosa dovresti vedere:

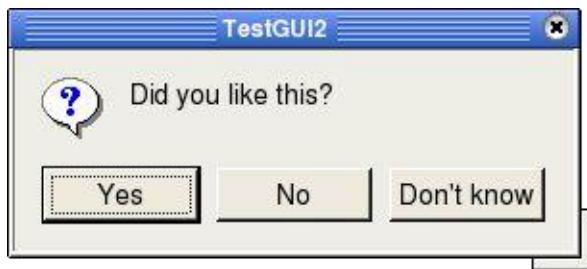


Figura 47 MessageBox di una domanda.

Messaggio di errore

La funzione Error visualizza un MessageBox di errore, con un massimo di tre pulsanti.

Viene restituito l'indice del pulsante cliccato dall'utente. L'errore è definito come:

```
FUNZIONE STATICA Errorre (messaggio AS String [, Btn1 AS String, Btn2 AS String, Btn3 AS String]) AS Integer
```

Vai al modulo e fai clic sul subMenuItem Messaggio di errore per impostare un evento clic e inserisci questo codice:

```
PUBLIC SUB subMenuItem1_Click () Message.Error  
    ("Wow! Questo è stato un errore.")  
FINE
```

Ecco cosa dovresti vedere quando esegui il programma:

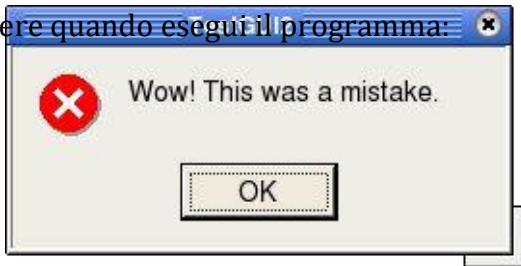


Figura 48 Un messaggio di errore.

Messaggi di avviso o di avviso

I messaggi di avviso visualizzano un MessageBox di avviso, con un massimo

Una guida per principianti a
di tre pulsanti
del pulsante cliccato dall'utente è restituito

l'indice
è

This product is released under the terms of the GIMP General Public License version 2 or later. It may be used, copied, modified, and distributed in accordance with the terms of the license, which are available at <http://www.gimp.org/licensing.html>. This release includes software developed by others, which may have their own specific licenses.

Una guida per principianti a

dichiarato come:

```
FUNZIONE STATICA Avviso (Messaggio AS String [, Btn1 AS String, Btn2 AS String, Btn3 AS String]) AS Integer
```

Nel nostro programma di esempio, i messaggi di errore, avviso ed eliminazione fanno parte del sottomenu che abbiamo creato per apparire quando la voce di menu Messaggi di avviso è attivata. Per codificare il messaggio di avviso, è necessario impostare l'evento clic andando al menu e facendo clic sul sottomenu Messaggio di avviso. Questo ci metterà nella finestra del codice dove inseriremo questo codice:

```
PUBLIC SUB subMenuItem2_Click ()
    Message.Warning ("Sei stato avvertito di questo!")
END
```

Quando esegui il programma, ecco cosa dovresti vedere:

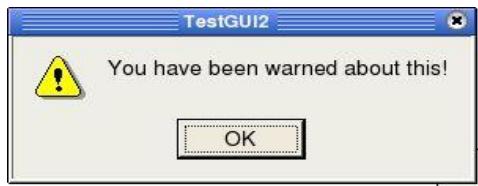


Figura 49 Un messaggio di avviso.

Elimina messaggi

La funzione Elimina visualizza un MessageBox di eliminazione, con un massimo di tre pulsanti. Viene restituito l'indice del pulsante cliccato dall'utente. Elimina è dichiarato come:

```
FUNZIONE STATICA Elimina (Messaggio AS String [, Btn1 AS String, Btn2 AS String, Btn3 AS String]) AS Integer
```

Imposta l'evento clic per subMenuItem Elimina messaggio e inserisci questo codice:

```
PUBLIC SUB subMenuItem3_Click ()
    SELEZIONA Message.Delete ("Delete?", "Yes", "No",
    "Cancel") CASO 1
        TextLabel1.Text = "Eliminato"
    CASO 2
        TextLabel1.Text = "Non
    CASO 3
        eliminato" TextLabel1.Text =
```

Una guida per principianti a

"

A

n

n

u

l

l

a

t

o

!

"

This product is (C) 2005 by John W. Rittich.
Gambas User Community under an Open
Source License (OCL) and may be distributed
under any other terms or conditions without
written consent of the author.

Una guida per principianti a

```
FINE  
SELEZIONA  
FINE
```

Quando esegui il programma, ecco cosa dovresti vedere:

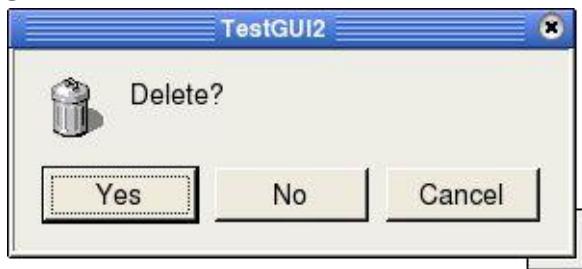


Figura 50 Elimina messaggio con tre pulsanti.

Funzioni relative ai file della classe di dialogo

Le finestre di dialogo standard fornite con Gambas supportano anche operazioni sui file come l'apertura di un file, il salvataggio di un file e la scelta di un percorso di directory. Sono abbastanza standard ma il grande vantaggio è che non devi crearli ogni volta che vuoi scrivere un programma. La coerenza nell'interfaccia utente è ora richiesta dagli utenti e qualsiasi cosa meno rende il tuo programma meno accessibile e, di conseguenza, meno accettato dagli utenti di quanto potrebbe essere. Questa sezione ti mostrerà quanto è facile usare le finestre di dialogo standard di Gambas per le operazioni sui file.

Funzione di dialogo OpenFileDialog

La funzione OpenFileDialog richiama la finestra di dialogo standard del file per ottenere il nome di un file da aprire. Questa funzione restituisce TRUE se l'utente ha fatto clic sul pulsante Annulla e FALSE se l'utente ha fatto clic sul pulsante OK. Si dichiara come segue:

```
FUNZIONE STATICA OpenFileDialog () AS Boolean
```

Dalla finestra del modulo del programma di esempio, fare clic sul menu FileDialogs e selezionare la voce di menu File apri ... Questo imposta l'evento clic e noi inseriremo questo codice:

```
PUBLIC SUB MenuItem1_Click ()  
    Dialog.Title = "Apri file ..."  
    Dialog.OpenFile
```

Una guida per principianti a

```
TextLabel1.Text = "OPEN:" & Dialog.Path  
END
```

Una guida per principianti a

Il codice sopra imposta semplicemente il titolo e chiama la finestra di dialogo standard. Quando l'utente termina, la stringa restituita dalla funzione OpenFileDialog viene assegnata alla variabile TextLabel1.Text. Ecco cosa dovrebbe apparire quando esegui il codice:

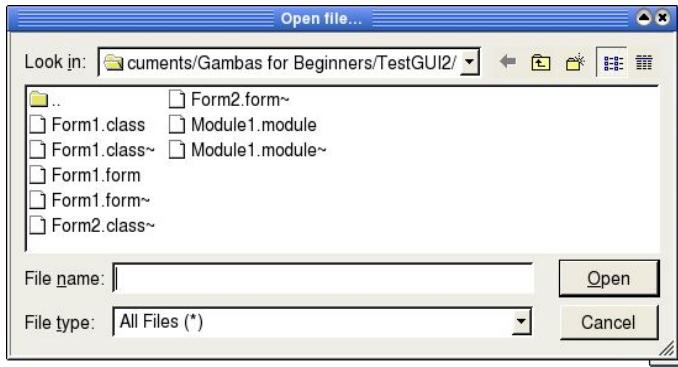


Figura 51 La finestra di dialogo Apri file.

Roba abbastanza facile. Salvare i file è altrettanto facile.

Finestra di dialogo Funzione SaveFile

La funzione SaveFileDialog chiama anche la finestra di dialogo standard del file per ottenere il nome di un file da salvare. Restituisce TRUE se l'utente ha fatto clic sul pulsante Annulla e FALSE se l'utente ha fatto clic sul pulsante OK. SaveFileDialog è dichiarato come:

```
FUNZIONE STATICA SaveFileDialog () AS Boolean
```

Crea un evento clic per la nostra voce di menu File Salva ... e nella finestra del codice inserisci questo codice:

```
PUBLIC SUB MenuItem2_Click ()  
  
    Dialog.Title = "Salva file in ..."  
    Dialog.SaveFileDialog  
    TextLabel1.Text = "SAVE:" & Dialog.Path  
  
END
```

Ecco il risultato:

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

This product is (C) 2005 by Iain Rittinghouse, all rights are reserved. It is released to the Gambas User Community under the Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

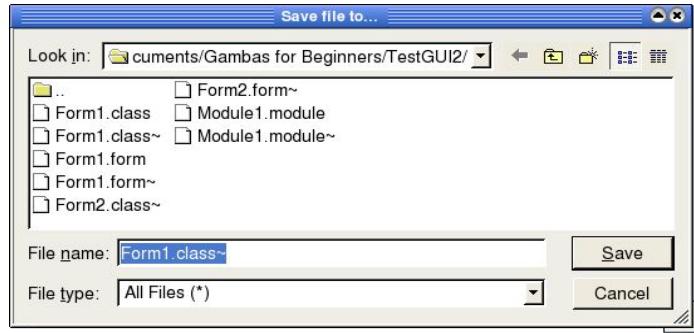


Figura 52 Finestra di dialogo Salva file.

Finestra di dialogo SelectDirectory Function

La funzione SelectDirectory Dialog richiama la finestra di dialogo standard del file per ottenere un nome di directory esistente. Restituisce TRUE se l'utente ha fatto clic sul pulsante Annulla e FALSE se l'utente ha fatto clic sul pulsante OK. SelectDirectory è dichiarato come:

```
FUNZIONE STATICA SelectDirectory () AS Boolean
```

Questo è l'evento clic finale che dobbiamo codificare per il nostro programma. Scegli la voce di menu Seleziona directory e inserisci questo codice:

```
PUBLIC SUB Menu3ItemClick ()
    Dialog.Title = "Scegli una
    directory ..."
    Dialog.SelectDirectory
    TextLabel1.Text = "SAVE to DIR:" & Dialog.Path
END
```

Salva il tuo progetto ed esegui il codice. Ecco cosa dovresti vedere per SelectDir:



Figura 53 La finestra di dialogo Seleziona directory.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed or modified in any other way without the express written consent of the author.

Una guida per principianti a

Ora abbiamo scritto tutto il codice necessario per completare il nostro programma MenuProject. Salvatela ed eseguite il programma. La sezione successiva presenta il file di classe completo da rivedere. È tempo per noi di passare a cose più grandi e migliori. Nel prossimo capitolo, inizieremo ad imparare come gestire l'input e l'output dell'utente utilizzando stringhe e file.

Elenco completo di esempio

```
"File di classe Gambas
'dichiara una variabile globale da utilizzare con il nostro modulo
SetColorName ColorName AS String

PUBLIC SUB Form_Open ()
    ME.Caption = "Lavorare con i menu"
END

PUBLIC SUB QuitBtn_Click
    () Form1.Close
FINE

PUBLIC SUB MenuItem1_Click ()
    Dialog.Title = "Scegli un colore di primo
    piano" Dialog.SelectColor
    ColorName = Module1.SetColorName (Dialog.Color)
    TextLabel1.ForeColor = Dialog.Color
    TextLabel1.Text = "<b> Il colore selezionato era </b>" &
    ColorName & "."
FINE

PUBLIC SUB MenuItem2_Click ()
    Dialog.Title = "Scegli un colore di
    sfondo" Dialog.SelectColor
    ColorName = Module1.SetColorName (Dialog.Color)
    TextLabel1.BackColor = Dialog.Color
    TextLabel1.Text = "<b> Il colore selezionato era </b>" &
    ColorName & "."
FINE

PUBLIC SUB MenuItem3_Click
    () DIM fontdata AS font
    DIM oldfontdata AS Font

    DIM attr AS String
    DIM sel AS String
    attr = ""
    oldfontdata = TextLabel1.Font

    Dialog.Title = "Scegli un
    carattere ..." Dialog.SelectFont
```

Una guida per principianti a

```
fontdata = Dialog.Font
SE fontdata.Italic
ALLORA
    sel = "Corsivo"
    attr = attr & sel
FINISCI SE
IF fontdata.Bold THEN
    sel = "Bold"
    attr = attr & sel
ENDIF
IF fontdata.Strikeout THEN
    sel = "Barrato"
    attr = attr & sel
ENDIF
IF fontdata.Underline THEN
    sel = "Sottolineato"
    attr = attr & sel
ENDIF
TextLabel1.Font = fontdata
TextLabel1.Text = "Font:" & fontdata.Name & "," & Str (Round
(fontdata.Size)) & attr
ATTENDERE 5.0

    SELEZIONA Message.Question ("Keep new font?", "Yes", "No",
"Don't CASE 1
        TextLabel1.Text = "Questo è ora il carattere
predefinito." CASO 2
        TextLabel1.Font = oldfontdata
        TextLabel1.Text = "Ripristinato l'impostazione del
carattere precedente." CASO 3
        TextLabel1.Font = oldfontdata
        TextLabel1.Text = "Non è stata apportata alcuna modifica
al carattere predefinito." FINE SELEZIONA
FINE

PUBLIC SUB Menu2Item1_Click
() IF Menu2Item1.Checked
THEN
    Menu2Item1.Checked = FALSE
ELSE
    Menu2Item1.Checked = TRUE
ENDIF
Message.Info ("Ecco alcune informazioni da
considerare.") END

PUBLIC SUB subMenuItem1_Click () Message.Error
("Wow! Questo è stato un errore.")
FINE
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the
Gambas User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

conosc

Una guida per principianti a

```
PUBLIC SUB subMenuItem2_Click ()
```

Una guida per principianti a

```
Message.Warning ("Sei stato avvertito di questo!")
END

PUBLIC SUB subMenuItem3_Click ()
SELEZIONA Message.Delete ("Delete?", "Yes", "No",
"Cancel") CASO 1
    TextLabel1.Text = "Eliminato"
CASO 2
    TextLabel1.Text = "Non
CASO 3
    eliminato"

    TextLabel1.Text = "Annullato!"
FINE
SELEZIONA
FINE

PUBLIC SUB Menu2Item3_Click ()
SELEZIONA Message.Question ("Ti è piaciuto questo?", "Sì", "No", "Non
so") CASO 1
    TextLabel1.Text = "Mi è piaciuto."
CASO 2
    TextLabel1.Text = "Non mi è
CASO 3
    piaciuto." TextLabel1.Text = "Non lo
    sapevo."
FINE
SELEZIONA
FINE

PUBLIC SUB Menu3Item1_Click ()
Dialog.Title = "Apri file ..."
Dialog.OpenFile
TextLabel1.Text = "OPEN:" & Dialog.Path
END

PUBLIC SUB Menu3Item2_Click ()
Dialog.Title = "Salva file in ..."
Dialog.SaveFile
TextLabel1.Text = "SAVE:" & Dialog.Path
END

PUBLIC SUB Menu3Item3_Click ()
Dialog.Title = "Scegli una
directory ..."
Dialog.SelectDirectory
TextLabel1.Text = "SAVE to DIR:" & Dialog.Path
END
```

This product is (C) 2005 by John W Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB Menu4Item1_Click ()  
    Message.Info ("Questo è tutto su <br> <b> Programmazione Gambas  
</b>! ") END
```

Una guida per principianti a

Modulo1. Elenco del modulo

```
'File del modulo Gambas
FUNZIONE PUBBLICA SetColorName (iVal AS Integer) AS String
    rval AS String

    SELEZIONA iVal
    CASE Color.Black
        rval = "Nero"
    CASE Color.Blue
        rval = "Blue"
    CASE Color.Cyan
        rval = "Cyan"
    CASE Color.DarkBlue
        rval = "DarkBlue"
    CASE Color.DarkCyan
        rval = "DarkCyan"
    CASE Color.DarkGray
        rval = "DarkGray"
    CASE Color.DarkGreen
        rval = "DarkGreen"
    CASE Color.DarkMagenta
        rval = "DarkMagenta"
    CASE Color.DarkRed
        rval = "DarkRed"
    CASE Color.DarkYellow
        rval = "DarkYellow"
    CASE Color.Gray
        rval =
        "Grigio"
    CASE Color.Green
        rval = "Green"
    CASE Color.LightGray
        rval = "LightGray"
    CASE Color.Magenta
        rval = "Magenta"
    CASE Color.Orange
        rval = "Orange"
    CASE Color.Pink
        rval = "Pink"
    CASE Color.Red
        rval = "Red"
    CASE Color.Transparent
        rval = "Trasparente"
    CASE Color.Violet
        rval = "Viola"
    CASE Color.White
        rval = "White"
    CASE Color.Yellow
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
rval = "Giallo"  
DEFAULT  
    rval = Str $ (ival) & "e non è una costante di colore  
predefinita" END SELECT  
RETURN rval  
END
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Capitolo 7 - Gestione di stringhe e conversione di tipi di dati

Una delle cose più importanti che un programmatore deve sapere è come gestire le stringhe e usare le funzioni incorporate fornite dal linguaggio di sviluppo con cui potrebbe lavorare. La manipolazione delle stringhe e la conversione da un tipo di dati a un altro è quasi sempre richiesta durante la programmazione. Più un programmatore è esperto nell'uso delle funzioni incorporate, più è probabile che il suo programma funzioni in modo efficiente.

Funzioni di stringa

In Gambas è disponibile un ricco set di funzioni per le stringhe. Useremo la console per questo capitolo e impareremo le seguenti funzioni:

- ✓ Len
- ✓ \$ / LCase \$ superiore e inferiore \$ / Ucase \$
- ✓ Trim \$ / RTrim \$ e LTrim \$
- ✓ Sinistra \$ / Media \$ / Destra \$
- ✓ Spazio \$
- ✓ Sostituisci \$
- ✓ Stringa \$
- ✓ Subst \$
- ✓ InStr
- ✓ RInStr
- ✓ Diviso

Avremo bisogno di creare una nuova applicazione console per svolgere il nostro lavoro in questo capitolo. Avvia Gambas e crea una nuova applicazione terminale denominata StringTests. Quando viene visualizzato l'IDE, creare una nuova classe di avvio denominata Class1. Dovrebbe apparire la finestra del codice e dovresti vedere qualcosa del genere:

```
"File di classe Gambas

STATICO PUBBLICO SUB

Principale ()

FIN
E
```

Una guida per principianti a

Ora

abb

iam

o

tutt

i i

pre

limi

nar

i

fuo

ri

ma

no

e

sia

mo

pro

nti

per

iniz

iare

Una guida per principianti a

imparando a conoscere le corde in Gambas. Potreste ricordare da un capitolo precedente che in Gambas un tipo di dato stringa è un riferimento a una stringa di lunghezza variabile. Ha una dimensione iniziale di quattro byte e quando viene creato viene inizializzato con un valore Null.

Len

Len restituisce la lunghezza di una stringa. Il valore restituito è un numero intero. Ecco come Len verrebbe utilizzato nel codice:

```
Lunghezza = Len (stringa)
```

Dalla finestra del codice, digita quanto segue:

```
STATICO PUBBLICO SUB Principale ()
    DIM sStringLength AS Integer
    DIM sTestString AS Stringa

    sTestString = "12345678901234567890"
    sStringLength = Len (sTestString)
    PRINT ">>>" & sStringLength & "è la lunghezza della nostra stringa di
test."

    sTestString = "12345"
    sStringLength = Len (sTestString)
    PRINT ">>>" & sStringLength & "è la lunghezza della nostra stringa di
test." FINE

    sTestString = "12345678901"
    sStringLength = Len (sTestString)
    PRINT ">>>" & sStringLength & "è la lunghezza della nostra stringa di
test." FINE
```

La console risponderà con questo:

```
=> 20 è la lunghezza della nostra stringa di test.
=> 5 è la lunghezza della nostra stringa di test.
=> 11 è la lunghezza della nostra stringa di test.
```

È importante sapere che la lunghezza della stringa inizia a contare alla posizione 1, non zero come in C e in alcune altre lingue. Ora che possiamo scoprire quanti caratteri ci sono in una stringa, impariamo come convertire la stringa da un caso all'altro.

Superiore \$ / Ucase \$ / Ucase e inferiore \$ / Lcase \$ / Lcase

Una guida per principianti a

Upper \$ restituisce una stringa convertita in maiuscolo. Ucase \$ e Ucase lo sono

Una guida per principianti a

synonyms per Upper \$ e possono essere usati in modo intercambiabile. \$ Inferiore restituisce una stringa convertita in minuscolo. Lcase \$ e Lcase sono sinonimi di Lower \$ e possono anche essere usati in modo intercambiabile. Notare che queste funzioni non funzionano con le stringhe UTF8. Ecco la sintassi del linguaggio Gambas standard per l'utilizzo di queste funzioni:

```
Risultato = Superiore $  
  
(stringa) Risultato =  
  
UCase $ (stringa)
```

Digita questo nella finestra del codice ed esegui il programma:

```
STATICO PUBBLICO SUB Principale ()  
    DIM sStringLength AS Integer  
    DIM sTestString AS Stringa  
  
    sTestString = "abcdefg"  
  
    PRINT "==>" & sTestString & "è la nostra stringa  
iniziale." PRINT "==>" & UCase $ (sTestString) & "ora è  
maiuscolo."  
    PRINT "==>" & LCASE $ (sTestString) & "è ora tornato in  
minuscolo." PRINT "==>" & Upper $ (sTestString) & "ora torna in  
maiuscolo."  
  
    sTestString = "123abc456def 789ghizzz"  
  
    PRINT "==>" & sTestString & "è la nostra stringa  
iniziale." PRINT "==>" & UCase (sTestString) & "ora è  
maiuscolo."  
    PRINT "==>" & LCASE (sTestString) & "è ora di nuovo in  
minuscolo." PRINT "==>" & Upper (sTestString) & "è ora di nuovo  
in maiuscolo." PRINT "==>" & Lower $ (sTestString) & "ora è  
tornato in minuscolo."  
FINE
```

La console risponde con:

```
==> abcdefg è la nostra stringa iniziale.  
==> ABCDEFG ora è maiuscolo.  
==> abcdefg è ora tornato in minuscolo.  
==> ABCDEFG è ora tornato in maiuscolo.  
==> 123abc456def 789ghizzz è la nostra stringa di partenza.  
==> 123ABC456DEF 789GHIZZZ ora è maiuscolo.
```

Una guida per principianti a
==> 123abc456def 789ghizzz è ora tornato in minuscolo.
==> 123ABC456DEF 789GHIZZZ è ora tornato in maiuscolo.
==> 123abc456def 789ghizzz è ora tornato in minuscolo.

Trim \$, LTrim \$ e RTrim \$

Taglia \$ rimuove tutti gli spazi bianchi da entrambe le estremità di una stringa. Non lo farà

Una guida per principianti a

elimina gli spazi bianchi dopo che viene incontrato il primo spazio non bianco e fondamentalmente ignora tutti gli spazi finché non viene incontrato l'ultimo carattere non bianco, dopodiché inizierà a tagliare gli spazi vuoti finali. Uno spazio bianco è qualsiasi carattere il cui codice ASCII è strettamente inferiore a 32. Trim \$ è fondamentalmente una chiamata a Ltrim \$ e Rtrim \$ simultaneamente. È usato in questo modo:

```
Risultato = Taglia $ (stringa)
```

Prova questo sulla tua console e guarda come funziona la funzione Trim \$:

```
STATICO PUBBLICO SUB Principale ()
    DIM sTestString AS String
    DIM sResult AS String
    DIM iLength AS Integer

    STAMPA "12"
    STAMPA "12345678901234567890"
    sTestString = "<abcdef
                    " iLength =
    Len (sTestString)

    PRINT sTestString & "> è la nostra stringa
iniziale." PRINT "È" & Str $ (iLength) & "caratteri
lunghi" sResult = Trim $ (sTestString)
    PRINT sResult & "> è il risultato di Trim $
call." FINE
```

La console risponde con questo:

```
1          2
12345678901234567890
<abcdef>>   è la nostra stringa di partenza.
È lungo 14 caratteri
<abcdef> è il risultato della chiamata di Trim $.
```

Left \$, Mid \$ e Right \$

Left \$ restituisce i primi caratteri Length di una stringa. Se Length non è specificato, viene restituito il primo carattere della stringa. Se Length è negativo, vengono restituiti tutta la stringa tranne gli ultimi caratteri Length. La sintassi standard del linguaggio Gambas è:

```
Risultato = Left $ (String [, Length])
```

Una guida per principianti a
Metà \$ restituisce una sottocatena contenente i caratteri Length dal primo

Una guida per principianti a

posizione. Se Lunghezza non è specificata, viene restituito tutto dalla posizione iniziale. Se la lunghezza è negativa, viene restituito tutto dalla posizione iniziale tranne gli ultimi caratteri della lunghezza. La sintassi standard del linguaggio Gambas è:

```
Risultato = Mid $ (String, Start [, Length])
```

Giusto \$ restituisce gli ultimi caratteri di una stringa. Se Length non è specificato, viene restituito l'ultimo carattere della stringa. Se Length è negativo, vengono restituiti tutta la stringa tranne i primi caratteri Length. La sintassi standard del linguaggio Gambas è:

```
Risultato = Right $ (String [, Length])
```

Prova questo sulla tua console:

```
STATICO PUBBLICO SUB Principale ()
  DIM sTestString AS String
  DIM sResult AS String
  DIM iLength AS Integer

  STAMPA "12"
  STAMPA "12345678901234567890"
  sTestString = "abcdefghijklm"
  iLength = Len (sTestString)

  PRINT sTestString & "è la nostra stringa iniziale."
  PRINT "È" & Str $ (iLength) & "caratteri lunghi"

  sResult = Left $ (sTestString, 3)
  PRINT sResult & "è il risultato di Left $ call."
  sResult = Mid $ (sTestString, 4,3)
  PRINT sResult & "è il risultato della chiamata
  Mid $. sResult = Right (sTestString, 3)
  PRINT sResult & "è il risultato di Right $ call." FINE
```

La console risponde con questo;

```
1           2
12345678901234567890
abcdefghijklm è la nostra stringa iniziale.
È lungo 9 caratteri
abc è il risultato di Left $
call. def è il risultato di Mid $
call. ghi è il risultato di Right
```

Una guida per principianti a
§ call.

The product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Spazio \$

Space \$ restituisce una stringa contenente Length spazi Standard
linguaggio Gambas è:

```
Stringa = Spazio $ (Lunghezza)
```

Prova questo sulla console:

```
STATICO PUBBLICO SUB Principale ()
  DIM sTestString AS String
  DIM sResult AS String

  STAMPA "12"
  STAMPA "12345678901234567890123456"
  sTestString = "a"
  PRINT sTestString & Space $ (24) &
"z" END
```

Sostituisci \$

Sostituisci \$ sostituisce ogni occorrenza della stringa Pattern nella stringa String con la stringa ReplaceString e restituisce il risultato. Se String è null, viene restituita una stringa null. Se Pattern è null, viene restituita la stringa String. La sintassi standard del linguaggio Gambas è la seguente:

```
Risultato = Sostituisci $ (String, Pattern, ReplaceString)
```

Prova questo programma sulla tua console:

```
STATICO PUBBLICO SUB Principale ()
  DIM sTestString AS String
  DIM sResult AS String

  sTestString = "abc123ghi"
  PRINT sTestString & "è la nostra stringa
iniziale." sResult = Sostituisci $ (sTestString,
"123", "def") PRINT sResult & "è il risultato
della chiamata di sostituzione."

  sTestString = "abcdefgh i"
  PRINT sTestString & "è la nostra stringa
iniziale." sResult = Sostituisci $ (sTestString,
"", "")
  PRINT sResult & "è il risultato della chiamata di
```

Una guida per principianti a
sostituzione." sTestString = "\ ta \ tb \ tc \ tdef"

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PRINT sTestString & "è la nostra stringa  
iniziale." sResult = Sostituisce $ (sTestString,  
"\t", "")  
PRINT sResult & "è il risultato della chiamata di  
sostituzione." FINE
```

La console risponde con:

```
abc123ghi è la nostra stringa iniziale.  
abcdefghi è il risultato della chiamata  
di sostituzione. abcdefghi è la nostra  
stringa iniziale.  
abcdefghi è il risultato della chiamata di sostituzione.  
    Abcdef è la nostra stringa iniziale.  
        abcdef è il risultato della  
chiamata di sostituzione.
```

Stringa \$

String \$ restituisce semplicemente una stringa contenente Length moltiplicato per questo Pattern.Use formato:

```
String = String $ (Lunghezza, Motivo)
```

Prova questo sulla console:

```
STATICO PUBBLICO SUB Principale ()  
DIM sTestString AS String  
DIM sResult AS String  
  
STAMPA "12"  
STAMPA "12345678901234567890123456"  
sTestString = "a"  
  
PRINT sTestString & String (24, ".") &  
"Z" END
```

La console risponde con:

```
1  
2  
12345678901234567890123456  
a .....  
z
```

Subst \$

Una guida per principianti a

Subst \$ sostituisce gli argomenti & 1, & 2, ecc. in un modello rispettivamente con la prima, la seconda e le successive ReplaceStrings e restituisce il risultato. Se Pattern è null, viene restituita una stringa null. Per gli sviluppatori C, questo non è diverso da un file

Una guida per principianti a

sprint semplificato. Questa funzione è molto utile quando è necessario concatenare stringhe che devono essere tradotte. Non utilizzare l'operatore &, poiché l'ordine di concatenazione potrebbe cambiare con la lingua. La sintassi standard del linguaggio Gambas è:

```
Risultato = Subst (Pattern, ReplaceString [, ReplaceString])
```

Prova questa piccola applicazione sulla tua console:

```
STATICO PUBBLICO SUB Principale ()
    DIM sTestString AS String
    DIM sResult AS String

    sTestString = "abcdef"
    sResult = "ghijkl"
    PRINT "La nostra stringa iniziale è:" &
    sTestString PRINT Subst $ ("La nostra stringa
    Subst è: & 1", sResult) END
```

La console risponde con:

```
La nostra stringa iniziale
è: abcdef La nostra stringa
Subst è: ghijkl
```

InStr

InStr restituisce la posizione della prima occorrenza di sottostringa in stringa. Se viene specificato Start, la ricerca inizia dalla posizione Start. Se la sottostringa non viene trovata, InStr () restituisce zero.

```
Posizione = InStr (String, Substring [, Start])
```

Il codice seguente è un po 'più complesso e dimostrerà la potenza di questa funzione. Vogliamo trovare ogni spazio (impostato nella stringa in posizioni con numero pari) e stampare la posizione in cui ricorre ogni spazio. Immettere questo codice nella finestra del codice della console:

```
'File di classe Gambas
STATIC PUBLIC SUB Main
()
    DIM sTestString AS String
    DIM sResult AS String
```

Una guida per principianti a

```
DIM iLength AS Intero  
DIM iPosition AS Intero  
DIM iNextCharPos AS Integer
```

Una guida per principianti a

```

DIM iCounter AS Integer

sTestString = "abc def ghi"
iPosition = Instr (sTestString,
""")
PRINT sTestString & "è la nostra stringa
iniziale." PRINT "Il primo spazio è nella
posizione:" & iPosition iNextCharPos = iPosition
+ 1
iPosition      =      Instr      (sTestString,      "",,
iNextCharPos)  PRINT "Lo spazio successivo è
nella posizione:" & iPosition PRINT

sTestString = "abcdefghijklmnopqrstuvwxyz" STAMPA "12345
                                         6 "
STAMPA "123456789012345678901234567890123456789012345
PRINT sTestString & "è la nostra nuova stringa
di test." STAMPA
iLength = Len (sTestString)
PRINT "La lunghezza di sTestString è:" & iLength

iPosition = Instr (sTestString, "")
PRINT "Il primo spazio è nella posizione:" &
iPosition FOR iCounter = iPosition TO iLength /
2
    iNextCharPos = iPosition + 1
    iPosition = Instr (sTestString, "", iNextCharPos)
    PRINT "Lo spazio successivo è nella posizione:" &
    iPosition
FINE
SUCCE
SSIIVA

```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

L'output risultante è:

```
abc def ghi è la nostra stringa  
iniziale. Il primo spazio è  
nella posizione: 4 Lo spazio  
successivo è nella posizione: 8  
  
123456  
123456789012345678901234567890123456789012345678901234567890  
abcdefghijklmnpqrstuvwxyz è la nostra nuova stringa di prova.  
  
La lunghezza di sTestString  
è: 51 Il primo spazio è nella  
posizione: 2 Lo spazio  
successivo è nella posizione:
```

Una guida per principianti a

4 Lo spazio successivo è

nella posizione: 6

.

.

**Lo spazio successivo è nella
posizione: 46 Lo spazio
successivo è nella posizione:
48 Lo spazio successivo è
nella posizione: 50**

Una guida per principianti a

RInStr

RInStr restituisce la posizione dell'ultima occorrenza di sottostringa in stringa. Se si specifica Start, la ricerca si ferma alla posizione Start. Se la sottostringa non viene trovata, RInStr () restituisce zero. La sintassi standard del linguaggio Gambas è:

```
Posizione = RInStr (String, Substring [, Start])
```

Inserisci questo codice nella console:

```
STATICO PUBBLICO SUB Principale ()
    DIM sTestString AS String

    DIM iPosition AS Integer

    sTestString = "abc def abc def abc"
    STAMPA "12"
    STAMPA "12345678901234567890"
    PRINT sTestString

    iPosition = RInstr (sTestString,
    "abc") PRINT
    PRINT "l'ultima occorrenza di abc inizia dalla posizione:" &
    iPosition END
```

La console risponde con:

```
12
12345678901234567890
abc def abc def abc

l'ultima occorrenza di abc inizia dalla posizione: 17
```

Diviso

Diviso divide una stringa in sottostringhe delimitate dai separatori designati come parametri. È possibile specificare anche caratteri di escape. Tutti i caratteri separatori racchiusi tra due caratteri di escape vengono ignorati nel processo di divisione. Nota che Split richiede solo tre argomenti, quindi se vuoi usare più separatori, dovrresti passarli come secondo parametro, concatenati in una singola stringa. Per impostazione predefinita, il carattere virgola è il separatore e non sono presenti caratteri di escape. Questa funzione restituisce una matrice di stringhe riempita con ciascuna sottostringa rilevata. La sintassi del linguaggio Gambas è:

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
Array = Split (String [, Separators, Escape])
```

Ecco un programma da provare sulla tua console:

```
STATICO PUBBLICO SUB Principale ()
DIM aWordArray AS String []
DIM sWord AS String
'nota che usiamo un delimitatore di spazio
aWordArray = Split ("Questa è davvero una caratteristica molto
interessante di Gambas!", "")

PER OGNI sWord IN aWordArray
    PRINT sWord
FINE
SUCCE
SSIVA
```

La console risponde con:

```
Questo
è
davver
o un
file
caratte
ristica
molto
interes
sante
di
Gambas !
```

Conversione dei tipi di dati

Asc e Chr \$

Asc restituisce il codice ASCII del carattere nella posizione Position della stringa. Se Position non è specificato, viene restituito il codice ASCII del primo carattere. Chr \$ restituisce il carattere il cui codice ASCII è Codice. Un avvertimento: Gambas utilizza internamente il set di caratteri UTF8, quindi qualsiasi codice carattere maggiore di 128 potrebbe non avere lo stesso significato che avrebbe con un altro set di caratteri (ad esempio, ISO88591). Ecco la sintassi del linguaggio Gambas:

Una guida per principianti a
Codice = Asc (Stringa [, Posizione])
Carattere = Chr \$ (Codice)

Ecco un programma di esempio che illustra l'uso di ASC e CHR \$:

```
"File di classe Gambas
```

Una guida per principianti a

```
STATICO PUBBLICO SUB Principale ()
DIM iAsciiCode AS Integer
DIM sTestString AS String
DIM iLength AS Integer
DIM counter AS Integer

sTestString = "Gambas è
fantastico." iLength = Len
(sTestString)
PRINT "La lunghezza di sTestString è:" & Str $ (iLength)
FOR counter = 1 TO iLength
    iAsciiCode = Asc (sTestString,
    counter) IF iAsciiCode <> 32 THEN
        PRINT iAsciiCode & "is char:" & Chr (9) & Chr $ (iAsciiCode) ELSE
        PRINT iAsciiCode & "è char:" & Chr (9) & "<space>"
    ENDIF
FINE
SUCCE
SSIIVA
```

Nel codice precedente, Chr (9) viene utilizzato per rappresentare il carattere TAB.
L'output risultante dalla console è:

```
La lunghezza di sTestString è: 16
71 è char: G
97 è char: a
109 è fascino
98 è char: b
97 è char: a
115 è char: s
32 è char: <space>
105 è char:     io
115 è char: s
32 è char: <space>
103 è char: g
114 è char: r
101 è char: e
97 è char: a
116 è grafico
46 è char :.
```

Bin \$

Bin \$ restituisce la rappresentazione binaria di un numero. Se si specifica Digits, la

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
rappresentazione viene riempita con zeri non necessari in modo che vengano
restituite le cifre di Digits.

```
String = Bin $ (Number [, Digits])
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Codice di esempio che puoi provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
DIM sBinaryCode AS String
DIM counter AS Integer

FOR counter = 0 TO 15
    sBinaryCode = Bin $(counter,
    4) PRINT sBinaryCode
    FINE
SUCCE
SSIIVA
```

L'output dovrebbe essere simile a questo:

```
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

CBool

CBool converte un'espressione in un valore booleano. Un'espressione è falsa se viene soddisfatta una delle seguenti condizioni:

- ✓ Un valore booleano falso
- ✓ Un numero zero
- ✓ Una stringa di lunghezza zero
- ✓ Un oggetto nullo

Altrimenti, l'espressione è vera in tutti gli altri casi. Ecco la sintassi del linguaggio Gambas:

This product is (C) 2005 by John Rittinghouse, all rights are reserved. It is released under the Gambar User Community under the Open Content License (OCL) and may not be distributed under any other terms or conditions; without the express written consent of the author.

Una guida per principianti a

```
Boolean = CBool (espressione)
```

Ecco un esempio:

```
STATICO PUBBLICO SUB Principale ()
    PRINT CBool (0); ""; CBool (1)
END
```

La console risponde con:

```
FALSA VERITÀ
```

CByte

CByte converte un'espressione in un byte. L'espressione viene prima convertita in un numero intero. Quindi, se questo numero intero supera l'intervallo di byte, (da 32767 a 32768) viene troncato.

```
Byte = CByte (espressione)
```

Esempio:

```
STATIC PUBLIC SUB Main
() PRINT CByte ("17")
STAMPA CByte
(32769) STAMPA
CByte (VERO)
FINE
```

La console risponde con:

```
17
1
255
```

CDate

CDate converte un'espressione in un valore di data / ora. Stai attento! La localizzazione corrente NON viene utilizzata da questa funzione. La sintassi del linguaggio Gambar è:

```
Data = CDate (espressione)
```

Esempio:

Una guida per principianti a

STATICO PUBBLICO SUB Principale ()

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
DIM sDateString AS String
sDateString = CDate
(ora)
PRINT sDateString; "è il nostro orario
di inizio." WAIT 1.0
PRINT CDate (ora); "è un secondo dopo."
PRINT CDate (sDateString); "è ancora dove abbiamo
iniziato." WAIT 1.0
STAMPA ora; "è un secondo dopo."
FINE
```

La console risponde con:

```
21/08/2005 20:52:40 è il nostro orario di
inizio. 21/08/2005 20:52:41 è un secondo dopo.
21/08/2005 20:52:40 è ancora dove abbiamo
iniziato. 21/08/2005 20:52:42 è un secondo
dopo.
```

CFloat

CFloat converte un'espressione in un numero in virgola mobile. Stai attento! La localizzazione corrente NON viene utilizzata da questa funzione. La sintassi del linguaggio Gambas è:

```
Float = CFloat (espressione)
```

Esempio:

```
STATICO PUBBLICO SUB Principale ()
DIM sFloatString AS String
DIM fFloatNum AS Float

sFloatString = "0,99"
fFloatNum = 0,01

PRINT fFloatNum + CFloat
(sFloatString) PRINT CFloat ("3.0E +
3")
FINE
```

La console risponde con:

```
1
3000
```

Una guida per principianti a

CInt / Cinteger e CShort

CInt è sinonimo di CInteger. Ciascuna chiamata converte un'espressione in un numero intero. La sintassi standard del linguaggio Gambas è:

```
Intero = CInt (espressione) Intero  
= CInteger (espressione)
```

CShort converte un'espressione in un numero intero breve. L'espressione viene prima convertita in un numero intero. Quindi, se questo numero intero supera l'intervallo corto, viene troncato. Ecco la sintassi del linguaggio Gambas per CShort:

```
Short = CShort (Expression)
```

Ecco un esempio che modifica leggermente il precedente esempio della console per mostrare entrambe queste funzioni:

```
STATICO PUBBLICO SUB Principale ()  
DIM sFloatString AS String  
DIM fFloatNum AS Float  
  
sFloatString = "0,99"  
fFloatNum = 120,901  
  
PRINT fFloatNum + CFloat (sFloatString)  
PRINT CInt (fFloatNum); "era un float, ora un int."  
PRINT CShort (fFloatNum); "era un galleggiante, ora un  
corto." FINE
```

Il risultato è:

```
121.891  
120 era un float, ora un int.  
120 era un galleggiante, ora un corto.
```

CStr / CString

CStr / CString converte un'espressione in una stringa. Stai attento! La localizzazione corrente NON viene utilizzata da questa funzione. Ecco come funziona la sintassi del linguaggio Gambas per CStr:

```
String = CStr (Expression)  
String = CString (Expression)
```

This product is (C) 2005 by John W. Ritchiehouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Esempio:

```
STATICO PUBBLICO SUB Principale ()
  DIM sFloatString AS String
  DIM fFloatNum AS Float

  fFloatNum = 120,901
  sFloatString = CStr
  (fFloatNum)
  PRINT sFloatString; "ora è una
stringa." FINE
```

L'output della console è:

```
120.901 è ora una stringa.
```

Hex \$

Hex \$ restituisce la rappresentazione esadecimale di un numero. Se si specifica Digits, la rappresentazione viene riempita con zeri non necessari in modo che vengano restituite le cifre di Digits. Ecco la sintassi del linguaggio Gambas:

```
Stringa = esadecimale $ (numero [, cifre])
```

Ecco un programma di esempio:

```
STATICO PUBBLICO SUB Principale ()
Contatore DIM AS Integer

Contatore FOR = da 1 a 15
  STAMPA esadecimale $
  (contatore, 2); ""; IL
PROSSIMO
FINE
```

Questo è il risultato:

```
01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

Conv \$

Conv \$ converte una stringa da un set di caratteri a un altro. Un set di caratteri è rappresentato da una stringa come "ASCII", "ISO88591" o "UTF8". L'interprete GAMBAS utilizza internamente il set di caratteri UTF8. Il set di caratteri utilizzato dal sistema viene restituito da una chiamata a

Una guida per principianti a
System.Charset. È in funzione ISO88591 su Mandrake 9.2

Una guida per principianti a

sistema, ma è UTF8 su un sistema operativo Linux RedHat. In futuro, quasi tutti i sistemi Linux saranno probabilmente basati su UTF8. Il set di caratteri utilizzato dall'interfaccia utente grafica viene restituito da una chiamata a Desktop.Charset. È UTF8 con il componente Qt. La sintassi del linguaggio Gambas è:

```
ConvertedString = Conv $ (String AS String, SourceCharset AS String,  
DestinationCharset AS String)
```

La conversione si basa sulla funzione di libreria GNU iconv (). Ecco un esempio di codice da provare:

```
STATICO PUBBLICO SUB Principale ()  
DIM sStr AS String  
DIM iInd AS Interger  
  
sStr = Conv $ ("Gambas", "ASCII", "EBCDICUS")  
FOR iInd = 1 TO Len (sStr)  
    STAMPA Hex $ (Asc (Mid $ (sStr, iInd, 1)),  
2); ""; IL PROSSIMO  
FINE
```

Ecco l'output:

```
C7 81 94 82 81 A2
```

Val e Str \$

Val converte una stringa in un valore booleano, un numero o una data, in base al contenuto della stringa. La localizzazione corrente viene utilizzata per convertire numeri e date. L'algoritmo di conversione utilizza il seguente ordine di precedenza:

Se la stringa può essere interpretata come data e ora (con separatori di data o ora), quindi vengono restituite la data e l'ora.

Altro se la stringa può essere interpretata come un numero in virgola mobile, viene restituito un numero in virgola mobile.

Altrimenti se stringa può essere interpretata come un numero intero, quindi viene restituito un numero intero.

Altrimenti se stringa è TRUE o FALSE, viene restituito il valore booleano corrispondente.

Altro Viene restituito NULL.

La sintassi standard del linguaggio Gambas è:

```
Espressione = Val (stringa)
```

his product is (C) 2005 by John W. Ritterhouse, all rights are reserved. It is released to the Gambas User Community under an OpenContent License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Str \$ converte un'espressione nella sua rappresentazione di stringa stampabile. È l'esatto contrario di Val (). La localizzazione corrente viene utilizzata per convertire numeri e date. La sintassi del linguaggio Gambas è:

```
String = Str $ (espressione)
```

Prova questo:

```
STATICO PUBBLICO SUB Principale ()
DIM sDateTime AS String
DIM dDate AS Date

STAMPA ora; "è l'ora di sistema
corrente." sDateTime = Val (Str $
(Now))
PRINT sDateTime; "è una rappresentazione di stringa dell'ora di sistema
corrente."
PRINT Val (sDateTime); "è VAL conv della rappresentazione di stringa"
dDate = Val (sDateTime)
PRINT dDate; "è la variabile DATE convertita con VAL."
PRINT Str (dDate); "è una rappresentazione in formato stringa della
variabile data." FINE
```

Ecco l'output:

```
21/08/2005 21:42:45 è l'ora di sistema corrente.
21/08/2005 21:42:45 è una rappresentazione di stringa dell'ora di sistema
corrente. 21/08/2005 21:42:45 è VAL conv della rappresentazione di
stringa.
21/08/2005 21:42:45 è la variabile DATE convertita con VAL.
21/08/2005 21:42:45 è una rappresentazione di stringa della variabile
data.
```

Ecco un altro programma di esempio da provare:

```
"File di classe Gambas

STATICO PUBBLICO SUB

Principale ()

DIM sInputLine AS String
Valore DIM AS Variant
```

Una guida per principianti a

```
DO WHILE sInputLine <> "quit"
    PRINT "==>";
    LINE INPUT sInputLine
    IF sInputLine = "" THEN
        sInputLine = "<CRLF>"
    ENDIF
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

```
PRINT "Hai digitato:"; Valore
sInputLine = Val (sInputLine)
STAMPA
SE IsBoolean (valore) ALLORA
    PRINT sInputLine; "è booleano."
ELSE IF IsDate (valore) THEN
    PRINT sInputLine; "è un
appuntamento". ELSE IF IsInteger
(valore) THEN
    PRINT sInputLine; "è un numero
intero." SE valore = 0 OR valore =
1 ALLORA
    PRINT "Potrebbe anche essere
booleano." FINISCI SE
    IF valore> 0 AND valore <255 THEN
        PRINT "Potrebbe anche essere un
byte."
    FINISCI SE
    IF value> 32767 AND value <32768 THEN
        PRINT "Potrebbe anche essere un
corto."
    FINISCI SE
    STAMPA ""
ALTRIMENTI SE IsFloat (valore) ALLORA
    PRINT sInputLine; "è un
galleggiante." ELSE IF IsString
(value) THEN
    PRINT sInputLine; "è una
stringa." ELSE IF IsNull (value) THEN
    PRINT sInputLine; "è zero."
ALTRO
    PRINT sInputLine; "è qualcos'altro."
FINISCI SE
STAMP
A
LOOP
STAMPA "Abbiamo
finito!" FINE
```

L'output dovrebbe essere simile a questo:

```
==> vero
Hai digitato: vero

true è booleano.
==> 214
Hai digitato: 214

214 è un numero intero.
```

Una guida per principianti a

Potrebbe anche essere un
byte. Potrebbe anche
essere un breve.

==> 23/08/05 12:23:55

Hai digitato: 23/08/05 12:23:55

This product is (C) 2005 by John W. Rittinghouse and is released under the terms of the Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
23/08/05 12:23:55 è un appuntamento.  
==> John  
Hai digitato: John  
  
John è una stringa.  
==> 32756  
Hai digitato: 32756  
  
32756 è un numero intero.  
Potrebbe anche essere un breve.  
-----  
==> esci  
Hai digitato:  
  
quit quit è una  
stringa. Abbiamo  
chiuso!
```

Formato \$

Format \$ converte un'espressione in una stringa utilizzando un formato che dipende dal tipo di espressione. Il formato può essere un formato predefinito (una costante intera) o un formato definito dall'utente (una stringa che rappresenta il formato). Se Format non è specificato, vengono utilizzati i formati dei componenti gb.Standard. Questa funzione utilizza le informazioni di localizzazione per formattare date, ore e numeri. La sintassi del linguaggio Gambas è:

```
Stringa = Format $ (Espressione [, Formato])
```

Un formato numerico definito dall'utente è descritto dai seguenti caratteri:

| Simbolo di formato | Significato |
|--------------------|--|
| "+" | stampa il segno del numero. |
| "" | stampa il segno del numero solo se negativo. |
| "#" | stampa una cifra solo se necessario. |
| "0" | stampa sempre una cifra, riempendola con uno zero se necessario. |
| "." | stampa il separatore decimale. |
| "%" | moltiplica il numero per 100 e stampa un segno di percentuale. |
| "E" | introduce la parte esponenziale di un numero float. Il segno dell'esponente è sempre stampato. |

Una guida per principianti a

Un formato di data definito dall'utente è descritto dai seguenti caratteri:

| Simbolo di formato | Significato |
|--------------------------------|--|
| "yy" | stampa l'anno utilizzando due cifre. |
| "aaaa" | stampa l'anno utilizzando quattro cifre. |
| "m" | stampe mese. |
| "mm" | stampa il mese utilizzando due cifre. |
| "mmm" | stampa il mese in forma di stringa abbreviata. |
| "mmmm" | stampa il mese nella sua forma di stringa completa. |
| "d" | stampa il giorno. |
| "dd" | stampa il giorno utilizzando due cifre. |
| "ddd" | stampa il giorno della settimana in forma abbreviata. |
| "dddd" | stampa il giorno della settimana nella sua forma completa. |
| "/" | stampa il separatore della data. |
| "h" | stampe ora. |
| "hh" | stampa l'ora utilizzando due cifre. |
| "n" | stampa i minuti. |
| "nn" | stampa i minuti utilizzando due cifre. |
| "S" | stampa i secondi. |
| "ss" | stampa i secondi utilizzando due cifre. |
| Stampa un separatore dell'ora. | |

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Ganga User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Ecco un programma che mostra diversi esempi:

```
STATICO PUBBLICO SUB Principale ()
STAMPA ""
PRINT "Esempi di formati numerici definiti
dall'utente:" PRINT Format $ (Pi, "#. ###")
PRINT Format $ (Pi, "+0 #. ### 0")
PRINT Format $ (Pi / 10, "###.
#%") PRINT Format $ (11 ^ 11, "#.
## E ##")
STAMPA ""
PRINT "Esempi di formati di data e ora definiti
dall'utente:" PRINT
PRINT Format $ (ora, "mm / dd / yyyy hh:
nn: ss") PRINT Format $ (ora, "m / d / yy
h: n: s")
Formato PRINT $ (ora, "gg gg mmmm aaaa")
Formato STAMPA $ (ora, "gg gg mmmmm aaaa")
FINE
```

This product is (C) 2005 by John W. Minninghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Ecco l'output:

```
-----
Esempi di formati numerici definiti
dall'utente: 3.142
+03.1416
31,4%
2,85 E + 11
-----
Esempi di formati di data e ora definiti
dall'utente: 21/08/2005 21:55:14
21/08/05 21:55:14
Domenica 21 agosto 2005
Domenica 21 agosto 2005
```

Gestione dei tipi di dati

Molte volte durante il controllo delle variabili, è importante sapere con quale tipo di dati si ha a che fare. Gambas fornisce una serie di funzioni a tale scopo. È possibile verificare i seguenti tipi di dati:

| | |
|-----------------------------|----------------------------|
| IsBoolean / Boolean? | IsByte / Byte? |
| IsDate / Data? | IsFloat / Float? |
| IsInteger / Intero? | IsNull / Nullo? |
| IsNumber / Numero? | IsObject / Oggetto? |
| È corto / Short? | IsString / Corda? |

Ciascuna delle funzioni precedenti viene chiamata utilizzando la sintassi del linguaggio Gambas standard di

```
BooleanResult = IsBoolean?
(Espressione) BooleanResult = IsByte?
(Espressione) BooleanResult = IsDate?
(Espressione) BooleanResult = IsFloat?
(Espressione) BooleanResult =
IsInteger? (Espressione) BooleanResult
= IsNull? (Espressione) BooleanResult =
IsNumber? (Espressione) BooleanResult
= IsObject? (Espressione) BooleanResult
= IsShort? (Espressione) BooleanResult
= IsString? (Espressione)
```

e la funzione data restituirà TRUE se un'espressione è del tipo di dati richiesto o FALSE se non lo è.

Una guida per principianti a

Tipo di

TypeOf restituisce il tipo di un'espressione come valore intero. Gambas ha definito diverse costanti predefinite per i tipi di dati restituiti da questa funzione. Le costanti predefinite supportate da Gambas sono mostrate nella tabella seguente. La sintassi standard del linguaggio Gambas per questa funzione è:

```
Type = TypeOf (Expression)
```

Costanti del tipo di dati predefinite

| Tipo di dati | Valore |
|--------------|--------------------------|
| gb.Null | Valore nullo |
| gb.Boolean | Valore booleano |
| gb.Byte | Numero intero byte |
| gb.breve | Numero intero breve |
| gb.Integer | Numero intero |
| gb.Float | Numero in virgola mobile |
| gb.Date | Valore di data e ora |
| gb.String | Stringa di caratteri |
| gb.Variant | Variante |
| gb.Object | Riferimento oggetto |

Questo è più o meno tutto ciò che tratteremo in questo capitolo. Nel prossimo capitolo, torneremo al Gambas ToolBox e inizieremo a imparare a usare alcuni dei controlli più avanzati, come iconview, listview, ecc. Ora dovresti essere adeguatamente preparato per gestire le tecniche di gestione dei dati più avanzate richieste per quei controlli.

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the Gambas User Community under an Open Content License (OCL). It may not be distributed or modified without the express written consent of the author.

Una guida per principianti a

Capitolo 8 - Utilizzo dei controlli avanzati

In questo capitolo studieremo alcuni dei controlli più avanzati, come i controlli IconView, ListView, GridView, ColumnView e Tabstrip e impareremo come usarli. Per il controllo IconView, questa volta adotteremo un approccio diverso, utilizzando uno dei programmi di esempio forniti da Gambas. Esamineremo il programma Explorer scritto dal creatore di Gambas Benoît Minisini e esamineremo ogni riga di codice per capire esattamente cosa sta succedendo nel programma e come viene utilizzato il controllo IconView.

Controllo IconView

Innanzitutto, avvia Gambas e seleziona il progetto Explorer dalla sezione Miscellaneous dei Programmi di esempio:



Figura 54 Scelta dell'esempio Explorer.

Una volta selezionato il progetto Explorer, l'IDE si aprirà e dovremo accedere al codice. Puoi farlo facendo doppio clic sul file di classe o, se il modulo è già attivo, fai doppio clic su un controllo e quindi spostati all'inizio del

Una guida per principianti a

finestra del codice. In entrambi i casi funzionerà fintanto che inizi nella parte superiore della finestra del codice dove vedi la prima riga:

```
"File di classe Gambas
```

La prima cosa che vediamo fatta in questo codice è la dichiarazione delle variabili globali del programma. Nota che sono preceduti da un carattere "\$" per maggiore chiarezza. Il

`$ sPath` variabile è dichiarata come PRIVATA e utilizzata per contenere il nome del percorso del file corrente.

```
PRIVATE $ sPath AS String
```

`$ b Nascosto` è una variabile booleana PRIVATA che fungerà da flag da utilizzare per determinare SE un file è nascosto o meno. Useremo la funzione Stat per controllare il suo stato.

```
PRIVATO $ b Nascosto como booleano
```

`$ bCtrl` è un booleano PRIVATO utilizzato come flag quando viene premuto un tasto CTRL. Se l'utente tiene premuto il tasto CTRL quando fa doppio clic su una cartella, si aprirà in una nuova finestra.

```
PRIVATO $ bCtrl AS Booleano
```

Questa è la prima subroutine che Gambas eseguirà quando il programma viene eseguito:

```
STATICO PUBBLICO SUB Principale ()
```

Le due righe successive dichiareranno una variabile locale per il form che vogliamo mostrare all'utente e creeranno un'istanza del form chiamata FExplorer, passando il parametro System.Home a una routine del costruttore, che è identificata come `_new ()`, e che accetta un parametro stringa `sPath` (System.Home) il cui valore abbiamo ottenuto dalla classe System.

```
DIM hForm AS Form
hForm = NUOVO FExplorer (System.Home)
```

Ora mostra il modulo:

```
hForm.Show
```

```
END
```

This product is (C) 2005 by John W. Ratinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

La subroutine successiva è il costruttore che viene chiamato quando viene attivata la subroutine Main () e viene istanziato il form FExplorer:

```
PUBLIC SUB _new (sPath AS String)
```

Assegneremo il percorso passato in (System.Home) come parametro alla nostra variabile globale \$ sPath:

```
$ sPath = sPath
```

Infine, dobbiamo chiamare la subroutine RefreshExplorer () per popolare il controllo iconview denominato ivwExplorer per aggiornare la visualizzazione per il nuovo controllo:

```
RefreshExplorer  
END
```

Ecco la subroutine RefreshExplorer (). È essenzialmente la carne del programma:

```
PRIVATE SUB RefreshExplorer ()
```

Per prima cosa, dichiariamo il local variabili inizia con una stringa var per contenere i nomi dei file:

```
DIM sFile AS String
```

Quindi, dichiara le variabili immagine per le nostre immagini icona: directory principale, cartella e file:

```
DIM hPictDir AS Picture  
DIM hPictParDir AS Picture 'aggiunto dall'autore  
DIM hPictFile AS Picture
```

cDir è un array di stringhe (che rappresentano i nomi di file o directory):

```
DIM cDir AS NEW String []
```

sName è una stringa di lavoro utilizzata per rappresentare i nomi di file trovati in una directory:

```
DIM sName AS String
```

La prossima dichiarazione o non è documentata in Gambas o la documentazione è irraggiungibile sul sito web di Gambas. Tuttavia, questa chiamata

Una guida per principianti a
sembra incrementare il valore della proprietà Busy definito nella classe Application.
It

Una guida per principianti a

sembra essere usato come un semaforo in modo che altri processi possano testare il flag prima di provare a ottenere il tempo del computer (ricorda, la maggior parte delle applicazioni utilizza una sorta di routine di pianificazione CPU roundrobin implementata dal sistema operativo per condividere le risorse del processore). Qui, è fondamentalmente un flag che dice a tutti gli altri processi chiamanti che devono aspettare fino a quando questo processo non è più occupato (indicato diminuendo il valore della proprietà Busy). Questo sembra essere il modo in cui Gambas imposta i flag per impedire l'interruzione dei processi critici.

Applicazione INC

Una volta lanciato il flag "non interrompermi", impostiamo il titolo della finestra sul percorso di sistema e chiamiamo la funzione Conv \$ 14 incorporata per convertire il set di caratteri di sistema (il set di caratteri che il sistema operativo ha fornito come predefinito) in quello che il user ha definito come il set di caratteri del desktop (il set di caratteri che l'utente ha scelto di vedere dal proprio desktop). Nota che Conv è un sinonimo di Conv \$ e puoi usarne uno in modo intercambiabile.

```
ME.Title = Conv ($ sPath, System.Charset, Desktop.Charset)
```

Ora, tutto ciò che può esistere nella vista a icone viene cancellato chiamando il metodo Clear di iconview:

```
ivwExplorer.Clear
```

Successivamente, assegna le icone alle variabili immagine che abbiamo dichiarato. Abbiamo un'icona per rappresentare le cartelle e una per i file:

```
hPictDir = Immagine ["folder.png"]
hPictParDir = Immagine ["ParentFolder.png"] 'aggiunto dall'autore
hPictFile = Immagine ["file.png"]
```

Se la variabile di percorso globale non è impostata al livello più alto (cioè il genitore) indicato dalla stringa "/", allora vogliamo creare una cartella su cui l'utente può cliccare per andare alla directory genitore del figlio corrente. Aggiungiamo questa cartella al controllo iconview denominato ivwExplorer utilizzando il suo metodo Add:

```
IF $ sPath <> "/" THEN ivwExplorer.Add ("D ..", "...", hPictParDir)
```

Nell'istruzione precedente, il metodo .Add accetta tre parametri. Il primo è il nome della chiave, in questo caso "D .." che è la nostra chiave per le directory. Il secondo è il testo posizionato sotto l'icona nel controllo IconView e il finale

Una guida per principianti a

14 Le funzioni stringa comuni non si occupano molto bene di UTF-8. In UTF-8, un carattere può avere 1-3 byte. La stringa Gambasi metodi di classe sono in grado di gestire UTF-8, quindi dovrebbero essere usati ogni volta che è possibile.

Una guida per principianti a

parametro è il nome della variabile che è un handle per il nome del file dell'icona che vogliamo visualizzare con questa voce. Ora inizieremo a scorrere ogni file nella directory corrente per vedere se è un file nascosto o meno e per contrassegnare ogni nome di file come una directory o un file.

```
PER OGNI sFile IN Dir ($ sPath)
```

Gambas inizializza le stringhe su null quando vengono create, quindi la prima volta che veniamo qui, il valore \$ bHidden non sarà TRUE e l'istruzione IF verrà eseguita nel codice seguente:

```
SE NON $ b Nascosto ALLORA
```

La prossima riga di codice è un po 'più complicata da decifrare. La funzione Stat accetta una stringa del nome del file come parametro. In questo caso, il percorso corrente contenuto nella variabile globale \$ sPath viene catenato con la stringa di lavoro sFile utilizzando la combinazione di simboli & /. Questo è un simbolo di catenazione speciale utilizzato in Gambas specificamente per catenare i nomi dei file. La funzione Stat viene chiamata con il nome del file catenato passato come parametro e contemporaneamente viene controllata la proprietà Stat.Hidden. Se la proprietà Stat.Hidden è impostata su un valore TRUE, viene eseguita l'istruzione CONTINUE, forzando il flusso del programma all'iterazione successiva del ciclo FOR. L'intero processo costringe fondamentalmente il programma a ignorare i file nascosti che incontra.

```
IF Stat ($ sPath & / sFile) .Hidden THEN 'è nascosto?  
    CONTINUA 'in tal caso, vai all'iterazione del ciclo successivo  
    FINISCI SE 'IF Stat  
FINISCI SE 'SE NON $ bHidden
```

Se abbiamo raggiunto questo punto nel codice, il file non era nascosto. Ora, useremo la funzione di gestione dei file incorporata IsDir per vedere se il percorso corrente e la stringa del file (catenata nella chiamata Stat) è una cartella o un file. Se abbiamo una cartella, la aggiungeremo all'array di stringhe cDir, prima taggandola con una lettera 'D' per la directory o 'F' per il file e poi aggiungendo il nome del file contenuto nella stringa di lavoro sFile alla 'D' o ' Tag F':

```
SE IsDir ($ sPath & / sFile) ALLORA 'era una directory  
    cDir.Add ("D" & sFile) 'lo aggiungiamo alla chiave delle directory  
ALTRO 'non è una directory e lo aggiungiamo al tasto "F" come file  
    cDir.Add ("F" &  
    sFile) ENDIF  
IL PROSSIMO 'questa è la fine del ciclo FOR ...
```

Una volta che tutto è stato caricato nell'array cDir con il ciclo FOR / NEXT,

Una guida per principianti a
chiamiamo il metodo Sort incorporato e ordiniamo l'array di stringhe:

Una guida per principianti a

```
cDir.Sort
```

Ora che l'array cDir è ordinato, eseguiremo un ciclo attraverso l'array utilizzando l'istruzione FOR EACH che viene utilizzata specificamente per gli oggetti enumerati in array o raccolte.

```
PER OGNI sFile IN cDir
```

Per ogni stringa nell'array cDir, riempiremo la stringa di lavoro sName con il nome del file. Tuttavia, dobbiamo prima rimuovere il tag. Useremo la funzione stringa incorporata Mid \$ per prendere tutte le lettere della stringa dopo il primo (il nostro tag) a partire dalla posizione 2 nella stringa:

```
sName = Mid $(sFile, 2)
```

Ora controlliamo il tag del nome del file usando la funzione stringa Left \$:

```
SE a sinistra $(sFile) = "D" ALLORA
```

Se viene trovato il tag "D", si tratta di una directory e aggiungiamo una directory al controllo IconView utilizzando il metodo Add e la nostra immagine di una cartella come ultimo parametro nella chiamata.

```
ivwExplorer.Add (sFile, sName, hPictDir)  
ALTRO 'altrimenti era un file e lo aggiungiamo con un'icona di file'  
ivwExplorer.Add (sFile, sName,  
hPictFile) ENDIF
```

Vogliamo anche impostare la proprietà IconView.Edit su TRUE in modo che l'utente possa rinominare gli elementi:

```
ivwExplorer.Item.Editable = TRUE  
NEXT 'uscire dal ciclo FOR EACH'
```

Le istruzioni seguenti sono state commentate ma sembrano essere state utilizzate per modificare l'ordinamento del controllo IconView. Salteremo semplicemente questi commenti e andremo alla dichiarazione FINALLY:

```
'ivwExplorer.Sorted = FALSE'  
ivwExplorer.Ascending = TRUE  
'ivwExplorer.Sorted = TRUE'
```

```
INFINE 'questa è l'ultima istruzione da eseguire nella subroutine'
```

Una guida per principianti a

L'ultima cosa che dobbiamo fare è riportare il flag Application.busy allo stato normale:

```
DEC Application.busy
```

Se si verifica un errore, lo individueremo con la clausola catch di seguito:

CATTURARE

Se si verifica un errore, dovrebbe apparire un MessageBox con il testo dell'errore visualizzato:

```
Message.Error (Error.Text)
```

```
FINE 'della nostra routine RefreshExplorer
```

Questa subroutine viene chiamata se il modulo FExplorer viene ridimensionato generando un evento di ridimensionamento. Sosterà il nostro controllo IconView nell'angolo in alto a sinistra e regolerà larghezza e altezza per adattarsi alle nuove dimensioni della finestra:

```
PUBLIC SUB Form_Resize ()  
    ivwExplorer.Move (0, 0, ME.ClientW, ME.ClientH)  
END
```

Se l'utente sceglie l'opzione di chiusura dal menu, questa routine esegue:

```
PUBLIC SUB mnuQuit_Click  
    () ME.Close  
FINE
```

Se l'utente seleziona l'opzione Aggiorna, chiameremo la subroutine RefreshExplorer descritta in precedenza.

```
PUBLIC SUB mnuViewRefresh_Click  
    () RefreshExplorer  
FINE
```

Questa subroutine viene chiamata quando l'evento di attivazione viene attivato da un utente facendo doppio clic su una cartella nel controllo IconView:

```
PUBLIC SUB ivwExplorer_Activate ()
```

Come sempre, dobbiamo dichiarare le nostre variabili locali. Dichiariamo sNewPath come una stringa per contenere il percorso del file per il percorso di

Una guida per principianti a destinazione su cui l'utente ha fatto clic (una nuova cartella o la cartella principale). Inoltre, dichiariamo un nuovo modulo per mostrare il contenuto della destinazione, hForm in una finestra separata se il tasto control viene premuto quando la destinazione è

Una guida per principianti a

selezionato.

```
DIM sNewPath AS String
DIM hForm AS Form
```

Questo se il controllo guarda il tag che mettiamo sulla stringa per vedere se la destinazione è una directory o il livello di root del sistema. Se siamo al livello radice, torniamo semplicemente. In caso contrario, il codice assegnerà i valori catenati di \$ sPath e il valore mantenuto da LAST.Current.Key del controllo IconView. Ciò si ottiene con la chiamata Mid \$ a partire dalla posizione 2 (per bypassare il nostro carattere di tag).

```
IF LAST.Current.Key = "D .."
  THEN IF $ sPath = "/" THEN
    RETURN
  s newPath = File.Dir ($
sPath) ALTRO
  s newPath = $ sPath & / Mid $ (LAST.Current.Key,
2) ENDIF
```

Controlliamo il tipo di dati del valore stringa appena assegnato per assicurarci che sia, in effetti, una directory e, se lo è, controlleremo successivamente per vedere se l'utente ha tenuto premuto il tasto control. Ricorda, tenendo premuto il tasto control nel nostro programma attiverà una nuova finestra, motivo per cui abbiamo dichiarato la variabile hForm locale.

```
SE IsDir (s newPath) ALLORA
```

Se il tasto control è stato tenuto premuto, ripristineremo il valore su FALSE prima di creare un'istanza di una nuova finestra. Quindi, sposteremo il nostro controllo in modo che venga spostato di 16 pixel a destra e sotto la finestra corrente, con la stessa altezza e larghezza utilizzando il metodo hForm.Move. Infine, mostriamo il modulo e aggiorniamo Explorer con la nostra subroutine RefreshExplorer.

```
IF $ bCtrl ALLORA
  $ bCtrl = FALSO
  hForm = NUOVO FExplorer (s newPath)
  hForm.Move (ME.X + 16, ME.Y + 16, ME.W,
  ME.H)
  hForm.Show
ELSE
```

Altrimenti, il tasto control non è stato tenuto premuto, quindi assegniamo semplicemente il valore s newPath al \$ sPath globale e aggiorniamo l'explorer:

Una guida per principianti a

```
$ sPath =  
$sNewPath  
AggiornaExplorer  
ENDIF  
ENDIF
```

Una guida per principianti a

FIN
E

Quando l'utente fa clic su questa routine, chiama la nostra subroutine per mostrare il file
file nascosti o, se vengono visualizzati, disattivare la visualizzazione dei file nascosti:

```
PUBLIC SUB mnuViewHidden_Click  
    () ToggleViewHidden  
FINE
```

Se l'utente sceglie di mostrare i file nascosti nell'explorer selezionando l'opzione Mostra file nascosti, viene eseguita la routine successiva. La prima riga controlla il valore della proprietà mnuViewHidden.Checked e alterna il valore essenzialmente facendo diventare \$ bHidden qualunque sia il valore logico NOT di mnuViewHidden.Checked al momento in cui viene controllato. Quindi assegna il valore opposto alla proprietà e aggiorna la visualizzazione chiamando RefreshExplorer. Il risultato è che vengono visualizzati i file nascosti.

```
PRIVATE SUB ToggleViewHidden ()  
    $ bHidden = NON  
    mnuViewHidden.Checked  
    mnuViewHidden.Checked = $ bHidden  
    AggiornaExplorer  
FINE
```

Questa subroutine viene eseguita quando la voce relativa al menu è cliccata. It visualizza semplicemente un MessageBox che accredita il codice al fondatore di Gambas, B. Minisini:

```
PUBLIC SUB mnuAbout_Click ()  
    Message ("Esempio di IconView scritto da \ nBenoît  
Minisini") END
```

La subroutine successiva viene eseguita quando l'utente fa un singolo clic su un elemento nell'explorer e sceglie di rinominare l'elemento. Come è stato scritto originariamente, non controlla se la ridenominazione ha effettivamente modificato l'elemento. Ad esempio, se l'utente premesse ESC, l'elemento tornerebbe al suo nome originale.

```
PUBLIC SUB ivwExplorer_Rename ()  
  
    Messaggio ("'" & Mid $ (LAST.Item.Key,  
        2) &  
        "'è stato rinominato in'" & LAST.Item.Text & "")  
FINE
```

This product is (C) 2001 Rittinghouse. All rights are reserved. It is licensed to the Cambas User Community under the Open Content License (OCL) and may only be distributed under any other terms with the express written consent of the author.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved.
Gambas User Community under an Open Content License (OCL) and
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Possiamo porre rimedio a questa svista semplicemente cambiando il codice in questo modo:

```
PUBLIC SUB ivwExplorer_Rename ()
  IF Mid $ (LAST.Item.Key, 2) <> Last.Item.Text
  THEN Message ("'" & Mid $ (LAST.Item.Key, 2) &
    "'è stato rinominato in'" & LAST.Item.Text & "'")
  ALTRO
  Messaggio ("'" & Mid $ (LAST.Item.Key, 2) & "' è rimasto
  invariato. '') ENDIF
FINE
```

Le ultime due subroutine alternano la variabile globale \$ bCtrl ogni volta che si preme o si rilascia il tasto control.

```
PUBLIC SUB ivwExplorer_KeyPress ()
  IF Key.Control THEN $ bCtrl =
  TRUE
FINE

PUBLIC SUB ivwExplorer_KeyRelease ()
  IF Key.Control THEN $ bCtrl =
  FALSE
FINE
```

Ecco qua. Tutto ciò che devi sapere sull'utilizzo del controllo IconView in una fantastica applicazione. Successivamente, esamineremo il controllo ListView.

Controllo ListView

Il controllo ListView eredita Control e implementa un elenco di elementi di testo selezionabili con icone. Gli elementi ListView sono indicizzati da una chiave. Visualizzano una stringa e un'icona. Questo controllo ha un cursore interno utilizzato per accedere ai suoi elementi. È necessario utilizzare i metodi Move (MoveAbove, MoveBelow, MoveCurrent, MoveFirst, MoveLast, MoveNext, MovePrevious, MoveTo) per spostare il cursore interno ed è necessario utilizzare la proprietà Item per ottenere il testo dell'elemento a cui punta il cursore. Questa classe è creabile e la sintassi standard del linguaggio Gambas è:

```
DIM hListView AS ListView
hListView = NEW ListView (Parent AS Container)
```

Il codice precedente crea un nuovo controllo ListView che agisce come una matrice di sola lettura. Creiamo un programma che implementerà il controllo

Una guida per principianti a ListView. Creare un nuovo progetto denominato ListView che è un progetto di tipo interfaccia utente grafica. Quando l'IDE viene visualizzato dopo aver eseguito la procedura guidata del progetto, creare un modulo semplice simile a questo:

This product is © 2000 J.W. Rittinghouse, all rights are reserved. It is illegal to copy or distribute this product without the express written consent of the author.

Una guida per principianti a

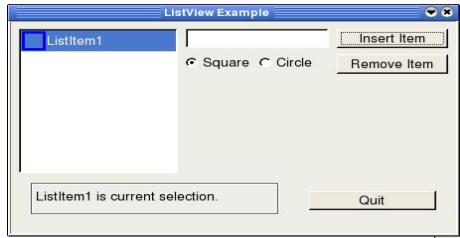


Figura 55 Layout per il nostro esempio ListView.

Nella parte superiore sinistra dell'immagine, hai il nostro controllo ListView denominato ListView1, una casella di testo denominata Textbox1 e il pulsante Inserisci elemento denominato Button1. Sono presenti due RadioButtons, denominati RadioButton1 e RadioButton2 e il pulsante Rimuovi elemento denominato Button2. In fondo al modulo c'è un TLabel denominato TLabel1 e il pulsante Esci denominato Button3. Dopo aver creato il modulo e averlo reso una classe di avvio, è necessario aggiungere il seguente codice alla finestra del codice per vedere come implementare un controllo ListView:

```
"File di classe
Gambas sStatus as
String

PUBLIC SUB Form_Open ()
    DIM picSquare COME NUOVA
    Immagine DIM picCircle COME
    NUOVA Immagine

    picCircle.Load ("circle.png")
    picSquare.Load ("square.png")
    'Questo aggiungerà un elemento a ListView con una voce iniziale
    ListView1.Add ("ListItem1", "ListItem1", picSquare)
    TLabel1.Text = ListView1.Item.Text
    ListView1_Click
END
```

Quando il programma inizia e il modulo viene visualizzato per la prima volta sullo schermo, viene eseguita la routine precedente. Creiamo due variabili locali per le nostre icone che verranno utilizzate nell'elenco e le carichiamo in memoria. Successivamente, aggiungiamo una chiave e un elemento predefiniti all'elenco utilizzando il metodo ListView1.Add, specificando l'icona picSquare da associare a questo elemento. Non preoccuparti per le icone per ora: le creeremo per ultime. Successivamente, dobbiamo creare un evento per aggiornare l'elenco ogni volta che un elemento viene aggiunto o viene cliccato dall'utente.

```
PUBLIC SUB ListView1_Click ()
    ListView1.MoveCurrent
    ListView1.Item.Selected = TRUE
```

Una guida per principianti a

```
TextLabel1.Text = ListView1.Item.Text e sStatus  
END
```

Una guida per principianti a

Nella subroutine ListView1_Click, la prima riga di codice sposta il cursore interno sull'elemento più corrente e lo evidenzia impostando la proprietà Item.Selected su TRUE nella seconda riga. Infine, la terza riga aggiorna il nostro valore TextLabel1.Text con il testo della selezione corrente (o appena aggiunta) e lo stato corrente (contenuto nella variabile globale sStatus AS String) aggiunto. Se l'utente fa clic sul pulsante Inserisci elemento (lo abbiamo chiamato Button1), viene eseguita la seguente routine:

```
Pulsante PUBLIC SUB1_Click ()
```

Dichiara una variabile locale, picToUse, per la visualizzazione della nostra icona:

```
IMMAGINE DIM UTILIZZABILE COME NUOVA Immagine
```

Quindi, controlla quale radioButton è stato cliccato è stato cliccato il primo pulsante, caricheremo l'immagine per square.png, altrimenti viene caricato circle.png.

```
IF Textbox1.Text <> NULL THEN
  IF RadioButton1.Value THEN
    picToUse.Load
    ("square.png") ELSE
    picToUse.Load
    ("circle.png") END IF
```

Ottieni l'elemento corrente (o non fare nulla) se ListView è vuoto:

```
ListView1.MoveCurrent ()
```

Ora aggiungeremo una nuova voce con una chiave e un nome nella casella di testo:

```
ListView1.Add (Textbox1.Text, Textbox1.Text, picToUse)
```

Questo svuota la casella di testo:

```
TextBox1.Text = ""
sStatus = "corrente." 'imposta lo stato su "corrente"
```

Questa chiamata alla subroutine ListView1_Click aggiornerà (aggiornerà) il nostro controllo ListView:

```
ListView1_Click
```

Successivamente, dobbiamo assicurarci che il nuovo elemento si trovi nell'area visibile del controllo:

This product is (C) by John W. Rittinghouse, all rights reserved. It is released to the Game's User Community under an Open Content License (CC-BY), and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
ListView1.Item.EnsureVisible  
END IF  
FINE
```

La subroutine Button2_Click viene chiamata ogni volta che l'utente decide di eliminare l'elemento attualmente selezionato nel controllo ListView.

```
PUBLIC SUB Button2_Click ()
```

La prossima riga di codice fa uscire il cursore allineato alla selezione corrente. Il metodo MoveCurrent sposta il cursore interno sull'elemento corrente. Restituisce un valore VERO se non è presente alcun elemento, nel qual caso torneremo semplicemente poiché non c'è nulla da eliminare:

```
IF ListView1.MoveCurrent () THEN RETURN
```

Se abbiamo raggiunto questo punto, abbiamo spostato il cursore su e dobbiamo rimuovere l'elemento corrente del cursore:

```
ListView1.Remove (ListView1.Item.Text)
```

Pulisci la nostra visualizzazione TextLabel1.Text con questa chiamata:

```
TextLabel1.Text = ""
```

Ora, dobbiamo aggiornare la posizione del cursore al nuovo elemento corrente (poiché ora siamo puntati su un elemento eliminato). Ma prima, dobbiamo controllare la proprietà .Count per vedere se abbiamo appena cancellato l'ultimo elemento nell'elenco. Se il conteggio è maggiore di zero, ci sono elementi rimasti nell'elenco. In caso contrario, l'istruzione IF non sarà vera e il codice verrà ignorato:

```
SE ListView1.Count > 0  
  
ALLORA  
  
    ListView1.MoveCurrent  
  
    ListView1.Item.Selected = TRUE 'Selezione l'elemento  
  
    corrente sStatus = "selected."  
  
    ListView1_Click 'questo forzerà un aggiornamento  
  
FINISCI SE  
FIN  
E
```

Se l'utente fa clic con il mouse su un elemento nel controllo ListView, this

Una guida per principianti a

si chiama routine. Aggioreremo semplicemente la proprietà TextLabel1.Text con il testo dell'elemento selezionato e aggioreremo il controllo ListView chiamando la nostra subroutine ListView1_Click.

```
PUBLIC SUB ListView1_Select ()  
    TextLabel1.Text = ListView1.Item.Text  
    sStatus = "selezionato".  
    ListView1_Click  
END
```

Se l'utente fa doppio clic con il mouse su un elemento nel controllo ListView, verrà generato l'evento Activate, che indica che l'utente ha scelto questo elemento per eseguire alcune azioni. In questo caso, aggioreremo semplicemente la proprietà TextLabel1.Text con il testo dell'elemento selezionato, aggiunto con la parola "attivato", e aggioreremo il controllo ListView chiamando la nostra subroutine ListView1_Click.

```
PUBLIC SUB ListView1_Activate ()  
  
    TextLabel1.Text = ListView1.Item.Text & "attivato".  
  
    sStatus = "attivato."  
  
    ListView1_Click  
  
FIN  
E
```

Se l'utente fa clic con il mouse sul pulsante Button3 (Esci), questa routine è chiamata. Vogliamo uscire in modo pulito, quindi invochiamo il metodo close per il form usando la chiamata ME.Close.

```
PUBLIC SUB Button3_Click  
    () ME.Close  
FINE
```

Utilizzo dello strumento di modifica delle icone di Gambas

A questo punto, tutto il codice viene creato per il nostro programma di esempio ListView. Dobbiamo ancora creare le icone circle.png e square.png. Per farlo useremo il Gambas Icon Editor. Per visualizzare l'editor delle icone, andremo alla finestra del progetto nell'IDE. Nella TreeView del progetto, trova la cartella Dati e fai clic con il pulsante destro del mouse. Seleziona la voce Nuovo e la voce di sottomenu Immagine. Vedrai apparire questa finestra di dialogo:

This product is (C) 2005 by John W. Wittenhouse, all rights are reserved. It is released to the Gamas User Community under the Open Content License (OCL) and may not be distributed under any other terms or conditions.

Una guida per principianti a

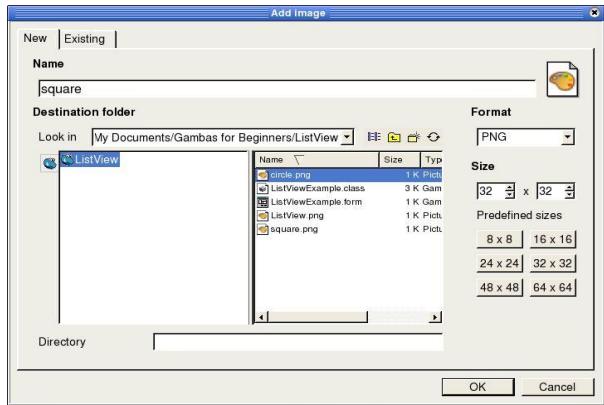


Figura 56 Creazione di una nuova immagine dell'icona in Gamas.

Digita semplicemente "quadrato" nel campo Nome e fai clic su OK. Ora viene visualizzato l'editor delle icone e puoi creare un'icona. Utilizzare l'opzione rettangolo dalla casella degli strumenti dell'Editor delle icone e creare un quadrato blu e salvarlo con l'icona a forma di disco.

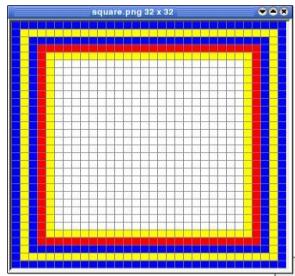


Figura 57 La nostra immagine icona quadrata.

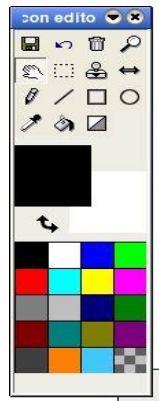


Figura 58
Casella degli strumenti dell'editor delle icone.

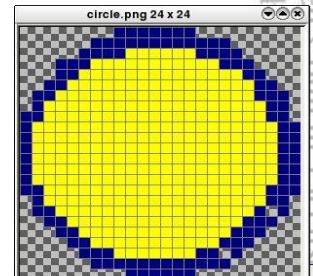


Figura 59 L'immagine dell'icona del nostro cerchio.

Ripeti questo processo, creando un'altra icona denominata "circle.png" e salva anche quella. Questo è tutto! Il tuo programma è pronto per essere eseguito. Provalo e guarda come funziona il controllo ListView.

Il controllo TreeView

Una guida per principianti a

Il controllo TreeView funziona in modo quasi identico al controllo ListView. La differenza principale è che TreeView supporta i "figli" annidati e ti permette di passare da un livello interno (figlio) verso l'esterno al genitore del figlio, fino in fondo

Una guida per principianti a

in cima all'albero. Questo controllo aggiunge alcuni metodi specificamente allo scopo di gestire l'attraversamento degli alberi genitore / figlio.

TreeView, come tutti gli altri controlli, eredita i suoi attributi dalla classe Control. Questo controllo implementa una visualizzazione ad albero di elementi di testo selezionabili con icone. Gli elementi della visualizzazione albero sono indicizzati da una chiave. Visualizzano una stringa e un'icona per ogni elemento. Questo controllo ha un cursore interno utilizzato per accedere ai suoi elementi. Utilizzare i metodi Move (Move, MoveAbove, MoveBack, MoveBelow, MoveChild, MoveCurrent, MoveFirst, MoveLast, MoveNext, MoveParent, MovePrevious, MoveTo) per spostare il cursore interno e la proprietà Item per ottenere l'elemento a cui punta. Questa classe è creabile. La sintassi standard del linguaggio Gambas è:

```
DIM hTreeView AS TreeView  
  
hTreeView = NUOVO TreeView (Parent AS Container)
```

Il codice precedente crea un nuovo controllo TreeView. Questa classe si comporta come un array di sola lettura:

```
DIM hTreeView AS TreeView  
  
DIM hTreeViewItem AS .TreeViewItem  
  
hTreeViewItem = hTreeView [Key AS String]
```

La riga di codice precedente restituirà un elemento TreeView dalla sua chiave. Il cursore interno viene spostato sull'elemento corrente. Ora, usiamo un altro programma di esempio dall'IDE di Gambas. Avvia Gambas e seleziona Esempi | Di base | Programma di esempio TreeView. Quando l'IDE si apre, dovresti vedere qualcosa di simile alla Figura 60 nella pagina seguente.

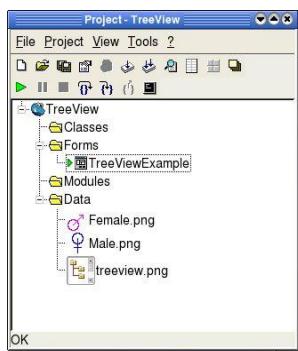


Figura 60 La finestra del progetto TreeView.

This document is © 2010 by John W. Rittinghouse, all rights reserved. It is released to the public domain under a Creative Commons Attribution-NonCommercial-ShareAlike license. It may not be distributed under any other conditions without the express written permission of the author.

Una guida per principianti a

Successivamente, fare doppio clic sul modulo TreeViewExample e visualizzare il modulo. Fai doppio clic sul modulo per aprire la finestra di modifica del codice e vedrai il seguente elenco, che analizzeremo riga per riga per capire come funziona:

```
"File di classe Gambas
PUBLIC intEventNumber AS Integer
```

La variabile globale intEventNumber viene utilizzata per tenere traccia del numero di eventi che si verificano per il nostro stack di eventi. Ogni volta che elaboriamo un evento, incrementeremo questa variabile. Dichiariamo due variabili Picture locali, picMale e picFemale e le carichiamo in memoria per il programma utilizzando il metodo Picture.Load.

```
PUBLIC SUB Form_Open ()
    DIM picMale COME NUOVA
    Immagine DIM picFemale COME
    NUOVA Immagine

    picFemale.Load
    ("Female.png") picMale.Load
    ("Male.png")

    'Questo popolerà il nostro TreeView con le nostre voci
    iniziali' Nota: manterrò il testo delle voci e la sua chiave
    uguali per mantenerlo 'semplice
```

Successivamente, aggiungeremo i valori iniziali del nostro controllo TreeView. Aggiungeremo quattro elementi al controllo. Ogni chiave dell'elemento sarà uguale al nome dell'elemento. Gli elementi Ted e Sally saranno figli dell'articolo Bill, mentre l'elemento Frank sarà figlio di Sally.

```
TreeView1.Add ("Bill", "Bill", picMale)
TreeView1.Add ("Ted", "Ted", picMale, "Bill")
TreeView1.Add ("Sally", "Sally", picFemale,
"Bill") TreeView1.Add ("Frank", "Frank", picMale,
"Sally")
```

Dopo aver aggiunto gli elementi, sposteremo il cursore sull'ultimo elemento aggiunto utilizzando il metodo TreeView1.MoveCurrent e lo evidenzieremo impostando la proprietà TreeView1.Item.Selected su VERO Infine, impostiamo la proprietà TreeView1.Item.Expanded su TRUE per consentire all'elemento di comprimersi o espandersi quando l'utente fa clic sull'icona più / meno. (NOTA: la documentazione di Gambas Wiki per questa proprietà lo elenca come Expand, non

Una guida per principianti a
Expanded. L'uso di Expand NON funzionerà.)

```
TreeView1.MoveCurrent  
TreeView1.Item.Selected = TRUE
```

Una guida per principianti a

```
TreeView1.Item.Expanded = TRUE  
END
```

Se l'utente fa clic su un elemento nel TreeView, vogliamo registrare l'evento Click nel nostro stack di eventi e aggiornare la visualizzazione dello stack (TextArea1.Text). Quindi incrementiamo il numero dell'evento di uno. Si noti che nell'aggiornamento del valore TextArea1.Text, si prende semplicemente il nuovo evento e si aggiunge tutto il resto nella memoria TextArea1.Text dopo il carattere di avanzamento riga, Chr (10). L'effetto netto di questa chiamata in un inserimento del nuovo evento nella parte superiore del testo visualizzato.

```
PUBLIC SUB TreeView1_Click ()  
    'Questo aggiorna solo il nostro stack di eventi  
    TextArea1.Text = "Evento (" & intEventNumber & "): fai clic su" & Chr  
(10) & TextArea1.Text  
    intEventNumber = intEventNumber + 1
```

Successivamente, scopriamo se l'elemento su cui si è fatto clic è un genitore (determinato dal fatto che abbia o meno figli). Se il numero di figli è maggiore di uno, vogliamo che la nostra etichetta (TextLabel1.Text) utilizzi la forma plurale di figlio. Se c'è un solo bambino, la nostra etichetta dirà bambino e se non ce ne sono, vogliamo che dica che l'elemento non ha figli. Usiamo l'istruzione IF THEN / ELSE per effettuare questo controllo:

```
"Questo piccolo segno di spunta aggiorna la nostra etichetta in modo da  
sapere quanti figli ha una voce".  
IF TreeView1.item.Bambini> 1 THEN  
    textlabel1.Text = (TreeView1.Item.Text & "has" &  
TreeView1.Item.Children & "children.")  
    ALTRIMENTI SE TreeView1.item.Children = 0 THEN  
        textlabel1.Text = (TreeView1.Item.Text & "non ha figli.") ELSE  
        textlabel1.Text = (TreeView1.Item.Text & "has 1 child.")  
    END IF  
FINE
```

Se l'utente immette i dati nel controllo Casella di testo e fa clic sul pulsante Inserisci nome (Pulsante1), viene eseguito il seguente clickevent. Per prima cosa dichiara due variabili locali. La variabile picToUse determinerà quale immagine caricare in base a quale RadioButton è stato fatto clic nel momento in cui è stato fatto clic sul pulsante Inserisci nome. Alla variabile stringa, sParent, viene assegnato il valore della chiave dell'elemento corrente. Se il metodo MoveCurrent tenta di spostare il cursore interno sull'elemento corrente e non è presente alcun elemento, restituisce TRUE. In caso contrario, verrà restituito FALSE e assegneremo alla stringa sParent il valore della chiave corrente.

Una guida per principianti a

```
PUBLIC SUB Button1_Click ()  
    DIM picToUse AS NEW Picture
```

This product is (C) 2005 by [www.vR.it](#). All rights are reserved. It is related to the Gambas User Community under the Open Content License (OCL) and may not be distributed under any other terms or conditions. The express written consent of the author is required.

Una guida per principianti a

```
DIM sParent AS String

IF Textbox1.Text <> NULL THEN
    IF RadioButton1.Value THEN
        picToUse.Load
        ("Male.png") ELSE
        picToUse.Load
        ("Female.png") END IF
    'Ottiene l'elemento padre: l'elemento corrente o niente è il
    TreeView' è nullo

    SE NON TreeView1.MoveCurrent ()
        ALLORA sParent = TreeView1.Key
    FINISCI SE
```

Ora, dobbiamo aggiungere la nuova voce con una chiave e un nome di ciò che era nella casella di testo. Lo posizioneremo nel controllo TreeView come figlio della voce attualmente selezionata. Nota che i nomi delle chiavi devono essere univoci o il programma si bloccherà. Potremmo usare il metodo Exist per scoprire se esiste una chiave con il nome Textbox1.Text prima di effettuare la chiamata al metodo Add, ma qui non è stato fatto. Se lo facessimo, il codice eseguirà un controllo simile a questo:

```
IF Exist (Textbox1.Text) <> TRUE THEN TreeView1.Add (Textbox1.Text,
    Textbox1.Text, picToUse, sParent)
FINISCI SE
```

Tuttavia, il codice nel programma forza semplicemente il metodo Add a prendere Textbox1.Text e inserirlo nell'elenco:

```
TreeView1.Add (Textbox1.Text, Textbox1.Text, picToUse, sParent)
TextBox1.Text = "" 'Questo svuota la casella di testo
```

La prossima riga aggiornerà la nostra etichetta e rifletterà il nuovo numero di bambini:

```
TreeView1_Click
```

Questa chiamata a GuaranteVisible assicurerà che l'elemento appena aggiunto all'elenco si trovi nell'area visibile del controllo. Se necessario, il controllo scorrerà in modo che l'elemento sia visibile.

```
    TreeView1.Item.EnsureVisible
END IF
FINE
```

Se l'utente desidera rimuovere un nome da TreeView facendo clic sul pulsante Rimuovi nome (Button2), viene eseguita questa subroutine:

Una guida per principianti a

```
PUBLIC SUB Button2_Click ()
```

Prima di tutto, dobbiamo allineare il cursore alla nostra selezione corrente e assicurarci che l'elemento corrente non sia un valore vuoto o nullo. Se il metodo MoveCurrent restituisce un valore true, l'elenco era vuoto e abbiamo finito. Il codice richiama la chiamata RETURN e torniamo al processo chiamante. Altrimenti, qualunque sia l'elemento corrente verrà rimosso chiamando il metodo Remove nella seconda riga, di seguito:

```
SE TreeView1.MoveCurrent () ALLORA
RITORNA 'Consente di rimuovere
l'elemento del cursore corrente

TreeView1.Remove (TreeView1.Item.Text)
```

Ora dobbiamo spostare il cursore sull'elemento corrente (poiché ora stiamo puntando su un elemento eliminato). Prima di farlo, dobbiamo controllare la proprietà count per assicurarci di non aver eliminato l'ultimo elemento nell'elenco. Se lo abbiamo fatto, ovviamente non eseguiamo questa parte del codice. L'istruzione IF verifica un conteggio maggiore di zero e, se TRUE, si sposta sull'elemento corrente, lo seleziona in modo che il cursore sia evidenziato e aggiorna il controllo con una chiamata alla subroutine TreeView1_Click.

```
SE TreeView1.Count > 0
ALLORA
TreeView1.MoveCurrent

"Questo seleziona o" evidenzia "l'elemento corrente
TreeView1.Item.Selected = TRUE

'Questo aggiornerà la nostra etichetta e rifletterà il nuovo
numero di bambini TreeView1_Click
FINISCI SE
FIN
E
```

Se l'utente fa clic sull'icona meno nel controllo TreeView, imposta un file Evento di compressione. Viene chiamata la routine seguente. Aggiorna semplicemente lo stack degli eventi, come descritto in precedenza, e incrementa il contatore degli eventi, intEventNumber, di uno.

```
PUBLIC SUB TreeView1_Collapse ()
'Questo aggiorna solo il nostro
stack di eventi
TextArea1.Text = "Event (" & intEventNumber & "): Collapse" & Chr
```

Una guida per principianti a

```
(10) & TextArea1.Text  
    intEventNumber = intEventNumber + 1  
END
```

Una guida per principianti a

Se l'utente fa doppio clic su un elemento nel controllo TreeView, imposta un evento Dbl_Click. Fondamentalmente è lo stesso di un evento Activate. Viene chiamata la routine seguente. Aggiorna semplicemente lo stack degli eventi, come descritto in precedenza, e incrementa il contatore degli eventi, intEventNumber, di uno.

```
PUBLIC SUB TreeView1_DblClick ()
    'Questo aggiorna solo il nostro
    stack di eventi
    TextArea1.Text = "Event (" & intEventNumber & "): Doppio clic" & Chr
(10) & TextArea1.Text
    intEventNumber = intEventNumber + 1
END

PUBLIC SUB TreeView1_Select ()
    'Questo aggiorna solo il nostro stack di eventi
    TextArea1.Text = "Event (" & intEventNumber & "): Seleziona" & Chr (10)
& TextArea1.Text
    intEventNumber = intEventNumber + 1
END
```

La prossima routine è codice morto. Non viene mai eseguito perché la subroutine Button2_Click si occupa della rimozione di un elemento. Per usarlo in modo efficace, dovrebbe essere chiamato dalla subroutine Button2_Click appena prima della riga che chiama la subroutine TreeView1_Click. In tal caso, lo stack degli eventi verrà aggiornato correttamente, prima con gli eventi di eliminazione e poi con i clic.

```
PUBLIC SUB TreeView1_Delete ()
    'Questo aggiorna solo il nostro stack di eventi
    TextArea1.Text = "Evento (" & intEventNumber & "): Elimina" & Chr (10)
& TextArea1.Text
    intEventNumber = intEventNumber + 1
END
```

Se l'utente fa clic sull'icona più nel controllo TreeView, imposta un evento di espansione. Viene chiamata la routine seguente. Aggiorna semplicemente lo stack degli eventi, come descritto in precedenza, e incrementa il contatore degli eventi, intEventNumber, di uno.

```
PUBLIC SUB TreeView1_Expand ()
    'Questo aggiorna solo il nostro stack di eventi
    TextArea1.Text = "Event (" & intEventNumber & "): Expand" & Chr (10) &
TextArea1.Text
    intEventNumber = intEventNumber + 1
END
```

Una guida per principianti a

Pulsante PUBLIC SUB 3_Click ()

Una guida per principianti a

```
TextArea1.Text = ""  
'IntEventNumber = 0  
FINE
```

L'Aiuto | La voce di menu Informazioni genera un evento clic che chiama la subroutine successiva. Dà semplicemente credito all'autore di questo esempio, C. Packard.

```
PUBLIC SUB About_Click ()  
    Message.Info ("Esempio TreeView scritto da C. Packard." & Chr (10) &  
    "Aug 2004")  
FINE
```

Se l'utente fa doppio clic su un elemento nel controllo TreeView, attiva un evento Activate. È fondamentalmente lo stesso di un evento DblClick. Viene chiamata la routine seguente. Aggiorna semplicemente lo stack degli eventi, come descritto in precedenza, e incrementa il contatore degli eventi, intEventNumber, di uno.

```
PUBLIC SUB TreeView1_Activate ()  
    'Questo aggiorna solo il nostro  
    stack di eventi  
    TextArea1.Text = "Event (" & intEventNumber & "): Activate" & Chr  
(10) & TextArea1.Text  
    intEventNumber = intEventNumber + 1  
END
```

Anche questa routine successiva è codice morto. Non viene mai eseguito perché il nulla crea un evento per arrivare qui. Per utilizzarlo in modo efficace, una voce di menu Rename dovrebbe essere aggiunta al menu Help e dovrebbe essere chiamata dall'evento MenuItem.Click. Se questo viene fatto, l'evento verrebbe generato e potresti scrivere un po 'di codice per ottenere il nuovo nome dall'utente e cambiarlo. Lo stack degli eventi verrebbe aggiornato correttamente, prima con l'evento rename e poi con l'evento click.

```
PUBLIC SUB TreeView1_Rename ()  
    'Questo aggiorna solo il nostro stack di eventi  
    TextArea1.Text = "Event (" & intEventNumber & "): Rename" & Chr (10) &  
    TextArea1.Text  
    intEventNumber = intEventNumber + 1  
END
```

Questo è tutto ciò che c'è nel programma TreeView. Puoi giocare con il codice e implementare i suggerimenti qui forniti. Una volta che sarai in grado di capire come utilizzare questi controlli in modo efficace, le tue interfacce Gambas diventeranno più sofisticate. Successivamente, daremo un'occhiata agli ultimi due

Una guida per principianti a
controlli di visualizzazione avanzati, i controlli GridView e ColumnView.

Una guida per principianti a

Il controllo GridView

Il controllo GridView, come tutti gli altri controlli nella casella degli strumenti, eredita i suoi attributi dalla classe Control. GridView implementa un controllo che visualizza i dati in una griglia. Questa classe è creabile. La sintassi del linguaggio Gambas standard per GridView è:

```
DIM hGridView AS GridView  
hGridView = NEW GridView (Parent AS Container)
```

Il codice precedente creerà un nuovo GridView. Questa classe si comporta come un array di sola lettura. Per recuperare il contenuto di una cella nella griglia, usa questo codice per dichiarare una variabile GridCell in grado di leggere la matrice GridView (GridCell è una classe virtuale):

```
DIM hGridView AS GridView  
DIM hGridCell AS .GridCell  
  
hGridCell = hGridView [Row AS Integer, Column AS Integer]
```

La riga sopra restituisce una cella dalla sua riga e dalla sua colonna. Il nostro programma di esempio per questo controllo creerà una griglia di quattro righe di tre colonne. Popoleremo le prime tre righe con testo e l'ultima riga con immagini. Prenderemo l'ultima riga di testo e dimostreremo come funziona la proprietà di allineamento per i dati di testo. Il nostro programma avrà tre pulsanti, Esci, Cancella e Ripristina.

Il pulsante Esci chiuderà semplicemente il programma. Clear richiamerà il metodo GridView.Clear per cancellare il contenuto della griglia e visualizzare le celle della griglia vuote. Il pulsante Ripristina ripopolerà il contenuto della griglia con i dati con cui il programma ha iniziato (cioè, i dati popolati quando è stato chiamato il metodo di costruzione del modulo). Il nostro programma avrà questo aspetto:

Una guida per principianti a

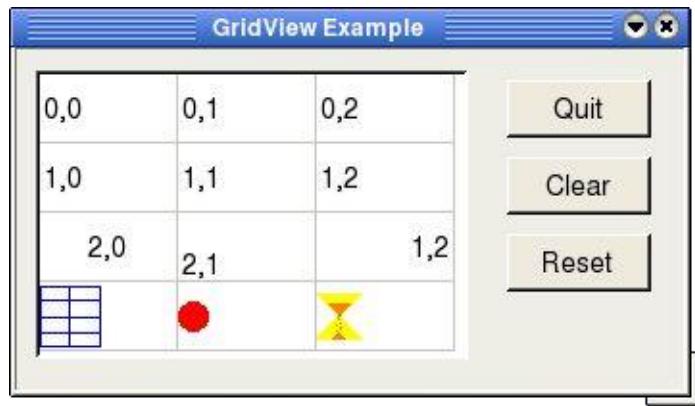


Figura 61 Come sarà il nostro GridView.

Per iniziare, avvia Gambas e seleziona l'opzione Nuovo progetto, scegliendo di creare un progetto di interfaccia utente grafica. Rendi il progetto traducibile e i controlli del modulo pubblici selezionando le due caselle nella procedura guidata. Assegna un nome al progetto GridView Example e inseriscilo nella directory GridView. Quando apparirà l'IDE, dovremo andare alla finestra del progetto e far apparire l'Editor ICON. Creeremo tre icone, denominate GridPic, GridPic1 e GridPic2, come mostrato sopra. Cerca di avvicinarli ragionevolmente a ciò che vedi nell'immagine sopra, se puoi. Crea un nuovo modulo che sia un modulo di classe di avvio e inserisci il controllo GridView, denominato GridView1 nel modulo come mostrato sopra. Quindi aggiungere i tre pulsanti, denominati Button1, Button2 e Button3 al form. Fai in modo che le proprietà Button.Text riflettano ciò che è mostrato nell'immagine sopra.

Ora, abbiamo il modulo progettato e sembra quello che ci aspettiamo. Iniziamo a programmare. Fare doppio clic sul form, ma non su un controllo, e la subroutine Form_Open verrà visualizzata nella finestra del codice. Imposteremo qui la didascalia per il tuo programma:

```
"File di classe Gambas

PUBLIC SUB Form_Open
()
    Form1.Caption = "Esempio GridView"
END
```

Quando il programma si avvia, se esiste un costruttore, viene eseguito per primo. Il nostro costruttore caricherà le nostre tre icone e popolerà la griglia. Innanzitutto, dichiara tre variabili locali per le immagini, hPic1, hPic2 e hPic3, come mostrato di seguito:

```
PUBLIC SUB _new ()
```

This product is (C) 2005 by Gambas User Community. All rights reserved. It is released to the public domain under the terms of the GNU General Public License (GPL). It may not be distributed in any other way without the express written consent of the author.

Una guida per principianti a

```
DIM hPic1 AS Picture  
DIM hPic2 AS Picture  
DIM hPic3 AS Picture
```

Una guida per principianti a

Ora, dobbiamo istanziare le variabili dell'immagine e caricare le immagini.

```
hPic1 = NUOVA immagine  
hPic1.Load  
("GridPic.png")  
  
hPic2 = NUOVA immagine  
hPic2.Load  
("GridPic2.png")  
  
hPic3 = NUOVA immagine  
hPic3.Load  
("GridPic1.png")
```

La prossima cosa che dobbiamo fare è definire le dimensioni della griglia. La nostra griglia avrà tre colonne e quattro righe, quindi impostiamo le proprietà Colonne e Righe usando la loro classe virtuale gridcolumn e proprietà gridrow, conta per impostare le righe e le colonne come mostrato di seguito:

```
GridView1.Columns.Count = 3  
GridView1.Rows.Count = 4
```

Possiamo anche definire le dimensioni di larghezza e altezza delle colonne e delle righe della griglia. Tieni presente che è possibile impostare valori di altezza diversi per ogni riga e valori di larghezza diversi per ogni colonna. Imposteremo le proprietà W e H per le nostre righe e colonne come mostrato di seguito:

```
GridView1.Columns.Width = 72  
GridView1.Rows.Height = 36
```

Successivamente, popoleremo la griglia con alcuni dati, identificando semplicemente la riga, la posizione della colonna con il testo:

```
GridView1 [0,0] .Text = "0,0"  
GridView1 [0,1] .Text = "0,1"  
GridView1 [0,2] .Text = "0,2"  
  
GridView1 [1,0] .Text = "1,0"  
GridView1 [1,1] .Text = "1,1"  
GridView1 [1,2] .Text = "1,2"
```

Per l'ultima riga di testo, vogliamo dimostrare come utilizzare la proprietà Alignment:

```
GridView1 [2,0] .Alignment = Align.Center  
GridView1 [2,1] .Alignment = Align.BottomLeft  
GridView1 [2,2] .Alignment = Align.TopRight
```

Una guida per principianti a

Dopo aver impostato le proprietà di allineamento per ogni cella nella riga,
aggiungeremo del testo:

This product is (C) 2005 by J.W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open CoCoa License (OCCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
GridView1 [2,0] .Text = "2,0"  
GridView1 [2,1] .Text = "2,1"  
GridView1 [2,2] .Text = "1,2"
```

Infine, aggiungeremo le nostre tre icone alla griglia:

```
GridView1 [3,0] .Picture =  
hPic1 GridView1 [3,1] .Picture  
= hPic2 GridView1 [3,2]  
.Picture = hPic3  
FINE
```

Il costruttore è completo. Ora, fai doppio clic sul pulsante Esci per ottenere la subroutine dell'evento Button1_Click, in cui aggiungeremo la nostra chiamata Me.Close per chiudere il programma:

```
PUBLIC SUB Button1_Click  
() ME.Close  
FINE
```

Il pulsante Cancella è il prossimo. Fare doppio clic sul pulsante Cancella per ottenere la subroutine dell'evento Button2_Click, in cui aggiungeremo il nostro codice per reimpostare la griglia su valori vuoti:

```
PUBLIC SUB Button2_Click  
() GridView1.Clear  
FINE
```

Il modo più semplice per ripopolare i dati è semplicemente invocare nuovamente il costruttore. Aggiungi questo codice in modo che sia quello che faremo.

```
Pulsante PUBLIC SUB 3_Click ()  
_new  
END
```

Il nostro programma è completo. Esegilo e guarda come funziona. Dopo che sei soddisfatto che funziona come pubblicizzato, chiudilo e quindi impariamo a conoscere il controllo ColumnView.

Una guida per principianti a

Il controllo ColumnView

Per il nostro prossimo esempio, creeremo un programma che visualizza i dati in un ColumnView, come mostrato di seguito:

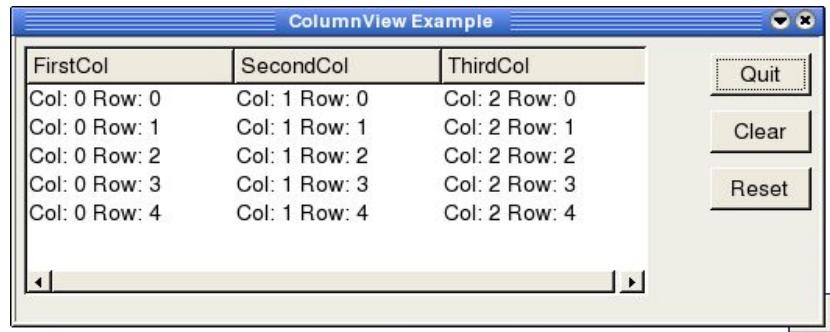


Figura 62 Il nostro esempio ColumnView.

Il nostro semplice esempio creerà tre colonne e cinque righe. Popoleremo i dati della colonna con la riga, l'id della colonna dell'elemento particolare. Per fare ciò, dovremo impostare la larghezza di ciascuna colonna da calcolare dinamicamente in base alla larghezza del controllo e al numero di colonne specificate. Lo faremo nel costruttore. All'avvio del programma, tutto ciò che faremo sulla chiamata Form_Open è impostare la didascalia della finestra, come mostrato nella subroutine Form_Open ():

```
"File di classe Gambas
PUBLIC SUB Form_Open()
    Form1.Caption = "Esempio ColumnView" END
```

Il metodo del costruttore è dove viene svolto tutto il lavoro per popolare il controllo. Dobbiamo dichiarare tre variabili intere locali, iwidth per contenere il valore della larghezza della colonna calcolata, irowcounter e icolcounter, che vengono utilizzati per le nostre strutture di loop.

```
PUBLIC SUB _new ()
'dichiarano i nostri
vars locali DIM iwidth
AS Integer
DIM irowcounter AS Integer
DIM icolcounter AS Integer

'imposta il numero di colonne
su 3
ColumnView1.Columns.Count = 3
'calcola la larghezza della colonna in base al numero di colonne
```

This product is (C) 2005 by John W. R. Rogers and rights are reserved. This is released to the
Gambas User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without express written consent of the author.

Una guida per principianti a

```
impostate iwidth = ColumnView1.Width / ColumnView1.Columns.Count
```

Una guida per principianti a

```
'per la prima colonna, imposta la larghezza della colonna e aggiungi un  
titolo di colonna ColumnView1.Columns [0] .Width = iwidth  
ColumnView1.Columns [0] .Text = "FirstCol"  
'per la seconda colonna, imposta la larghezza della colonna e aggiungi un  
titolo di colonna ColumnView1.Columns [1] .Width = iwidth  
ColumnView1.Columns [1] .Text = "SecondCol"  
'per la terza colonna, impostare la larghezza della colonna e aggiungere  
un titolo di colonna ColumnView1.Columns [2] .Width = iwidth  
ColumnView1.Columns [2] .Text = "ThirdCol"
```

Ora, dobbiamo impostare i loop esterno (riga) e interno (colonna). Fondamentalmente, eseguiremo un ciclo attraverso ogni riga nel controllo e, ad ogni riga, eseguiremo il ciclo attraverso ogni colonna, popolando gli elementi dell'array ColumnView1 [riga] [colonna] con una stringa di testo costruita dinamicamente che rappresenterà la posizione corrente all'interno il ciclo interno ed esterno.

```
PER irowcounter = da 0 a 4 'per tutte e cinque le nostre righe  
    'chiama il metodo Add con [row] [col] per aggiungerlo al controllo  
    ColumnView1.ColumnView1.Add (irowcounter, icolcounter) 'inizializza il  
    ciclo esterno (riga) FOR icolcounter = 0 TO  
    ColumnView1.Columns.Count - 1  
        'aggiunge del testo alla voce [riga] [col] ColumnView1  
        [irowcounter] [icolcounter] = "Col:" & icolcounter & "  
Riga: "& irowcounter  
    NEXT 'colonna  
    IL PROSSIMO 'riga  
FINE 'il nostro costruttore
```

Il pulsante Esci qui funziona proprio come nell'esempio precedente. Per ottenere la subroutine dell'evento Button1_Click, fare doppio clic e aggiungere la chiamata Me.Close per chiudere il programma:

```
PUBLIC SUB Button1_Click  
    () ME.Close  
FINE
```

Il pulsante Cancella è il prossimo. Fare doppio clic sul pulsante Cancella per ottenere la subroutine dell'evento Button2_Click, in cui aggiungeremo il nostro codice per reimpostare la griglia su valori vuoti:

```
PUBLIC SUB Button2_Click  
    () ColumnView1.Clear  
FINE
```

Per il nostro pulsante Ripristina, il modo più semplice per ripopolare i dati è semplicemente invocare nuovamente il costruttore. Aggiungi questo codice in modo

This product is (C) 2003 Jim W. Rittinghouse, all rights reserved. It is released to the
Gambari User Community under an Open Content License (OGL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
che sia quello che faremo.

Una guida per principianti a

```
Pulsante PUBLIC SUB 3_Click ()  
    _new  
END
```

Questo è tutto ciò che serve per creare e popolare il controllo ColumnView.

Successivamente, daremo un'occhiata a come utilizzare i vari controlli di layout in Gambas.

Controlli del layout: HBox, VBox, HPanel e Vpanel

Tutti i controlli in questo gruppo sono simili in quanto cercano di organizzare i controlli che contengono automaticamente. Lo fanno utilizzando la proprietà .Arrangement che si basa su costanti definite nella classe Arrange. Le costanti utilizzate dalla proprietà Arrangement di questi controlli contenitore includono quanto segue: Fill, Horizontal, LeftRight, None, TopBottom e Vertical. Se non specifichi un valore di proprietà .Arrangement, il controllo viene impostato per impostazione predefinita su qualsiasi cosa sia progettato per fare, ovvero HBox disporrà i controlli orizzontalmente, VBox verticalmente, ecc.

HBox e VBox

Entrambe queste classi sono contenitori che dispongono i loro figli orizzontalmente o verticalmente. Come tutti gli altri controlli ToolBox, ereditano i propri attributi dalla classe Container. Entrambi questi controlli agiscono come un pannello senza un bordo la cui proprietà Arrangement sarebbe impostata su Arrange.Horizontal. Queste classi sono creabili. La sintassi standard del linguaggio Gambas per creare un nuovo Hbox è:

```
DIM hHBox AS HBox  
DIM hVBox AS VBox  
hHBox = NEW HBox (Parent AS Container)  
hVBox = NEW VBox (Parent AS Container)
```

HPanel e Vpanel

Entrambe queste classi sono contenitori che organizzano i loro figli dall'alto verso il basso, quindi da sinistra a destra per l'Hpanel e da sinistra a destra, dall'alto in basso per il VPanel. Sono entrambi come un normale controllo Panel senza bordo ma la cui proprietà Arrangement sarebbe impostata su Arrange.TopBottom (dove un contenitore impila i suoi figli dall'alto verso il basso; se i controlli figli non possono adattarsi verticalmente, una nuova colonna di controlli è iniziato subito dopo la colonna precedente) o Arrange.LeftRight (dove un contenitore impila i suoi

Una guida per principianti a
figli da sinistra a destra; se i controlli figli non possono adattarsi orizzontalmente,
viene avviata una nuova riga di controlli sotto il

Una guida per principianti a

riga precedente). Entrambi questi tipi di controlli ereditano i propri attributi dalla classe Container. Questa classe è creabile. La sintassi standard del linguaggio Gambas è:

```
DIM hHPanel AS HPanel  
DIM hVPanel AS VPanel  
hHPanel = NUOVO HPanel (Parent AS Container)  
hVPanel = NEW VPanel (Parent AS Container)
```

Creiamo un semplice programma che dimostrerà ciascuno di questi controlli di layout e mostrerà come possono essere utilizzati in un'applicazione. Per questa applicazione, avvia Gambas e crea un programma di interfaccia utente grafica. Denominalo Layout e crea un modulo di classe di avvio, Form1. Per questa applicazione, dovremo utilizzare alcune icone. La maggior parte delle distribuzioni basate su Linux hanno file di icone archiviati nella cartella denominata:

file: / usr / share / icons / defaultkde / 32x32 / actions

Il tuo sistema potrebbe essere diverso. Indipendentemente da dove le trovi, scegli sette icone e copiale nella cartella Layout. Per i nostri scopi, qualsiasi sette andrà bene, ma se riesci a trovare le icone annulla, ripeti, aiuto, configura, colora, stampante e tipo di uscita sarà più facile. Ecco le icone che ho scelto di utilizzare per questo progetto:



Figura 63 Icône del progetto di layout.

Il nostro programma è abbastanza semplice. Creeremo un HBox, un VBox, un HPanel e un VPanel e inseriremo alcuni semplici controlli in ogni contenitore. Aggioreremo semplicemente un'etichetta di testo per mostrare quando è stato fatto clic su qualcosa. Ecco come apparirà il nostro modulo in modalità progettazione:

Una guida per principianti a

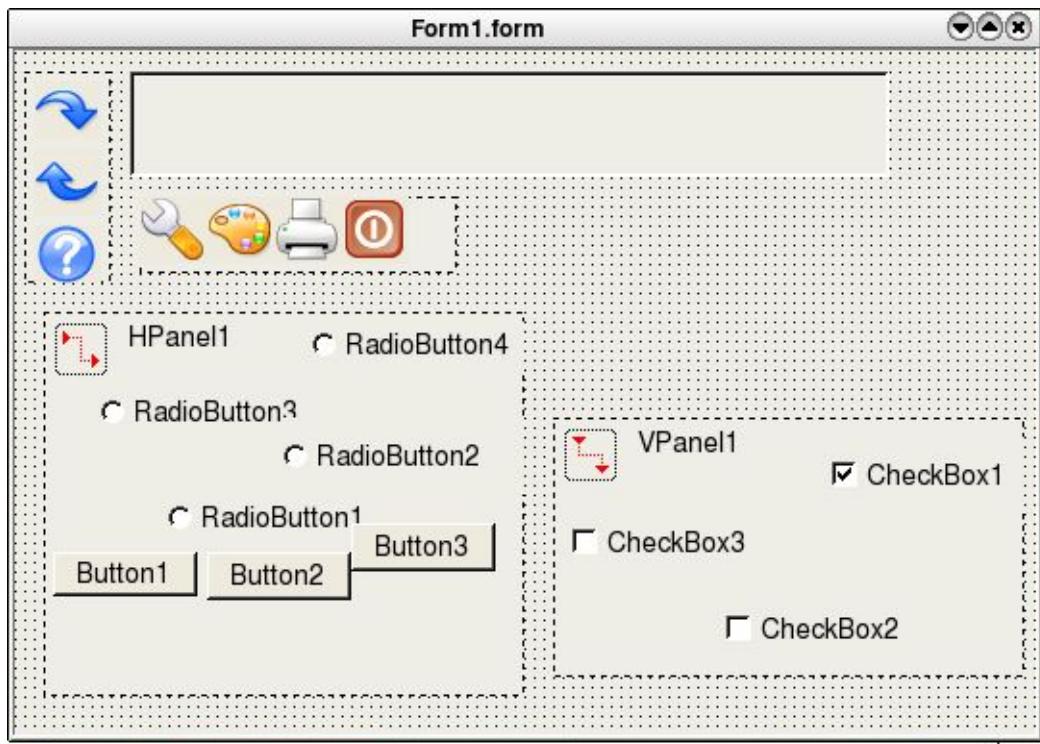


Figura 64 Modalità di progettazione Form1 che mostra il layout dei nostri controlli.

Dall'angolo in alto a sinistra della figura sopra, iniziamo con un VBox e un TLabel. Sotto il TLabel è l'HBox. Quando crei questi controlli, Gambas suggerirà i nomi predefiniti per ogni controllo. Prendi i nomi predefiniti per ogni controllo perché il nostro programma utilizza questi valori predefiniti. Posiziona tre ToolButtons nel VBox e quattro ToolButtons nell'HBox. Successivamente, imposta le proprietà dell'immagine di ogni ToolButton sulle icone che hai scelto di utilizzare (si spera uguali a quelle sopra) per il nostro progetto. Una volta visualizzate le icone, dobbiamo iniziare con il primo ToolButton nel VBox e fare doppio clic per visualizzare la finestra del codice. Dovrai farlo per ciascuna delle sette icone e aggiungere il codice come segue:

```
"File di classe Gambas
PUBLIC SUB ToolButton1_Click ()
    TLabel1.Text = "Vert Undo cliccato."
FINE

PUBLIC SUB ToolButton2_Click ()
    TLabel1.Text = "Vert Redo cliccato."
FINE

PUBLIC SUB ToolButton3_Click ()
    TLabel1.Text = "Vert Help cliccato."
```

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. This software is released to the public domain under the terms of the MIT License. It is not be distributed or modified without the explicit written consent of the author.

This product is (C) 2005 by John W. Ritter. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without express written consent of the author.

Una guida per principianti a

FINE

```
PUBLIC SUB ToolButton4_Click ()  
    TextLabel1.Text = "Horiz Configure Btn ha fatto  
clic." FINE  
  
PUBLIC SUB ToolButton5_Click ()  
    TextLabel1.Text = "Pulsante Horiz Color  
cliccato." FINE  
  
PUBLIC SUB ToolButton6_Click ()  
    TextLabel1.Text = "Horiz Printer Btn ha fatto  
clic." FINE  
  
PUBLIC SUB ToolButton7_Click ()  
    TextLabel1.Text = "Pulsante di uscita Horiz  
cliccato." ATTENDERE 0.5  
    ME.Close  
END
```

Il prossimo contenitore di controlli che creeremo è HPanel, come mostrato nella figura sopra. Aggiungeremo quattro RadioButtons e tre pulsanti normali a questo controllo contenitore. Prova a disporre i pulsanti come mostrato nella figura. Vedrai perché questo è importante una volta eseguito il programma e vedrai come HPanel riorganizza i controlli. Ancora una volta, fai doppio clic su ciascun pulsante di opzione e aggiungi il seguente codice:

```
PUBLIC SUB RadioButton1_Click ()  
    TextLabel1.Text = "RadioBtn 1 cliccato."  
FINE  
  
PUBLIC SUB RadioButton2_Click ()  
    TextLabel1.Text = "RadioBtn 2 cliccato."  
FINE  
  
PUBLIC SUB RadioButton3_Click ()  
    TextLabel1.Text = "RadioButton 3  
cliccato."  
FINE  
  
PUBLIC SUB RadioButton4_Click ()  
    TextLabel1.Text = "RadioButton 4  
cliccato."  
FINE
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gamasas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB Button1_Click ()
    TextLabel1.Text = "Pulsante 1
                      cliccato."
FINE

PUBLIC SUB Button2_Click ()
    TextLabel1.Text = "Pulsante 2
                      cliccato."
FINE

PUBLIC SUB Button3_Click ()
    TextLabel1.Text = "Pulsante 3
                      cliccato."
FINE

L'ultima cosa che faremo è creare il VPanel con tre CheckBox. Ricorda di disporre i CheckBox nel tuo modulo come quelli nell'immagine sopra per vedere come il controllo li riorganizza automaticamente. Aggiungi questo codice dopo aver posizionato i CheckBox nel VPanel:
```

```
PUBLIC SUB CheckBox1_Click ()
    IF CheckBox1.Value = TRUE THEN
        TextLabel1.Text = "Casella di controllo
                          1 selezionata."
    ALTRO
        TextLabel1.Text = "Casella di controllo 1
                          deselezionata." FINISCI SE
FINE

PUBLIC SUB CheckBox2_Click ()
    IF CheckBox2.Value = TRUE THEN
        TextLabel1.Text = "Casella di controllo
                          2 selezionata."
    ALTRO
        TextLabel1.Text = "Casella di controllo 2
                          deselezionata." FINISCI SE
FINE

PUBLIC SUB CheckBox3_Click ()
    IF CheckBox3.Value = TRUE THEN
        TextLabel1.Text = "Casella di controllo
                          3 selezionata."
    ALTRO
        TextLabel1.Text = "Casella di controllo 3
                          deselezionata." FINISCI SE
FINE
```

Una guida per principianti a

Ora salva il tuo lavoro ed esegui questo semplice programma. Dovresti vedere qualcosa di simile a questo:

Una guida per principianti a



Figura 65 Programma di layout all'avvio.

Si noti come i RadioButtons nell'HPanel si siano allineati verticalmente da RadioButton4 a RadioButton1 e, poiché c'era spazio, ha posizionato i controlli dei pulsanti a destra e ha continuato l'allineamento dall'alto verso il basso, da sinistra a destra. Per il VPanel, guarda come i CheckBox si sono allineati. Questi tipi di controlli sono imprevedibili nel modo in cui impagineranno i loro figli. È meglio utilizzarli nel codice in modo dinamico quando è necessario creare moduli al volo e non si ha la possibilità di disporli in un pannello fisso in modalità progettazione.

I controlli VBox e HBox contengono i ToolButtons e ti consentono di creare piccole barre degli strumenti che funzionano bene con i controlli guidati iconici, come quelli raffigurati con le nostre icone. Una volta che hai finito di giocare con la nostra piccola applicazione, chiudi il programma e passeremo a conoscere TabStrips.

Il controllo TabStrip

Il controllo TabStrip Gambas implementa un controllo contenitore a schede. Come tutti i controlli, eredita i suoi attributi dalla classe Container. Questa classe è creabile. Questa classe si comporta come un array di sola lettura. La sintassi standard del linguaggio Gambas è:

```
DIM hTabStrip AS TabStrip  
hTabStrip = NEW TabStrip (padre come contenitore)  
  
DIM hTab AS .Tab  
hTab = hTabStrip [Index AS Integer]
```

Invocare il codice sopra riportato restituirà un oggetto tab virtuale dal suo indice. I TabStrip sono utili per organizzare e presentare le informazioni in unità autosufficienti. Le TabStrip ti consentono di ottenere input dagli utenti presentando un'interfaccia semplice e facile da seguire che può letteralmente guiderli attraverso

This product is (C) 2005 Gambas User Community under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
una configurazione o un'impostazione

Una guida per principianti a processi. Ecco un esempio di TabStrip in uso:

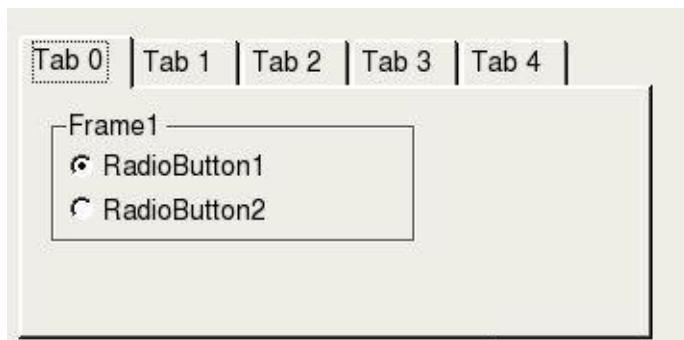


Figura 66 Un controllo TabStrip.

Per dimostrare quanto siano facili i TabStrips da creare, costruiremo una piccola applicazione che implementa l'interfaccia a schede che vedi sopra. La nostra applicazione che ti mostrerà come utilizzare i controlli a schede per presentare le informazioni a un utente e rispondere ad esse nelle tue applicazioni. Avvia Gambas e crea un nuovo programma di interfaccia utente grafica. Assegna un nome alle schede del progetto, fai clic sulla procedura guidata di creazione del progetto e quando arrivi all'IDE di Gambas, crea un nuovo modulo, Form1 che è una classe di avvio. Successivamente, sarà necessario posizionare un controllo TabStrip nel nuovo modulo. Una volta posizionato TabStrip, vedrai solo una scheda visualizzata. Devi andare alla finestra delle proprietà e impostare la proprietà Count su 5 per avere il controllo simile a quello mostrato sopra. Notare che la numerazione delle schede inizia da zero in Gambas. Ciascuna scheda funge da contenitore per i controlli che vengono posizionati su di essa. Dovremo anche aggiungere una TLabel e un pulsante al nostro modulo. Il TLabel verrà utilizzato per mostrare le azioni dell'utente in risposta alla nostra interfaccia a schede. Il pulsante sarà un pulsante Esci che useremo per terminare l'applicazione. Ecco come dovrebbe apparire il tuo modulo quando inizi a progettare l'interfaccia:



Figura 67 Scheda Progetto Form1.form Design.

This product is (C) 2005 by [Johannes Röder](http://www.gambas-project.org). It is released to the Gambas User Community under the terms of the [GNU Lesser General Public License](http://www.gnu.org/licenses/lgpl.html). It may not be distributed under any other terms or conditions.

Una guida per principianti a

Dovrai anche copiare le icone di configurazione, aiuto e uscita dal progetto Layout alla cartella Tabs in modo da poterle usare con questo progetto. Gambas è progettato per contenere tutto il lavoro del progetto dall'IDE alla cartella che crei quando avvii un progetto. Di conseguenza, è necessario uscire dall'IDE per copiare i file da un'altra cartella al progetto corrente (un capriccio di progettazione, forse?). Comunque, andiamo avanti con questo progetto. Per Tab0 aggiungeremo una cornice e inseriremo due RadioButtons in essa, in questo modo:

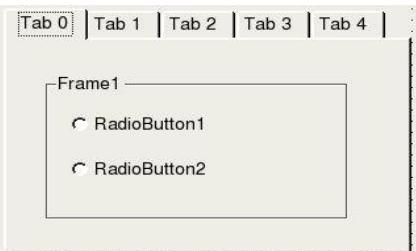


Figura 68 Layout Tab0.

Successivamente, fai doppio clic su ciascun RadioButton e inserisci questo codice:

```
PUBLIC SUB RadioButton1_Click ()  
    TextLabel1.Text = "Hai fatto clic su: <br> <center> <strong>  
RadioButton1" END  
  
PUBLIC SUB RadioButton2_Click ()  
    TextLabel1.Text = "Hai fatto clic su: <br> <center> <strong>  
RadioButton2" END
```

Ecco cosa faremo per Tab1:



Figura 69 Layout Tab1.

Fai doppio clic su ogni CheckBox e inserisci questo codice:

```
PUBLIC SUB CheckBox1_Click ()  
    SE CheckBox1.Value = TRUE THEN  
        TextLabel1.Text = "Hai controllato: <br> <center> <strong> CheckBox1"
```

This document is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

This product is (C) 2005 J. Rittinger. All rights reserved. It is released to the
Gambas User Community under the OpenContent License (OCL) and may not be distributed
under any other terms or conditions without explicit written consent of the author.

Una guida per principianti a

```
ALTRÒ
  TextLabel1.Text = "Hai deselezionato: <br> <center> <strong>
    CheckBox1" ENDIF
FINE

PUBLIC SUB CheckBox2_Click ()
  SE CheckBox2.Value = TRUE THEN
    TextLabel1.Text = "Hai controllato: <br> <center> <strong>
      CheckBox2" ELSE
    TextLabel1.Text = "Hai deselezionato: <br> <center> <strong>
      CheckBox2" ENDIF
FINE
```

Il codice precedente determinerà se l'utente sta selezionando un elemento o deselectandolo guardando prima la proprietà Value. Se è TRUE, la casella è già selezionata e lo indicheremo nell'aggiornamento di TextLabel1.Text. In caso contrario, verrà deselezionato e aggioreremo l'etichetta per affermare questo fatto.

Per Tab2, posizioneremo un controllo Panel sulla scheda e aggiungeremo tre ToolButtons. Non commettere l'errore di posizionare un ToolButton e provare a copiarlo e spostare la copia nel pannello. Non verrà riconosciuto come figlio del pannello di controllo. Apparirà su tutte le schede (credo che questa sia un'altra stranezza di Gambas). Affinché il controllo del pannello accetti i ToolButtons come figli, è necessario fare clic individualmente sul ToolButton nella casella degli strumenti e creare uno come hai fatto con il primo ToolButton. Per ogni ToolButton, assegna alla proprietà picture il nome dell'icona che stiamo usando per quel pulsante, come mostrato qui:



Figura 70 Layout Tab2 ToolButton con icone.

Il prossimo passo è fare doppio clic su ogni ToolButton e aggiungere questo codice:

```
PUBLIC SUB ToolButton1_Click ()
  TextLabel1.Text = "Hai fatto clic su: <br> <centro> <strong> Configura
  icona" END
```

Una guida per principianti a

```
PUBLIC SUB ToolButton2_Click ()  
    TextLabel1.Text = "Hai cliccato: <br> <center> 1 '<strong> icona della  
guida" FINE  
  
PUBLIC SUB ToolButton3_Click ()  
    TextLabel1.Text = "Hai fatto clic su: <br> <center> 1 '<strong> icona  
di uscita" END
```

Ora aggiungeremo un ComboBox a Tab3. Niente di speciale, solo un semplice ComboBox con tre elementi. Seleziona il controllo ComboBox e posizionalo nella scheda come mostrato di seguito:



Figura 71 Layout Tab3 con un ComboBox.

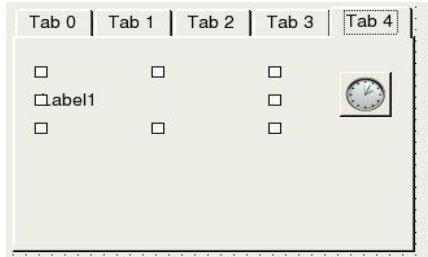
Ricorda, per impostare gli elementi nel ComboBox, devi andare nella finestra Proprietà e selezionare la proprietà List. Apparirà un pulsante con tre punti (a indicare che conduce all'Editor elenco) e sarà necessario fare clic su di esso. Aggiungi i seguenti elementi: "Scegli qualcosa" (da visualizzare come testo predefinito), "Questa cosa", "Quella cosa" e "Qualche altra cosa" all'elenco. Infine, dobbiamo impostare un evento per il ComboBox facendo un singolo clic sul controllo ComboBox, quindi facendo clic con il pulsante destro del mouse. Scegli Eventi e seleziona l'evento Modifica. Ogni volta che il ComboBox cambia, vogliamo visualizzare il testo modificato sulla nostra TextLabel. Ora aggiungi questo codice nell'editor del codice per l'evento ComboBox1_Change ():

```
PUBLIC SUB ComboBox1_Change  
() DIM comboitem AS stringa  
  
comboitem = ComboBox1.Text  
TextLabel1.Text = "Elemento selezionato: <br> <center> <strong>" &  
comboitem END
```

Per la nostra ultima scheda, Tab4, visualizzeremo semplicemente l'ora corrente. Ciò richiederà l'uso del controllo Timer. Avremo bisogno di aggiungere un'etichetta e un controllo Timer a Tab4, come mostrato di seguito:

This product is (C) 2004 by John W. Rittinger. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) which may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a



Affinché il timer funzioni, dobbiamo abilitarlo al primo avvio del modulo. Fai doppio clic sul modulo da qualche parte tranne che su un controllo e vedrai la subroutine Form_Open () apparire nella finestra del codice. Aggiungi questo codice alla routine:

```
PUBLIC SUB Form_Open ()
Form1.Caption = "Giocare con TabStrips"
Timer1.Enabled = TRUE
TextLabel1.Text = ""
END
```

Per utilizzare effettivamente il timer, dobbiamo impostare un evento. Lo facciamo facendo clic una volta sul controllo Timer, quindi facendo clic con il pulsante destro del mouse. Scegli Eventi e seleziona l'evento Timer. Verrai inserito nell'editor del codice e dovrà aggiungere una singola riga di codice:

```
PUBLIC SUB Timer1_Timer
() Label1.Text = Str $
(ora)
FINE
```

Questa riga di codice aggiornerà il campo Label1.Text ogni secondo che viene visualizzato Tab4. In effetti, in questo modo hai creato un orologio sulla scheda. Questo è tutto quello che c'è da fare. Esegui il programma e guarda come funzionano i TabStrip. Successivamente, passiamo alle operazioni sui file.

Una guida per principianti a

Capitolo 9 - Lavorare con i file

In questo capitolo vi presenteremo la gestione dei file Gambas e le operazioni di input / output (file i / o). Gambas ha un insieme completo di funzioni e subroutine per supportare quasi ogni tipo di operazione di i / o file che potresti dover implementare nei tuoi programmi.

- ✓ Accesso
- ✓ Dir
- ✓ Eof
- ✓ Esistere
- ✓ IsDir / Dir?
- ✓ Lof
- ✓ statistica
- ✓ Temp / Temp \$
- ✓ APRIRE e CHIUDERE
- ✓ INPUT, LINE INPUT e PRINT
- ✓ LEGGI, CERCA, SCRIVI e RISCIACQUA
- ✓ COPIA, UCCISIONE e RINOMINA
- ✓ MKDIR e RMDIR
- ✓ LINK

Accesso

L'accesso è una funzione utilizzata per determinare se un file è accessibile o meno. La sintassi del linguaggio Gambas è

```
Accessibile = Accesso (Percorso [, Modalità])
```

La chiamata alla funzione Access restituisce TRUE se il file specificato da Path è accessibile. Se il valore di Mode è gb.Read, la funzione restituisce TRUE se il file può essere letto. Questo è il valore predefinito per tutti i file. Se Mode è gb.Write, Access restituisce TRUE se è possibile scrivere nel file. Quando gb.Exec è specificato per Mode, la funzione restituisce TRUE se il file può essere eseguito. I flag precedenti possono essere combinati con l'operatore OR. Per una directory, il flag di esecuzione significa che la directory può essere sfogliata. Per esempio:

```
PRINT Access ("/ home / rabbit", gb.Write OR gb.Exec)
```

sarebbe tornato

Una guida per principianti a

Vero

mentre la dichiarazione

```
PRINT Accesso ("/ root", gb.Write)
```

tornerà

Falso

Dir

La funzione Dir restituisce una matrice di stringhe che contiene i nomi dei file situato nella directory che corrisponde al modello di file specificato. Se non viene specificato alcun modello, viene restituito qualsiasi nome file esistente nella directory. Il modello può contenere gli stessi caratteri generici consentiti per l'operatore LIKE. In altre parole, quei caratteri si trovano nella seguente tabella:

| Carattere generico | Partite |
|--------------------|---|
| * | qualsiasi numero di qualsiasi tipo di carattere. |
| ? | qualsiasi singolo carattere di qualsiasi tipo. |
| [abc] | qualsiasi carattere specificato tra parentesi. |
| [xy] | qualsiasi carattere che si trova all'interno dell'intervallo specificato. |
| [^ xy] | qualsiasi personaggio che non si trova nell'intervallo. |

Il carattere generico speciale \ impedisce che il carattere che lo segue venga interpretato come generico. La sintassi standard del linguaggio Gambas è:

```
Matrice del nome del file = Dir (Directory [, File pattern])
```

Ecco un segmento di codice di esempio per aiutarti

```
'Stampa una directory
SUB PrintDirectory (Directory AS
    String) DIM sFile AS String

    PER OGNI sFile IN Dir (Directory, "*.
        *") PRINT sFile
    FINE
SUCSES
```

This product is (C) 2005 by Jolien Wiersma. All rights are reserved. It is released to the Gambas User Community under the terms of the GNU General Public License (GPL) and may not be distributed under any other terms or conditions. The express written consent of the author is required.

Una guida per principianti a
SIVA

Eof

La funzione Eof restituisce un valore booleano di TRUE se siamo alla fine del flusso. La sintassi standard del linguaggio Gambas è:

```
Boolean = Eof (File)
```

Segue un esempio di come viene utilizzato Eof:

```
...
OPEN FileName FOR READ AS #hFile
WHILE NOT Eof (hFile)
    LINE INPUT #hFile, OneLine
    PRINT OneLine
WEND
CHIUDI #hFile
...
```

Esistere

La funzione Exist restituisce TRUE se esiste un file o una directory. L'uso di ~ come identificatore di directory non funziona. Se il percorso specificato non esiste, FALSE è restituito. La sintassi del linguaggio Gambas è:

```
Boolean = Exist (Path)
```

Di seguito è riportato un esempio di utilizzo di Exist per determinare se esiste un file prima di chiamare OPEN per leggere il file:

```
...
DIM sCurrentFileName AS String

sCurrentFileName = "The Raven.txt"
SE esiste (sCurrentFileName)
ALLORA
    ... 'apri il file
ALTRO
    Message.Info ("File" & sCurrentFileName & "non esiste.") ENDIF
...
...
```

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

IsDir / Dir?

La funzione IsDir restituisce TRUE se un percorso punta a una directory. Se il percorso non esiste o non è una directory, questa funzione restituisce FALSE. La sintassi del linguaggio Gambas per IsDir è:

```
Boolean = IsDir (percorso)
```

Alcuni esempi di come utilizzare IsDir forniti nel Wiki della documentazione di Gambas sono:

```
PRINT IsDir ("/ etc /  
password") False  
  
PRINT IsDir (Application.Home & /  
.gambas") True  
  
PRINT IsDir ("/  
windows") False
```

statistica

La funzione Stat restituisce informazioni su un file o una directory. Le informazioni sull'oggetto File restituite includono il tipo, la dimensione, la data / ora dell'ultima modifica, i permessi, ecc. Le seguenti proprietà del file possono essere ottenute utilizzando questa funzione:

| | | | | |
|------------------|---------|-------------|-------------|---------------------|
| .Group | .Hidden | .LastAccess | .LastChange | .LastUpdate |
| .Modalità | .Perm | .SetGID | .SetUID | .Dime nsion e |
| .Appiccic oso | .Tempo | .Genere | .Utente | |

La sintassi standard del linguaggio Gambas per Stat è:

```
Informazioni sul file = Stat (Path)
```

Ecco un esempio di come utilizzare Stat e cosa verrà restituito dalla console:

```
CON Stat ("/ home")  
PRINT .Type = gb.Directory  
PRINT Round (.Size / 1024);  
"K"  
FINISCI CON
```

Una guida per principianti a

La console risponderà con:

Una guida per principianti a

True
4K

Temp / Temp \$

La funzione Temp \$ restituisce un nome file univoco utile per creare file temporanei File. Il nome del file creato si troverà nella directory / tmp. La sintassi del linguaggio Gambas è:

```
Nome file = Temp $ ()
```

Esempio:

```
STAMPA Temp $ ()  
/tmp/gambas.0/12555.1.tmp
```

APRIRE e CHIUDERE

OPEN e CLOSE lavorano in squadra. OPEN apre un file per leggere, scrivere, creare o aggiungere dati. A meno che non venga specificata la parola chiave CREATE, il file deve esistere. Se viene specificata la parola chiave CREATE, il file viene creato o cancellato se esiste già. Se il file viene aperto per la scrittura, il percorso deve essere assoluto perché, per impostazione predefinita, si presume che i percorsi relativi facciano riferimento a file che esistono all'interno del progetto Gambas corrente o della cartella di archivio. Sintassi del linguaggio Gambas per OPENè:

```
OPEN Nome file FOR [READ] [WRITE] [CREATE | APPEND] [DIRECT] [WATCH]  
[BIG | LITTLE] COME # Variabile
```

Se viene specificata la parola chiave APPEND, il puntatore del file viene spostato nel file fine del file subito dopo il il file viene aperto. Se viene specificata la parola chiave DIRECT, l'inputoutput non viene memorizzato nel buffer. Se viene specificata la parola chiave WATCH, il file viene controllato dall'interprete utilizzando le seguenti linee guida condizionali:

- Se è possibile leggere almeno un byte dal file, viene chiamato il gestore di eventi File_Read () .
- Se è possibile scrivere almeno un byte nel file, viene chiamato il gestore di eventi File_Write () .

Se viene specificata la parola chiave BIG o LITTLE, tutte le successive

Una guida per principianti a operazioni READ e WRITE su questo file utilizzeranno la rappresentazione dei dati bigEndian o littleEndian. *Variabile* riceve l'oggetto che rappresenta il flusso aperto. CLOSE è la funzione reciproca e chiude semplicemente un file aperto. La sintassi per l'utilizzo di CLOSE è:

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

CHIUDI # File

Ecco un esempio che apre e chiude un file:

"Gambas Class File

```
sCurrentFileName AS String  
sCurrentFileLength AS Integer  
  
PUBLIC SUB Form_Open ()  
  
    DIM sInputLine AS String  
    DIM hFileIn AS File  
  
    sCurrentFileName = "The Raven.txt"  
    SE esiste (sCurrentFileName)  
    ALLORA  
        APRIRE sCurrentFileName PER LEGGERE COME  
        hFileIn sCurrentFileLength = Lof  
        (hFileIn)  
        MENTRE NON Eof (hFileIn)  
            LINE INPUT #hFileIn, sInputLine 'LINE INPUT è spiegato di seguito  
            TextArea1.Text = TextArea1.Text & Chr (13) & Chr (10) & sInputLine  
        WEND  
        CHIUDI  
        hFileIn ALTRO  
        Message.Info ("File" & sCurrentFileName & "non esiste.") ENDIF  
    FINE
```

INGRESSO DI LINEA

LINE INPUT legge un'intera riga di testo sullo standard ingresso. Può essere utilizzato sulla console o con i file. La sintassi del linguaggio Gambas è:

LINEA INPUT Variabile

oppure, per i file in cui i dati vengono letti dal flusso File:

LINE INPUT # File, variabile

Qui è un esempio che puoi provare a vedere come funziona LINE INPUT:

```
STATICO PUBBLICO SUB Principale ()  
    DIM hFile AS File
```

Una guida per principianti a

DIM sInputLine AS String

This product is (C) 2004 by John W. Williams, all rights reserved.
Gambas User Community under an open Content License (CC-BY-SA). It may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
DIM lineCtr AS Integer

APRIRE "The Raven.txt" PER LEGGERE COME
#hFile MENTRE NON Eof (hFile)
    LINE INPUT #hFile, sInputLine
    PRINT Str (lineCtr) & Chr (9) &
    sInputLine INC lineCtr
WEND
CHIUDI
#hFile END
```

NOTA: non utilizzare la chiamata LINE INPUT per leggere i dati da un file binario perché perderà i caratteri di avanzamento riga. Per leggere dati da file binari, devi invece usare il comando READ:

```
LEGGI #hFile, OneLine, 256
```

dove 256 è il numero massimo di byte che si desidera leggere (preceduto da un trattino) simbolo. Il 256 può essere un numero qualsiasi, ma dovresti limitarlo alla dimensione massima di qualsiasi variabile che il tuo programma utilizzerà per contenere i dati letti.

LEGGI, CERCA, SCRIVI e RISCIACQUA

La funzione READ legge i dati dallo standard output. Tratta il flusso di input come dati binari il cui tipo di dati è specificato dal tipo di variabile in cui sono archiviati i dati durante l'operazione di lettura. La rappresentazione binaria dei dati dovrebbe essere la stessa rappresentazione creata utilizzando l'istruzione WRITE. Se Variabile è una stringa, è possibile specificare una lunghezza che indica il numero di byte da leggere. Se il valore del parametro Length è negativo, i byte di lunghezza vengono letti dalla fine del flusso. Se non viene specificato alcun parametro Length per una stringa, l'intera stringa viene letta dal flusso. Il valore della stringa quindi deve essere stato scritto con l'istruzione WRITE. La sintassi standard del linguaggio Gambas per READ è:

```
READ Variabile [, Lunghezza]
```

Se READ viene utilizzato con file operazioni, la sintassi del linguaggio Gambas è:

```
LEGGI # File, Variabile [, Lunghezza]
```

La funzione WRITE scrive le espressioni nell'output standard utilizzando la loro rappresentazione binaria. Se Expression è una stringa, è possibile specificare una lunghezza che indica il numero di byte da scrivere. Se non viene

Una guida per principianti a
specificata alcuna lunghezza per un valore stringa,l'intero valore della stringa
viene scritto direttamente al flusso. Ecco i Gambas

This product is (C) 2004 John W. Rittinghouse, all rights are reserved.
Gambas User Community under an Open Content License (OCL).
It may not be distributed in any other conditions without the express written consent of the author.

Una guida per principianti a

sintassi del linguaggio per l'input standard i flussi:

WRITE Expression [, Length]

e per scrivere su un file, questa è la convenzione:

WRITE # File, Espressione [, Lunghezza]

La funzione SEEK definisce (o sposta) la posizione del puntatore del flusso, la maggior partespesso in preparazione per la successiva operazione di lettura / scrittura. Se la posizione specificata dal parametro Length è negativa, il puntatore del flusso viene spostato all'indietro Length byte dalla fine del file. Per spostare il puntatore del flusso oltre la fine del file, è necessario utilizzare la funzione Lof (). La sintassi standard del linguaggio Gambas è:

SEEK # File, Posizione

Ecco alcuni esempi di usare SEEK:

```
'Passa all'inizio del file SEEK
#hFile, 0

'Sposta dopo la fine del file
SEEK #hFile, Lof (#hFile)

'Sposta 100 byte prima della fine del file
SEEK #hFile, 100
```

La funzione FLUSH viene chiamata per svuotare (o scaricare) il contenuto di un flusso di dati bufferizzato. Se non viene specificato alcun flusso, ogni flusso aperto viene scaricato. In genere, FLUSH viene chiamato dopo più operazioni di scrittura e prima di chiudere un programma routine per garantire che tutti i dati contenuti in un buffer vengano scritti nel file prima di uscire. Ecco la sintassi del linguaggio Gambas per FLUSH:

FLUSH [# File]

COPY, KILL e RENAME

La funzione COPIA copierà un file di origine in un file di destinazione specificato. Il percorso di destinazione non deve avere lo stesso nome del percorso di origine. Non è possibile copiare le directory in modo ricorsivo con questa funzione. La sintassi del linguaggio Gambas per questa funzione è:

Una guida per principianti a

```
COPY Source_path TO Destination_path
```

Ecco un esempio di come utilizzare COPY:

```
'Salva il file di configurazione gambas
COPY System.Home & / ".gambas" / gambas.conf" TO "/mnt/save/gambas.conf.save"
```

La funzione KILL rimuove un file. La sintassi del linguaggio Gambas è KILL Path e tutto ciò che esiste in Path viene eliminato definitivamente. RINOMINARE semplicemente cambia il nome di un file o di una cartella da Vecchio nome a Nuovo nome. Ecco un esempio:

```
RENAME Oldname AS Newname
```

MKDIR, RMDIR

La funzione MKDIR viene utilizzata per creare (creare) una cartella nel file system e RMDIR viene utilizzata per rimuovere una cartella. Entrambe le funzioni accettano un singolo parametro, aPercorso e nome del file. La sintassi del linguaggio Gambas è:

```
Mkdir "file: / rittingj / My Documents / Gambas /
test" Rmdir "file: / rittingj / My Documents /
Gambas / test"
```

In ordine to dimostrare most di tlui file e io/ O divertimentoazioni descritto in thIn questo capitolo, creeremo un'applicazione che fungerà da semplice editor di testo. L'intento qui non è creare un editor di testo completo, ma mostrarti come utilizzare queste funzioni nel contesto di un'applicazione. L'applicazione che costruiremo in questo capitolo è un po 'più complessa di quella che abbiamo visto finora in quanto creeremo due moduli, un file di input, un file di aiuto e utilizzeremo alcune icone di sistema che verranno copiate dal Cartelle delle icone di sistema della distribuzione del sistema operativo Linux. Ecco come apparirà la nostra applicazione in fase di esecuzione:



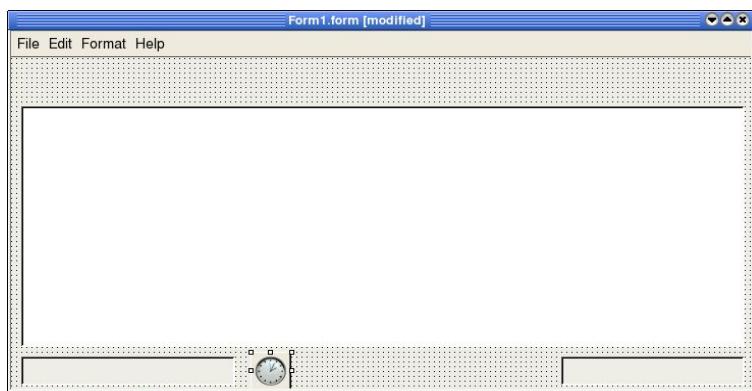
This product is released under the terms of the GNU General Public License version 2, or (at your option) any later version. It is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Una guida per principianti a

Figura 72 Il programma FileOps in fase di esecuzione.

Una guida per principianti a

In questo programma creeremo un menu principale, un'area di testo per visualizzare e modificare il testo e creeremo elementi di stato nella parte inferiore del display. Introdurremo anche l'uso del controllo Timer (mostrato come l'orologio nell'immagine sotto). Questo modulo, denominato Form1.form, avrà questo aspetto in modalità progettazione:



Per iniziare, carica Gambas e crea un nuovo progetto di interfaccia utente grafica. Chiamalo FileOps e quando arrivi all'IDE Gambas, crea un nuovo form, Form1 e make è una classe di avvio. Quando il modulo viene visualizzato sullo schermo, stiamo andando per iniziare creando i menu. È possibile premere CTRL – E per visualizzare l'Editor dei menu. Aggiungere le seguenti voci di menu mostrate nelle tabelle seguenti utilizzando l'editor. Per le icone, dovrà copiare le icone dalla stessa directory che abbiamo specificato nell'ultimo capitolo. La maggior parte delle distribuzioni basate su Linux ha file di icone che di solito sono archiviati nelle cartelle denominate file: /usr/share/icons / defaultkde / 32x32 / actions o file: /usr/share/icons/defaultkde / 32x32 / applications. Il tuo particolare sistema potrebbe essere leggermente diverso. Indipendentemente da dove trovi le icone, scegli le icone appropriate per il nostro programma e copiale nella cartella FileOps. Ecco le 17 icone che ho scelto di utilizzare per questo progetto:





Una guida per principianti a



**Modifi
ca
Configura**



**Format
o
colori**



**Format
ta i
caratt
eri**



Aiuto About Help

**Contenu
ti**

ÅBC

Una guida per principianti a

Se non riesci a trovare la stessa identica icona, scegli semplicemente quella più appropriata e salvalo con il nome file per quell'opzione come mostrato nelle tabelle seguenti. Non è fondamentale che ogni icona sia esattamente la stessa. In effetti, l'icona SelectAll è stata creata utilizzando Gambas Icon Editor.

FileMenu

| Nome variabile | Didascalia | Proprietà icona (nome file) |
|-------------------|-----------------|-----------------------------|
| FileNewItem | "Nuovo" | Immagine ["filenew.png"] |
| FileOpenItem | "Aperto..." | Immagine ["fileopen.png"] |
| FileSaveItem | "Salva..." | Immagine ["filesave.png"] |
| FileSaveAsItem | "Salva come..." | Immagine ["filesaveas.png"] |
| FilePrintItem | "Stampa" | Immagine ["fileprint.png"] |
| FileExitItem Menu | "Uscita" | Immagine ["exit.png"] |

EditMenu

| Nome variabile | Didascalia | Proprietà icona (nome file) |
|-------------------|-------------------|-----------------------------|
| EditSelectAllItem | "Seleziona tutto" | Immagine ["selectall.png"] |
| EditUndoItem | "Disfare" | Immagine ["undo.png"] |
| EditRedoItem | "Rifare" | Immagine ["redo.png"] |
| EditCutItem | "Taglio" | Immagine ["editcut.png"] |
| EditCopyItem | "Copia" | Immagine ["editcopy.png"] |
| EditPasteItem | "Incolla" | Immagine ["editpaste.png"] |
| EditPrefsItem | "Preferenze" | Immagine ["configure.png"] |

FormatMenu

| Nome variabile | Didascalia | Proprietà icona (nome file) |
|-----------------|------------|-----------------------------|
| FormatColorItem | "Colori" | Immagine ["colorize.png"] |
| FormatFontItem | "Font" | Immagine ["charset.png"] |

HelpMenu

| | | |
|------------------|-------------|---------------------------------|
| HelpContentsItem | "Contenuti" | Immagine ["contents.png"] |
| HelpAboutItem | "Di" | Immagine ["buildingblocks.png"] |

Una guida per principianti a

Ecco il codice che useremo per Form1. Andremo attraverso questa riga per riga quindi tutto è completamente spiegato. Iniziamo il nostro programma dichiarando due variabili globali di classe, sCurrentFileName e sCurrentFileLength.

```
"File di classe Gambas
sCurrentFileName AS String
sCurrentFileLength AS Integer
```

All'avvio del programma, viene chiamata questa subroutine. Chiama un costruttore, _new() da inizializzare. Il costruttore è spiegato nella sezione successiva. Quando il nostro programma si avvia, lo faremo caricare automaticamente il nostro file di testo predefinito, chiamato "The Raven.txt". Questo file è stato copiato da Internet ed è possibile inserire qualsiasi file di testo nella cartella del progetto FileOps da utilizzare per questo progetto. Ricorda solo che il nome del file deve essere modificato nel codice seguente se scegli di non utilizzare il nostro nome file predefinito.

```
PUBLIC SUB Form_Open ()
    DIM sInputLine AS String
    DIM hFileIn AS File
```

Assegneremo il nostro nome file alla variabile stringa sCurrentFileName e controlla se il file esiste nella cartella del progetto. Se esiste, il programma aprirà il file e otterrà la lunghezza del file iniziale. Visualizzeremo il nome del file e la dimensione nella proprietà Form1.Caption. Anche se non è consuetudine visualizzare la dimensione del file in questo modo, viene fatto qui semplicemente per dimostrare l'uso della funzione Lof.

```
sCurrentFileName = "The Raven.txt"
SE esiste (sCurrentFileName)
ALLORA
    APRIRE sCurrentFileName PER LEGGERE COME hFileIn
    sCurrentFileLength = Lof (hFileIn) 'per mostrare come ottenere
    la dimensione del file Form1.Caption = "" & sCurrentFileName &
    ", size:" & Str
(sCurrentFileLength) e "byte".
```

Ora, entreremo in una struttura di loop per leggere tutte le righe del file di testo nella nostra proprietà TextArea1.Text. Continueremo a eseguire il loop fino a quando non avremo raggiunto il fine del file. Mentre leggiamo ogni riga in memoria e la assegniamo alla variabile sInputLine, dobbiamo ricordarci di aggiungere il CRLF (un Carriage Return (Chr(13)) e un Line Feed (Chr(10))) alla fine della riga in modo che visualizzare correttamente nel controllo.

Una guida per principianti a

```
MENTRE NON Eof (hFileIn)
    LINE INPUT #hFileIn, sInputLine
    TextArea1.Text = TextArea1.Text & Chr (13) & Chr (10) & sInputLine
WEND
```

Una guida per principianti a

Non appena usciamo dal ciclo, il che significa che abbiamo raggiunto la fine del file, chiuderemo il file.

```
CHIUDI hFileIn
```

Se il nostro controllo per vedere se il file esiste non è riuscito, dovremmo inserire questo ELSE blocco di codice e visualizzare un messaggio che indica che il file non esiste. Successivamente, chiamiamola nostra subroutine FileOpenItem_Click () per fare in modo che l'utente scelga un file da aprire.

```
ALTRÒ
Message.Info ("File" & sCurrentFileName & "non esiste.")
FileOpenItem_Click () 'forza la selezione di un file se non è
predefinito.
ENDIF
END
```

Questo è tutto ciò che c'è nella subroutine Form_Open (). Ecco il costruttore dimodulo:

```
PUBLIC SUB _new ()
    Timer1.Delay = 1000
    Timer1.Enabled = TRUE
FINE
```

Il costruttore imposta semplicemente il ritardo del timer su 1 secondo (1000 ms) e lo imposta la proprietà enabled su true. L'effetto netto qui è che il timer si aggiornera ogni secondo. Implementeremo il controllo del timer più avanti in questo codice.

Rivolgiamo ora la nostra attenzione al file | Nuova opzione di menu. Affinché un nuovo file possa essere creato dall'utente, dobbiamo prima cancellare il testo in TextArea1.Textproprietà. Chiederemo all'utente un nuovo filename e salva il file vuoto su disco per crearlo. Qualsiasi modifica che l'utente apporterà verrà salvata all'uscita o ogni volta che l'utente sceglie di salvare dal menu. L'ultima cosa che facciamo in questa subroutine è impostare la proprietà Form1.Caption sul nome del file appena salvato.

```
PUBLIC SUB FileNewItem_Click
() TextArea1.Text = ""
Dialog.Title = "Inserisci nuovo nome file ..."
Dialog.Filter = ["File di testo (*.txt)", "Tutti i file (*.*)"]
Dialog.SaveFile
Form1.Caption = Dialog.Path
```

END

Una guida per principianti a

La routine successiva viene attivata quando l'utente fa clic su File | Apri voce di menu.

Una guida per principianti a

```
PUBLIC SUB FileOpenItem_Click ()
```

Questa routine richiede due variabili locali. hFileIn è l'handle per l'oggetto FILE che apriremo e sInputLine è una variabile di stringa che useremo per leggere il testo del file nella nostra proprietà TextArea1.Text.

```
DIM hFileIn AS File
DIM sInputLine AS String
```

Appena prima della chiamata alla finestra di dialogo standard, possiamo impostare la finestra didascalia (o titolo) utilizzando la proprietà Dialog.Title. Inoltre, imposteremo un filtro per la finestra di dialogo per cercare solo i file con estensione .txt. Come ultima risorsa, consentiremo all'utente di scegliere di visualizzare tutti i tipi di file selezionando il filtro*.

* Dalla finestra di dialogo.

```
Dialog.Title = "Scegli un file di testo"
Dialog.Filter = ["File di testo (*.txt)", "Tutti i file (*.*)"]
```

Quando viene effettuata la chiamata a Dialog.OpenFile, ricorda che restituirà un valore TRUE se l'utente fa clic sul pulsante Annulla, altrimenti viene restituito FALSE. Verificheremo un valore VERO e restituiremo se questo è ciò che otteniamo. Altrimenti, esso significa che l'utente ha selezionato un file e lo elaboreremo.

```
IF Dialog.OpenFile ()
    THEN RETURN
ALTRO
```

Il nome del file selezionato dall'utente è memorizzato nella proprietà Dialog.Path. Lo assegneremo alla variabile globale sCurrentFileName in modo che altre parti del programma possano accedervi. Imposteremo la proprietà Form1.Caption sul nome del file appena aperto e cancelleremo la visualizzazione corrente di TextArea1 richiamando il metodo Clear.

```
sCurrentFileName = Dialog.Path
Form1.Caption = "" & sCurrentFileName & ""
TextArea1.Clear
```

Successivamente, apriremo il file e leggeremo ogni riga in memoria utilizzando la funzione LINE INPUT e la nostra variabile di stringa sInputLine. Una volta che i dati sono stati archiviati in sInputLine, lo aggiungeremo alla fine di tutto ciò che è attualmente memorizzato nella proprietà TextArea1.Text (concatenandosi effettivamente alla fine del valore di testo

Una guida per principianti a corrente) e aggiungeremo i nostri caratteri CRLF. Una volta raggiunta la fine del file, usciremo dal ciclo e chiuderemo il file.

This product is (c) 2005 by W. Rittinghouse, all rights are reserved.
Gambas User Components under an Open Content License (OCL) are freely distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
APRIRE sCurrentFileName PER LEGGERE COME
hFileIn MENTRE NON Eof (hFileIn)
    LINE INPUT #hFileIn, sInputLine
    TextArea1.Text = TextArea1.Text & Chr (13) & Chr (10) &
sInputLine WEND
CHIUDI
hFileIn ENDIF
FINE
```

Il file | Salva la voce di menu, se cliccata, lo farà invocare questa subroutine. Funziona quasi come il precedente tranne per il fatto che conosciamo già il nome del file contenuto nella variabile globale sCurrentFileName. Quando torniamo dalla chiamata di dialogo standard, aggiorniamo la variabile globale con tutto ciò che è stato restituito, senza preoccuparci se il file ha lo stesso nome o meno. Qualunque cosa sia o sia diventata sarà ciò che verrà salvato.

```
PUBLIC SUB FileSaveItem_Click ()
    Dialog.Title = "Salva file di
    testo"
    Dialog.Filter = ["File di testo (*.txt)", "Tutti i file (*.*)"]
    SE Dialog.SaveFile () ALLORA
        RETURN
    ELSE
        sCurrentFileName = Dialog.Path.File.Save
        (sCurrentFileName, TextArea1.Text)
    ENDIF
END
```

La subroutine successiva che codificheremo è la voce di menu FileSaveAs. Se l'utente vuole salvare il file esistente con un nome file diverso, questo è il file routine per gestire questa opzione. Per questa subroutine, funzionerà in modo quasi identico alla subroutine FileSave ma dovremo controllare se il nome del file restituito dalla finestra di dialogo del file standard è effettivamente diverso dal nome del file corrente. Se è diverso, salveremo il file con il nuovo nome. Se il nome è lo stesso, verrà visualizzato un MessageBox e informeremo l'utente che non è stata intrapresa alcuna azione di salvataggio perché i nomi erano identici.

```
PUBLIC SUB FileSaveAsItemClick
() DIM sTmpName AS stringa
```

Usiamo la variabile stringa locale sTmpName per fare il nostro controllo. Prima di tutto, abbiamo bisogno del percorso completo dell'applicazione e

Una guida per principianti a concatenare il nome del file corrente a quello per ottenere il nome del percorso completo per il nostro file. A tale scopo utilizziamo la classe Application, controllando la proprietà Path per ottenere il nome e il percorso dell'applicazione in esecuzione. Notare l'uso dell'operatore & / che è

This product is CC2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or condition without the express written consent of the author.

Una guida per principianti a

utilizzato specificamente per catenare i nomi dei file.

```
sTmpName = Application.Path & / sCurrentFileName
```

Proprio come abbiamo fatto in precedenza, imposteremo il titolo e le proprietà del filtro, quindi chiama la finestra di dialogo file standard.

```
Dialog.Title = "Salva file di testo con nome ..."
```

```
Dialog.Filter = ["File di testo (*.txt)", "Tutti i file (*.*)"]
```

Se la chiamata alla finestra di dialogo standard restituisce TRUE, l'utente ha annullato la routine e noi torneremo semplicemente al processo chiamante (cioè il nostro evento standard ciclo continuo).

```
IF Dialog.SaveFile ()  
THEN RETURN
```

Se i nomi dei file sono gli stessi, verrà visualizzato il MessageBox e torneremo:

```
ELSE IF sTmpName = Dialog.Path THEN  
    Message.Info ("File non salvato: nome uguale al file corrente."  
    "OK") RETURN
```

Altrimenti, impostiamo la variabile globale sul nome del file restituito dalla finestra di dialogo file standard e chiama il metodo Save, passando il nome del file e il nostro testo come parametri. Aggiorniamo la didascalia della finestra come ultima cosa da fare prima di tornare al processo di chiamata.

```
ALTRO  
    sCurrentFileName = Dialog.Path File.Save  
    (sCurrentFileName, TextArea1.Text)  
    Form1.Caption = sCurrentFileName  
ENDIF  
END
```

Stampare qualsiasi tipo di articolo è abbastanza facile in Gambas. Il nostro file | La voce di menu Stampa richiamerà questa subroutine quando viene cliccata. Dichiareremo le nostre variabili locali primo:

```
PUBLIC SUB FilePrintItem_Click ()  
  
'variabili intere per contenere i nostri  
valori di margine DIM iLeftMargin AS Integer  
DIM iTopMargin AS Interio DIM  
iRightMargin AS Interio DIM  
iBottomMargin AS Interio
```

Una guida per principianti a

```
'variabili stringa che  
useremo DIM arsTxtToPrint AS  
String [] DIM iTxtPosY AS  
Numero intero  
DIM sTxtLine AS String  
DIM strMsg AS String  
  
'variabili intere che il nostro programma utilizza per calcolare  
le dimensioni della pagina DIM iPrinterAreaWidth AS Integer  
DIM iPrinterAreaHeight AS Integer
```

DOBBIAMO chiamare la subroutine Printer.Setup 0 prima di provare effettivamente a stampare qualcosa. È obbligatorio chiamare questa funzione in Gambas.

```
IF Printer.Setup () THEN RETURN
```

La chiamata successiva garantisce che il formato carta predefinito sia impostato su 8,5 x 11 (Letter). Se non ti trovi negli Stati Uniti, probabilmente dovrebbe essere impostato su "A4" poiché è lo standard utilizzato nella maggior parte dei luoghi al di fuori degli Stati Uniti. La finestra di dialogo di configurazione della stampante Gambas ha come impostazione predefinita le impostazioni "A4" ogni volta che lo chiami, quindi se stai utilizzando una stampante che utilizza solo carta in formato Letter, ignorare questa proprietà è il modo più semplice per gestire quella situazione.

```
'sovrascrive la finestra di dialogo printer.setup in  
modo da non dover' impostare questa impostazione ogni  
volta che si stampa qualcosa Printer.Size = "Lettera"  
'o A4, o qualunque cosa tu usi
```

```
'le stampanti meno recenti non sembrano essere impostate su DPI  
inferiori a 600 in GB' quindi lo imposteremo manualmente per una  
vecchia stampante HP Laserjet III 'se la tua stampante è più  
recente, puoi commentare questa riga Printer.Resolution = 300 'le  
mie impostazioni HPLJIII
```

Per una stampante a 300 dpi, un valore di 300 equivale a un margine di un pollice. Ora, impostiamo i valori di margine utilizzando le proprietà della stampante (o quelle che abbiamo sovrascritto sopra) restituito dalla chiamata Printer.Setup.

```
'imposta il margine sinistro a un pollice, uguale al DPI della  
stampante iLeftMargin = Printer.Resolution
```

```
'imposta il margine superiore a un pollice, uguale al DPI della
```

Una guida per principianti a
stampante **iTopMargin = Printer.Resolution**

'imposta il margine destro a un pollice, uguale al DPI della stampante
iRightMargin = Printer.Resolution

'imposta il margine superiore a un pollice, uguale al DPI della
stampante

Una guida per principianti a

```
iBottomMargin = Printer.Resolution  
  
'Definisce la larghezza dell'area stampabile  
iPrinterAreaWidth = Printer.Width - iLeftMargin - iRightMargin  
  
'Definisce l'altezza dell'area stampabile  
iPrinterAreaHeight = Printer.Height - iTopMargin - iBottomMargin
```

Il calcolo per ottenere iPrinterAreaWidth e iPrinterAreaHeight sottrae i margini sinistro / destro e superiore / inferiore dalla larghezza / altezza totale della pagina. Per la carta in formato Lettera (8,5 x 11) il valore della larghezza sarebbe 8,5 pollici per 300 dpi o 2550 e l'altezza sarebbe 11 pollici per 300 dpi o 3300. Il nostro calcolo sottrae 600 (300 per il margine sinistro + 300 per il margine destro) da i valori di larghezza e altezza restituiti dalla chiamata Printer.Setup. Successivamente, dobbiamo creare la stringa di output per la nostra chiamata Message.Info e visualizzare i dati all'utente:

```
'crea una stringa per MessageBox con i dati dalla chiamata di  
configurazione della stampante strMsg = "Res:" & Str  
(Printer.Resolution) & ", Width:" & Str  
(Printer.Width) & ", Altezza:" & Str (Printer.Height)  
  
'mostra un MessageBox con i dati ottenuti dalla chiamata alla  
configurazione della stampante message.Info (strMsg)
```

Successivamente, dobbiamo analizzare tutto il contenuto del testo nel controllo TextArea1 in una matrice di stringhe. È molto facile da fare in Gambas e richiede solo una singola riga di codice. Per fare ciò, useremo la funzione Split e imposteremo il parametro delimitatore sul carattere '\ n' che designa la fine della riga di testo:

```
'usa un array di stringhe per memorizzare ogni riga di testo in  
TextArea1.Text arsTxtToPrint = Dividi (TextArea1.Text, "\ n")
```

Siamo pronti per la stampa. Per fare ciò, DEVI prima chiamare il metodo Begin e specificare che vuoi che l'oggetto Printer sia il dispositivo di output usato dal metodo Draw:

```
Draw.Begin (stampante) 'Inizializza la routine di disegno per la  
stampante
```

Nel codice seguente, abbiamo chiamato la funzione Draw.Rect per disegnare un bordo in uno margini di pollice intorno alla nostra pagina.

```
'crea un bordo intorno alla pagina Draw.Rect (iLeftMargin, iTopMargin,  
iRightMargin, iBottomMargin)
```

This product is copyrighted by W. Rittinghouse, all rights reserved. It is released to the public under the terms of the GNU General Public License (GPL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

'ora passeremo in rassegna ogni riga dell'array e la stamperemo

Una guida per principianti a

```
PER OGNI sTxtLine IN arsTxtToPrint

    'incrementa la posizione della linea verticale (Y) per ogni linea
    stampata iTxtPosY = iTxtPosY + Draw.TextHeight (sTxtLine)

    'prova la posizione della linea ... se è > la parte inferiore
    'dell'area della stampante', dobbiamo resettarla a zero ed
    eseguire una nuova pagina SE iTxtPosY >= (iPrinterAreaHeight)
    ALLORA
        iTxtPosY = 0 'reinizializza la Current Text Position Y
        Printer.NewPage

    'crea un bordo intorno alla nuova pagina Draw.Rect (iLeftMargin,
    iTopMargin, iRightMargin, iBottomMargin)
FINISCI SE 'il nostro test per vedere se la linea supera il margine
inferiore
```

Ora, chiameremo Draw.Text per stampare effettivamente il testo. Il primo parametro che Draw.Text accetta è l'oggetto stringa sTxtLine che rappresenta la nostra riga di testo da stampare dall'array di stringhe arsTxtToPrint. I due parametri successivi, iLeftMargin e (iTopMargin + iTxtPosY) vengono utilizzati per definire il margine sinistro e la posizione Y del testo verticale corrente. Ogni riga viene incrementata del valore contenuto nella proprietà Draw.TextHeight. Facoltativamente, è possibile specificare i margini destro e inferiore e una proprietà di allineamento. Nel nostro caso sotto, abbiamo specificato il margine destro, definito da iPrinterAreaWidth (come abbiamo calcolato sopra) e omesso il margine inferiore poiché è mantenuto nel codice del FOR LOOP sopra. Infine, abbiamo specificato l'allineamento del testo da allineare a sinistra.

```
'Ora stampiamo la riga di testo nella nuova posizione della riga
Draw.Text (sTxtLine, iLeftMargin, iTopMargin + iTxtPosY, iPrinterAreaWidth,
Align.Left) IL PROSSIMO 'iterazione del ciclo FOR
```

Per terminare il processo di stampa, DEVI ricordarti di chiamare Draw.

```
Draw.End 'Quindi stampare il contenuto finale ed
espellere l'ultima pagina FINE
```

Poiché il nostro programma di esempio funziona solo con file di testo, esiste un metodo di stampa alternativo, a volte più semplice. È possibile utilizzare il comando pr (print) integrato nel sistema. Viene richiamato utilizzando il metodo Gambas SHELL. Questa chiamata può consentire di stampare qualsiasi file di testo dall'interno del programma direttamente sulla stampante predefinita. Per vedere come funziona questo metodo, aggiungiamo una voce di menu al nostro programma e proviamo questo. Vai a Form1.form e premi CTRLE per

This product is (C) 2009 by John Williams. All rights reserved. It is released under the terms of the GNU General Public License (GPL) and may not be distributed without the author's consent of the author.

Una guida per principianti a visualizzare l'editor di menu. Sotto il file | Voce di menu Stampa, aggiungi una nuova voce, FileSysPrintItem e assegnagli la didascalia Print (metodo di sistema). Crea un evento clic e aggiungi questo codice:

```
PUBLIC SUB FileSysPrintItem_Click ()
```

This product is (a) 0551.0000.0000.0000. All rights are reserved.
Gambas User Guide under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
DIM aString AS String

aString = "pr h \" "& sCurrentFileName & " \" & "o 10 <\\" "&
sCurrentFileName & " \\" | lpr"

message.Info (aString,
"Ok") SHELL aString
FINE
```

Sembra molto più difficile di quanto lo sia davvero. Andiamo riga per riga nel codice. Innanzitutto, dichiariamo una variabile stringa, aString. Successivamente dobbiamo costruire la stringa che sarà passata al sistema come un comando della console e assegnarla alla variabile aString. Poiché dobbiamo assicurarci che la stringa sia costruita esattamente come dovrebbe essere digitata dalla console, dobbiamo assicurarci che contenga virgolette incorporate. La prima parte della stringa:

```
"pr h \" "
```

richiamerà il comando di sistema pr. Il parametro h specifica che vogliamo avere un'intestazione stampata su ogni pagina, in questo caso sarà il nome del file corrente, contenuto nella variabile sCurrentFileName, che, nel nostro programma, è di portata globale. Questo nome file deve essere incorporato tra virgolette, quindi è necessario utilizzare il carattere \ prima delle virgolette di apertura e terminare questa parte della stringa con un altro segno di virgolette. La parte successiva della stringa:

```
& sCurrentFileName & "\\" "
```

concatena il nome del file e la virgoletta di chiusura per il nome del file passato come una variabile stringa, ancora una volta preceduta da un carattere \. Le virgolette successive terminano questa parte della stringa. La parte successiva del comando da concatenare è:

```
& "o 10 <\\" "
```

che aggiunge il parametro o (che specifica un offset) e un valore di 10 spazi. Il carattere < viene utilizzato sulla riga di comando per designare il flusso di input in arrivo dal nome del file che segue il carattere <. In questo caso, dobbiamo delimitare nuovamente la nostra stringa con virgolette, quindi il carattere \ viene utilizzato per consentire al carattere di virgolette di far parte della stringa di output. Il carattere di virgolette successivo inizia la stringa del nome del file tra virgolette. L'ultima parte del nostro comando è:

Una guida per principianti a

```
sCurrentFileName & "\" | lpr "
```

Una guida per principianti a

Specifica il file da leggere (e stampare) e termina il nome del file tra virgolette come spiegato prima. La parte finale della stringa è il | carattere che reindirizza (reindirizza) l'output alla stampante di linea (lpr). In tutto, sono necessarie una dichiarazione di variabile e tre righe di codice per stampare un file utilizzando questa tecnica. Entrambe le tecniche funzioneranno, quindi scegli quella più adatta a te. Un altro metodo che puoi usare se hai installato KDE è stampare su un file postscript e inviare il file .ps a "kprinter". Questo ti fornirà l'accesso alle finestre di dialogo di stampa di KDE ea tutte le stampanti e le funzionalità CUPS installate sul tuo sistema.

Se l'utente fa clic sul file | Esci dalla voce di menu potremmo semplicemente chiudere il programma. Tuttavia, vogliamo dare all'utente la possibilità di salvare prima di uscire, quindi verrà visualizzata una finestra di dialogo per chiedere se desidera salvare il file. Se è così, lo salveremo. In caso contrario, usciamo semplicemente dal programma.

```
PUBLIC SUB FileExitItem_Click ()  
    DIM iResult AS Integer  
    iResult = Message.Question ("Salva il tuo lavoro?", "Sì", "No",  
    "Annulla") SELEZIONA CASO iResult  
        CASO 0  
            File.Save (sCurrentFileName, TextArea1.Text)  
        CASO 1  
        CASO 2  
        CASO  
        ALTRO  
    FINE SELEZIONA  
    ME.close  
END
```

Questo è tutto per il menu File. Ora, il vero potere di Gambas diventerà ovvio man mano che costruiremo il codice per le funzioni del menu Modifica. Da questo menu, vogliamo che l'utente sia in grado di selezionare tutto il testo, annullare, ripetere, tagliare, incollare, copiare o chiamare una routine di configurazione. Per i nostri scopi, la routine di configurazione sarà piuttosto semplice, ma dovrebbe essere sufficiente per mostrarti come eseguire l'operazione.

Se l'utente fa clic su Modifica | Seleziona tutto la voce di menu, selezionerà tutto il testo nella TextArea. Questo viene fatto ottenendo la lunghezza del testo utilizzando la proprietà TextArea1.Length e passando tale valore al metodo TextArea1.Selection con un punto iniziale zero e un punto finale Lunghezza caratteri da quel punto iniziale, come mostrato nel

Una guida per principianti a codice seguente.

```
PUBLIC SUB EditSelectAllItem_Click ()  
    DIM ilength AS Integer  
    ilength = TextAreal.Length  
    TextAreal.Selection (0,  
    ilength) END
```

Una guida per principianti a

Il codice sopra potrebbe anche essere stato scritto in un modo più semplificato senza utilizzare la variabile Integer iLength. Avrebbe potuto essere ridotto a un singololinea, come mostrato di seguito:

```
PUBLIC SUB EditSelectAllItem_Click ()  
    TextArea1.Selection (0,  
        TextArea1.Length)  
FINE
```

Entrambi i metodi funzioneranno, ma la preferenza dell'autore è quella di utilizzare il secondo metodo poiché riflette la semplicità di Gambas durante la programmazione. Questa semplicità vede nell'uso dei metodi incorporati Undo, Redo, Cut, Paste e Copy forniti da Gambas anche per TextArea1.Text. Gambas fornisce questi metodi, quindi fornire questa funzionalità agli utenti del programma è semplicemente una questione di aggiungere una singola riga di codice per l'evento clic di ciascuna voce di menu, come mostrato di seguito:

```
PUBLIC SUB EditUndoItem_Click  
    () TextArea1.Undo  
FINE  
  
PUBLIC SUB EditRedoItem_Click  
    () TextArea1.Redo  
FINE  
  
PUBLIC SUB EditCutItem_Click  
    () TextArea1.Cut  
FINE  
  
PUBLIC SUB EditPasteItem_Click  
    () TextArea1.Paste  
FINE  
  
PUBLIC SUB EditCopyItem_Click  
    () TextArea1.Copy  
FINE
```

La nostra modifica | L'opzione del menu Preferenze verrà utilizzata per mostrarti come portare una seconda forma nel tuo programma. Salveremo i dati raccolti da Form2.form in un file di configurazione denominato FileOps.conf e questo ci consentirà di passare i valori salvati avanti e indietro tra il modulo principale, Form1.form e il secondo modulo, Form2.form. Questo è il modo più semplice per scambiare dati tra classi, ma è possibile passare dati come parametri tra i moduli. Ne parleremo più avanti in questo libro. La subroutine EditPrefsItem_Click () è piuttosto semplice. Mostriamo solo la

Una guida per principianti a
seconda forma.

```
PUBLIC SUB EditPrefsItem_Click ()
```

Una guida per principianti a

```
Form2.Show  
END
```

Ecco come apparirà Form2.form:

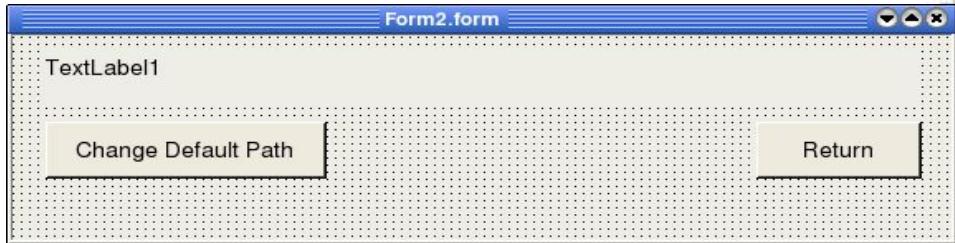


Figura 73 Modalità di progettazione Form2.form.

Come puoi vedere, è piuttosto semplice. Un `TextLabel` viene utilizzato per visualizzare il valore predefinito. Esistono due pulsanti, uno per modificare il percorso predefinito chiamando la routine `Dialog.SelectDirectory` standard e l'altro per tornare al processo chiamante, `Form1.form` in questo caso. Quando il modulo è aperto, visualizzeremo la didascalia "Preferenze" nella parte superiore della finestra. La routine del costruttore `_new()` viene chiamata automaticamente. Ecco il codice per il nostro Form2:

```
"File di classe Gambas  
PUBLIC SUB Form_Open()  
    Form2.Caption = "Preferenze"  
END
```

Il costruttore chiamerà la nostra subroutine `ReadPrefs`, passando il nome del file di configurazione come parametro. `FileOps.conf` è il nome che useremo per conservare i dati di configurazione della nostra applicazione. Successivamente, popoliamo la proprietà `TextLabel1.Text` con la directory predefinita corrente, che si trova nella posizione in cui risiede l'applicazione su disco.

```
PUBLIC SUB _new()  
    ReadPrefs ("FileOps.conf")  
    TextLabel1.Text = Application.Path & "è la directory predefinita."  
FINE
```

Quando l'utente fa clic sul pulsante Modifica percorso predefinito, la seguente routine è chiamato:

```
PUBLIC SUB Button1_Click()  
    Dialog.SelectDirectory
```

This product is (C) 2005 by John Doe, all rights are reserved. It is released to the Gambas User Community under an open source license. It may not be distributed under any other conditions without express written consent from the author.

Una guida per principianti a

```
TextLabel1.Text = Dialog.Path  
FINE
```

Una guida per principianti a

La subroutine precedente chiama semplicemente la funzione di dialogo standard di Gambas Dialog.SelectDirectory e aggiorna la proprietà TextLabel1.Text con ciò che l'utente ha scelto come valore predefinito. Quando l'utente ritorna da questa subroutine, l'unica altra opzione nel modulo è fare clic sul pulsante Return, che esegue questo codice:

```
PUBLIC SUB Button2_Click
() DIM MyPath AS String

MyPath = application.Path & / "FileOps.conf"
WritePrefs (MyPath)
Form2.Close
END
```

Come puoi vedere dall'alto, una variabile stringa locale MyPath viene dichiarata e popolata concatenando il percorso dell'applicazione corrente con il nome del file di configurazione FileOps.conf. Successivamente, chiamiamo la nostra subroutine WritePrefs e passiamo il valore di MyPath come parametro. Quel percorso verrà memorizzato nel file e potrà essere letto da qualsiasi altra classe che abbia bisogno di trovare queste informazioni. Ecco il ReadPrefs subroutine per eseguire questa operazione:

```
PUBLIC SUB ReadPrefs (nome_file AS
String) 'dichiara le nostre variabili
locali
DIM sInputLine AS String
DIM hFileIn AS File

'controlla se il file esiste già, in tal caso
apriolo. IF Exist (filename) THEN
OPEN filename FOR READ AS #hFileIn

'leggi la nostra singola riga di dati
'se ci fossero più linee, potremmo impostare un WHILE NOT Eof
LOOP LINE INPUT #hFileIn, sInputLine

'aggiorna i dati del display
TextLabel1.Text = sInputLine

'chiudi il file
CHIUDI #
hFileIn
ALTRO
'non c'era alcun file .conf da aprire
Message.Info ("Nessun file .conf delle preferenze
presente.", "OK") ENDIF
FINE
```

Una guida per principianti a

La routine reciproca, WritePrefs () è mostrata di seguito:

```
PUBLIC SUB WritePrefs (nome file AS
    String) 'dichiara le nostre variabili
    locali
    DIM MyPath AS String
    DIM hFileOut AS File

    'nota che stiamo usando il valore Dialog.Path, non Application.Path
    MyPath = Dialog.Path & / filename

    'se esiste già un file di configurazione, aprilo per
    la scrittura' in caso contrario, aprilo per creare e
    scrivere
    SE esiste (MyPath) ALLORA
        APRI MyPath PER SCRIVERE COME #
        hFileOut ALTRO
        APRI MyPath PER SCRIVERE CREA COME #
        hFileOut ENDIF

    'stampa i nostri dati nel
    file PRINT # hFileOut,
    MyPath

    'chiudi il file
    CHIUDI #
    hFileOut
FINE
```

Come puoi vedere, è piuttosto semplice ma può essere facilmente espanso per ospitare quasi tutti i set di dati di configurazione che la tua applicazione potrebbe dover memorizzare. Sei limitato solo dalla tua immaginazione in Gambas. Ora, diamo un'occhiata al nostro codice del menu Formato. Nel menu Formato, le uniche opzioni che abbiamo sono cambiare i colori o i caratteri. Utilizzeremo le finestre di dialogo standard di Gambas per svolgere entrambe le attività. Ecco come si fa:

```
PUBLIC SUB FormatColorItem_Click ()

    'per prima cosa, fai apparire un messaggio per spiegare come lavoreremo
    Message.Info ("Seleziona prima il colore di sfondo, poi il colore del
    testo.", "OK")

    'imposta la didascalia della finestra di dialogo per ricordare
    all'utente che stiamo selezionando BACKGROUND Dialog.Title = "Seleziona
    colore di sfondo"
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It may not be distributed or modified without the express written consent of the author.

Una guida per principianti a

```
'chiama la routine di dialogo standard per selezionare il colore  
Dialog.SelectColor
```

```
'aggiorna il nostro TextArea1.Backcolor con il valore selezionato.  
TextArea1.BackColor = Dialog.Color
```

Una guida per principianti a

```
'imposta la didascalia della finestra di dialogo per ricordare  
all'utente che stiamo selezionando IN PRIMO PIANO Dialog.Title =  
"Seleziona colore testo"  
  
'chiama la routine di dialogo standard per selezionare il colore  
Dialog.SelectColor  
  
'aggiorna il nostro TextArea1.ForeColor con il valore selezionato  
TextArea1.ForeColor = Dialog.Color  
FINE
```

La selezione del carattere e dei suoi attributi per il nostro programma è altrettanto semplice era scegliere i colori. Fondamentalmente, ci metteremo la didascalia della finestra di dialogo, chiama la routine Dialog.SelectFont standard e aggiorna la nostra proprietà TextArea1.Font, in questo modo:

```
PUBLIC SUB FormatFontItem_Click  
() Dialog.Title = "Seleziona  
carattere" Dialog.SelectFont  
TextArea1.Font = Dialog.Font  
FINE
```

All'inizio del nostro programma, nella routine del costruttore, dovevamo abilitare il timer. Per far sì che il timer faccia qualcosa, è necessario programmarlo. Facendo doppio clic sul controllo del timer si accederà alla subroutine Timer1_Timer () , mostrata di seguito. Se ricordi, faremo in modo che il nostro programma visualizzi l'ora corrente su un'etichetta e la posizione corrente del cursore all'interno dell'area di testo su un'altra etichetta. Ecco come eseguiamo questo compito:

```
PUBLIC SUB Timer1_Timer ()  
'prendi l'ora per adesso e converti in una stringa'  
per inserire la proprietà TimeLabel.Text  
TimeLabel.Text = Str $ (ora)  
  
'otterremo i dati di riga e colonna dalle proprietà .Line  
e' .Column del controllo Label e aggiorneremo la  
proprietà PosLabel.Text con i loro valori:  
PosLabel.Text = "Line:" & Str (TextArea1.Line) & "Col:" & Str  
(TextArea1.Column)  
FINE
```

L'ultimo menu è il nostro menu Aiuto. Ha due opzioni, Contenuti e Informazioni.

```
PUBLIC SUB HelpContentsItem_Click ()  
SHELL "konqueror \" file: / root / My Documents / Gamasas for Beginners / FileOps /  
Help.html \ ""
```

FINE

Una guida per principianti a

La chiamata della shell precedente punta a un file denominato
Help.html.For scopi di

Una guida per principianti a

dimostrando questa funzione, copia semplicemente qualsiasi file .html sul tuo sistema nella directory del programma e rinominalo help.html. In alternativa, puoi creare il tuo file .html se sai come farlo. Questo è ciò che farebbero i veri programmati. □ Ecco l'ultima cosa che dobbiamo fare:

```
PUBLIC SUB HelpAboutItem_Click ()  
    Message.Info ("<b> Simple Text Editor </b> di <br> J. Rittinghouse",  
    "OK") END
```

Esegui il programma e quando sei soddisfatto, passiamo a fare un po' di matematica.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gamasutra User Community under an Open Content License (OCL) and may not be distributed or copied to any other teams or conditions without the express written consent of the author.

Capitolo 10 - Operazioni matematiche

Gambas esegue operazioni matematiche con una libreria di funzioni intrinseche e derivate. Secondo l'encyclopedia online Wikipedia¹⁵, “un *intrinsecofunzione* è una funzione disponibile in un dato linguaggio la cui implementazione è gestita specialmente dal compilatore. In genere, sostituisce una sequenza di istruzioni generate automaticamente per la chiamata di funzione originale, simile a una funzione inline. I compilatori che implementano funzioni intrinseche generalmente le abilitano solo quando l'utente ha richiesto l'ottimizzazione, tornando a un'implementazione predefinita fornita dall'ambiente di runtime del linguaggio in caso contrario.”

Le funzioni intrinseche sono incorporate nel compilatore di un linguaggio di programmazione, di solito per sfruttare caratteristiche specifiche della CPU che sono inefficienti da gestire tramite funzioni esterne. L'ottimizzatore del compilatore e il generatore di codice integrano completamente le funzioni intrinseche, il che può portare a sorprendenti accelerazioni delle prestazioni nei calcoli. Le funzioni derivate, d'altra parte, sono costruite in Gambas utilizzando formule matematiche note per applicare algoritmi specifici che implementano funzioni intrinseche. Alla fine di questo capitolo si trova una tabella delle funzioni derivate e dei loro algoritmi corrispondenti.

Precedenza delle operazioni

Quando usi vari operatori in un'espressione in Gambas, esiste una precedenza specifica delle operazioni che Gambas utilizza per eseguire l'operazione. Viene seguito un protocollo ben definito:

- Tutte le espressioni sono semplificati tra parentesi dall'interno all'esterno
- Vengono eseguite tutte le operazioni esponenziali, procedendo da sinistra a destra
- Tutti i prodotti e quozienti vengono eseguiti procedendo da sinistra a destra
- Vengono eseguite tutte le somme e le differenze procedendo da sinistra a destra

Tenendo presente l'ordine delle operazioni (cioè la precedenza), è possibile utilizzare le funzioni matematiche in Gambas per eseguire praticamente qualsiasi operazione matematica necessaria per i propri programmi. Tratteremo ogni funzione, intrinseca o derivata in questo capitolo e forniremo esempi di come vengono utilizzate. C'è untabella fornita alla fine di questo capitolo che mostra la versione "sviluppatore" ufficiale della gerarchia delle operazioni, come

Una guida per principianti a
definita nel file sorgente `gb_reserved_temp.h` Gambas.

15 Vedi URL: http://en.wikipedia.org/wiki/Intrinsic_function.

Una guida per principianti a

Addominali

Abs è una funzione intrinseca che restituisce il valore assoluto di un numero. Il valore assoluto di un numero è la sua grandezza senza segno. Ad esempio, Abs (1)e Abs (-1) restituiscono entrambi 1. L'argomento numero può essere qualsiasi espressione numerica valida. Se numero contiene Null, viene restituito un errore; se è una variabile inizializzata, viene restituito zero. La sintassi standard del linguaggio Gambas è:

Abs (numero)

L'esempio seguente usa la funzione Abs per eseguire il calcolo il valore assoluto di un numero. Prova questo sulla tua console:

```
STATICO PUBBLICO SUB
    Principale ()
DIM MyNum AS Variant

MyNum = Abs (1.03E3)
PRINT MyNum
MyNum = Abs (2.5 +
6.5) PRINT MyNum
FINE
```

Acs / ACos

Acs / ACos è una funzione derivata che calcola il coseno inverso (arcocoseno) di un numero. Y = ACos (X) restituisce l'arcocoseno per ogni elemento di X. Per elementi reali di X nel dominio [-1,1], ACos (X) è reale e nell'intervallo. Per elementi reali di X al di fuori del dominio [-1,1], ACos (X) è complesso. La funzione ACos opera in modo elementare sugli array. I domini e gli intervalli della funzione includono valori complessi. Tutti gli angoli vengono restituiti in radianti. La sintassi è:

valore = ACos (numero)

Esempio:

```
PRINT Acs
(0.5)
1.047197551197

STAMPA Acs
(1)
```

3.14159265359

Una guida per principianti a

Una guida per principianti a

Acsh / ACosh

Acsh / ACosh è una funzione derivata che calcola l'arco iperbolico coseno (coseno iperbolico inverso) di un numero X. Funziona in modo elementare su array. Tutti gli angoli sono in radianti. Il dominio include numeri reali X tali che $X \geq 1$. L'intervallo della funzione arcoseno iperbolica è limitato a numeri reali non negativi Y tali che $Y \geq 0$. La sintassi del linguaggio Gambas è la seguente:

```
valore = Acsh (numero)
valore = ACosh (numero)
```

Ecco un esempio da provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
    DIM y AS
    Variante Y = 2
    STAMPA Acsh
(Y) END
```

La console risponde con:

```
1.316957896925
```

Asn / ASin

Asn / ASin è una funzione derivata che calcola l'arcoseno di un numero. La funzione ASin 0 calcola il valore principale dell'arcoseno di x. Il valore di x dovrebbe essere compreso nell'intervallo $[1,1]$. Dopo il completamento con successo, Asn 0 restituisce l'arcoseno di x, nell'intervallo $[\pi/2, \pi/2]$ radianti. Se il valore di x non è compreso nell'intervallo $[1,1]$, viene restituito 0,0. La funzione ASin 0 fallirà se il valore utilizzato per x non è compreso nell'intervallo $[1,1]$. La sintassi standard del linguaggio Gambas è:

```
valore = Asn (numero)
valore = ASin (numero)
```

Ecco un esempio:

```
STATICO PUBBLICO SUB Principale ()
    DIM y AS
    Variante Y = 0,5
    PRINT Asn
(Y) Y = 1.0
```

Una guida per principianti a

PRINT Asn
(Y)

Una guida per principianti a

FINE

La console risponde con:

```
0.523598775598
1.570796326795
```

Asnh / ASinh

Asnh / ASinh è una funzione derivata che calcola l'iperbolico inverso seno (arcoseno iperbolico) per ogni elemento di X. La funzione ASinh opera per elementi sugli array. I domini e gli intervalli della funzione includono valori complessi. Tutti gli angoli sono espressi in radianti. La sintassi standard del linguaggio Gambas è:

```
valore = Asnh (numero)
valore = ASinh (numero)
```

Un esempio è il seguente:

```
STATICO PUBBLICO SUB Principale ()
DIM y AS Variant

y = 2
PRINT Asnh
(2) END
```

La console risponde con:

```
1.443635475179
```

Atn / ATan

Atn / ATan è una funzione intrinseca che calcola l'arcotangente di un numero. Per gli elementi reali di X, ATan (X) è nell'intervallo $[-\pi/2, \pi/2]$. La funzione ATan opera in modo elementare sugli array. I domini e gli intervalli della funzione includono valori complessi. Tutti gli angoli sono in radianti. La sintassi standard del linguaggio Gambas è:

```
valore = Atn (numero)
valore = ATan (numero)
```

Esempio:

This product is (C) 2005 by General Electric Company. All rights reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed or modified without the express written consent of the author.

Una guida per principianti a

```
STATICO PUBBLICO SUB Principale ()
    DIM y AS Variant

    Y = 0,5
    STAMPA FINE
    AN (Y)
```

La console risponde con:

0.463647609001

Atnh / ATanh

Atnh / ATanh è una funzione derivata che calcola l'iperbolico inverso tangente (o arcotangente iperbolico) di un numero. La funzione opera in base agli elementi sugli array. I suoi domini e gli intervalli includono valori complessi. Tutti gli angoli sono in radianti. La sintassi del linguaggio Gambas di $Y = \text{ATanh}(X)$ restituisce l'arcotangente iperbolico per ogni elemento di X.

```
valore = Atnh (numero)
valore = ATanh (numero)
```

Ecco un esempio che puoi provare sulla console usando Atanh ():

```
STATICO PUBBLICO SUB Principale ()
    DIM MyResult AS Float
    DIM MyNum AS Float

    MyNum = 0,5
    MyResult = ATanh (0,5)
    PRINT "L'arcotangente iperbolico di" & MyNum & "è:" & MyResult END
```

La console risponde con:

0,549306144334

Cos

Cos () è una funzione intrinseca che restituisce il coseno di un angolo. La funzione Cos prende un angolo e restituisce il rapporto tra due lati di un triangolo rettangolo. Il rapporto è la lunghezza del lato adiacente all'angolo diviso per la lunghezza dell'ipotenusa. Il risultato è compreso tra 1 e -1. Per convertire i gradi in radianti,

Una guida per principianti a

moltiplicare gradi di \square / 180. Per convertire i radianti in gradi, moltiplicare i radianti per 180 / \square . Standard La sintassi del linguaggio Gambas è:

```
Risultato = Cos (numero)
```

L'argomento numero può essere qualsiasi espressione numerica valida che esprime un angolo in radianti. L'esempio seguente usa la funzione Cos per restituire il coseno di un angolo:

```
STATICO PUBBLICO SUB Principale ()
  DIM MyAngle AS Float
  DIM MySecant AS Float

  MyAngle = 1.3           'Definisci l'angolo in
  radianti. MySecant = 1 / Cos (MyAngle)  'Calcola
  secante.

  PRINT "La secante dell'angolo" & MyAngle & "(in radianti) è:" &
  MySecant END
```

La console risponde con:

```
La secante dell'angolo 1.3 (in radianti) è: 3,738334127075
```

Cosh

Cosh è una funzione derivata che calcola il coseno iperbolico di un numero. La funzione Cosh opera in modo elementare sugli array. I domini e gli intervalli della funzione includono valori complessi. Tutti gli angoli sono in radianti. La sintassi standard del linguaggio Gambas è:

```
Valore = Cosh (numero)
```

Ecco un esempio da provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
  DIM MyAngle AS Float
  DIM MyResult AS Float

  MyAngle = 1.0 'Angolo espresso in radianti.
  MyResult = Cosh (MyAngle)
  PRINT "Cosh () of angle" & MyAngle & "(in radianti) è:" & MyResult
END
```

La console risponde con:

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas 1.5.0 Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
Cosh () dell'angolo 1 (in radianti) è: 1,543080634815
```

Deg e Rad

Deg è una funzione di conversione intrinseca che converte i radianti in gradi. Rad è una conversione reciproca funzione che converte i gradi in radianti. La sintassi del linguaggio Gambas per Deg e Rad è:

```
Valore = Deg (Angolo)
Valore = Rad (Angolo)
```

Un esempio da provare sulla console è:

```
STATIC PUBLIC SUB Main
    () PRINT Deg (Pi / 2)
    PRINT Rad (90)
    STAMPA Rad (180) Pi
FINE
```

La console risponde con:

```
90
1.570796326795
0
```

Exp

Exp è una funzione intrinseca che calcola l'esponenziale di un numero. Restituisce e (la base dei logaritmi naturali, dove la costante e è approssimativamente 2.718282) elevato a potenza. L'argomento numero può essere qualsiasi espressione numerica valida. Se il valore di numero supera 709.782712893, si verifica un errore. La funzione Exp completa l'azione della funzione Log e talvolta viene definita antilogaritmo. La sintassi del linguaggio Gambas è:

```
Valore = Exp (numero)
```

Ecco un esempio:

```
STATIC PUBLIC SUB Main
    () PRINT Exp (1)
FINE
```

Una guida per principianti a

La console risponde con:

2.718281828459

Ecco un altro esempio che utilizza la funzione Exp per restituire e elevato a una potenza:

```
STATICO PUBBLICO SUB Principale ()
    DIM MyAngle AS Float
    DIM MyHSin AS Float 'Definisci l'angolo in radianti.

    MyAngle = 1.3      'Calcola il seno iperbolico
    MyHSin = (Exp (MyAngle) Exp (1 * MyAngle)) / 2
    PRINT MyHSin
FINE
```

La console risponde con:

1.698382437293

Fix e Frac

Fix è una funzione intrinseca che restituisce la parte intera di un numero. Frac è una funzione intrinseca che calcola la parte frazionaria di un numero. Si noti che nell'esempio seguente, Frac () restituisce il valore assoluto del risultato frazionario di

□ . La sintassi standard del linguaggio Gambas è:

```
Valore = Fix (numero)
Valore = Frac (numero)
```

Esempio:

```
STATIC PUBLIC SUB Main
    () PRINT Fix (Pi)
    PRINT Frac (Pi)
    PRINT Fix (Pi)
    PRINT Frac (Pi)
FINE
```

La console risponde con:

**3
0.14159265359**

Una guida per principianti a

3

0.14159265359

Una guida per principianti a

Int

Int (Number) restituisce la parte intera matematica di un numero, ovvero il maggiore intero minore di numero.

```
Valore = Int (numero)
```

Ecco alcuni esempi che puoi provare sulla console:

```
PRINT Int  
(Pi) 3  
  
PRINT Int (Pi)  
4
```

Log

Log è una funzione intrinseca che calcola il logaritmo naturale di e. Il logaritmo avente base e. La costante e è approssimativamente 2,718282 dove e è un numero univoco con la proprietà che l'area della regione delimitata dall'iperbole $y = 1/x$, l'asse x e le linee verticali $x = 1$ e $x = e$ è 1. La notazione $\ln x$ è usata in fisica e ingegneria per denotare il logaritmo naturale, mentre i matematici usano comunemente la notazione $\log x$. L'argomento Numero può essere qualsiasi espressione numerica valida maggiore di 0. La sintassi del linguaggio Gambas standard è:

```
valore = Log (numero)
```

Ecco un esempio:

```
STATIC PUBLIC SUB Main  
() PRINT Log (2.71828)  
PRINT Log (1)  
FINE
```

La console risponde con:

```
0.999999327347  
0
```

Puoi creare un file funzione derivata per calcolare i logaritmi basen per qualsiasi numero x dividendo il logaritmo naturale di x per il logaritmo naturale di n come

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

segue:

$$\text{Logn}(x) = \text{Log}(x) / \text{Log}$$

(n) Ecco un esempio da

provare:

```
STATICO PUBBLICO SUB Principale ()
    DIM n AS Float
    DIM x AS Float
    DIM logn AS Float

    n = 2
    x = 10
    logn = Log (x) / Log (n)
    PRINT logn & "è il risultato di Logn (" & x & ")"
END
```

La console risponde con:

```
3.321928094887 è il risultato di Logn (10)
```

Log10

Log10 è una funzione derivata che calcola il logaritmo decimale di un numero. $\text{Log10}(x) = \text{Log}(x) / \text{Log}(10)$. La sintassi del linguaggio Gambas è:

```
valore = Log10 (Numero)
```

Esempio :

```
STAMPA Log10
(10) 1
```

Max e Min

Max restituisce l'espressione maggiore dell'elenco. Le espressioni devono essere numeri o valori di data / ora. Min restituisce l'espressione più piccola dell'elenco. Le espressioni devono essere numeri o valori di data / ora.

```
valore = Max (Espressione [, Espressione ...])
valore = Min (Espressione [, Espressione ...])
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Gamas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Esempi:

```
STATIC PUBLIC SUB Main ()  
    PRINT Max (6, 4, 7, 1, 3)  
    PRINT Max (Now, CDate ("01/01/1900"), CDate  
    ("01/01/2100")) PRINT  
    STAMPA Min (6, 4, 7, 1, 3)  
    PRINT Min (Now, CDate ("01/01/1900"), CDate  
    ("01/01/2100")) END
```

La console risponde con:

```
7  
01/01/2100  
  
1  
01/01/1900
```

Pi

Pi restituisce $\pi * \text{Number}$. Se Number non è specificato, si presume che sia uno.

Sintassi standard del linguaggio Gambas è:

```
Risultato = Pi ([Number])
```

Esempi:

```
STAMPA Pi  
3.14159265359  
  
STAMPA Pi  
(0,5)  
1,570796326795
```

Randomizza e Rnd

Randomize inizializza un generatore di numeri pseudocasuali. Utilizza una metodologia che si semina con la data e l'ora correnti. La sintassi del linguaggio Gambas è:

```
Randomizza (numero)
```

dove l'argomento numero può essere qualsiasi valore numerico valido espressione

Una guida per principianti a
usare il
parametro Number per inizializzare il numero casuale della funzione Rnd

Una guida per principianti a

generatore, dandogli un nuovo valore di seme. Se si omette il numero, il valore restituito dal timer di sistema viene utilizzato come nuovo valore di inizializzazione. Se Randomize non viene utilizzato, la funzione Rnd (senza argomenti) utilizza lo stesso numero come seme la prima volta che viene chiamato, e successivamente utilizza l'ultimo numero generato come valore seme. In altre parole, per ogni dato seme iniziale, viene generata la stessa sequenza numerica perché ogni chiamata successiva alla funzione Rnd utilizza la precedente numero come seme per il numero successivo nella sequenza.

Per ripetere sequenze di numeri casuali, chiama Rnd con un argomento negativo immediatamente prima di utilizzare Randomize con un argomento numerico. Prima di chiamare Rnd, utilizzare l'istruzione Randomize senza argomenti per inizializzare il filegeneratore di numeri casuali con un seme basato sul timer di sistema. L'uso di Randomize con lo stesso valore per number non ripete la sequenza precedente.

Rnd calcola un numero in virgola mobile pseudocasuale, utilizzando l'algoritmo di Lehmer. Se non vengono specificati parametri, Rnd restituisce un numero pseudocasuale nell'intervallo [0, 1]. Se viene specificato un solo parametro, Rnd restituisce un numero pseudo casuale nell'intervallo [0, Min]. Se vengono specificati entrambi i parametri, Rnd restituisce un numero pseudocasuale nell'intervallo [Min, Max]. La sintassi standard del linguaggio Gambas per Rnd è:

```
Rnd ([Min [, Max]])
```

L'argomento numero può essere qualsiasi espressione numerica valida. La funzione Rnd restituisce un valore inferiore a 1 ma maggiore o uguale a 0. Il valore di numero determina il modo in cui Rnd genera un numero casuale. Per produrre numeri interi casuali che rientrano in un determinato intervallo, utilizza questa formula:

```
Int ((upperbound lowerbound + 1) * Rnd + lowerbound)
```

Qui, upperbound è il numero più alto nell'intervallo e lowerbound è il numero più basso nell'intervallo. Ecco un programma di esempio da provare sulla console:

```
STATICO PUBBLICO SUB Principale ()
  DIM Dice AS Integer

  PRINT "Tra 0 e 1:"; PRINT
  Rnd
```

Una guida per principianti a

```
PRINT "Tra 0 e 2:"; STAMPA  
Rnd (2)  
PRINT "Tra Pi e Pi * 2:";
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambar User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PRINT Rnd (Pi, Pi  
(2)) PRINT  
Rendi casuale  
  
FARE MENTRE Dadi <> 1  
Dadi = Int (Rnd (1,7))  
'produrre un numero casuale compreso tra 1 e 6 per simulare il lancio  
dei dadi STAMPA "Hai lanciato un" e dadi  
FINE  
DEL  
CICLO
```

La console risponde con:

```
Tra 0 e 1: 7,826369255781E6  
Tra 0 e 2: 0,263075576164 Tra Pi e  
Pi * 2: 5,515396781706  
  
Hai tirato un  
6 Hai tirato  
un 4 Hai  
tirato un 1
```

Il giro

Round arrotonda un numero al numero intero più vicino se Cifre non è specificato. Se Cifre è specificato, arrotonda a 10^{cifre} .

```
Valore = rotondo (numero [, cifre])
```

Esempio:

```
STATIC PUBLIC SUB Main  
() PRINT Round (Pi, 2)  
PRINT Round (1972,  
2) END
```

La console risponde con:

```
3.14  
2000
```

Sgn

Sgn restituisce un numero intero che indica il segno di un numero. Se il numero è zero, ritorna zero. Se il numero è strettamente positivo, restituisce

Una guida per principianti a
il numero intero

This product is (C) 2005 by John W. Rittinghouse. It is released to the
Gambas User Community under an Open Content License. It may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

numero +1. Se il numero è strettamente negativo, restituisce il numero intero 1. La sintassi del linguaggio Gambas è:

```
Segno = Sgn (Numero)
```

Esempio:

```
STATIC PUBLIC SUB Main
() PRINT Sgn (Pi)
PRINT Sgn (Pi)
PRINT Sgn (0)
FINE
```

La console risponde con:

```
1
1
0
```

Peccato

Il peccato è una funzione intrinseca che calcola il seno di un angolo. L'angolo è specificato in radianti. La funzione seno sin x è una delle funzioni di base incontrate nella trigonometria (le altre sono cosecante, coseno, cotangente, secante e tangente). Sia θ un angolo misurato in senso antiorario dall'asse x lungo un arco del cerchio unitario. Allora $\sin \theta$ è la coordinata verticale del punto finale dell'arco. Come risultato di questa definizione, la funzione sin è periodica con periodo 2π . Secondo il teorema di Pitagora, anche $\sin \theta$ obbedisce all'identità $\sin^2 \theta + \cos^2 \theta = 1$. La sintassi standard del linguaggio Gambas per Sin è:

```
Valore = Sin (Angolo)
```

Esempio:

```
STATIC PUBLIC SUB Main
() PRINT Sin (Pi / 2)
FINE
```

La console risponde con:

```
1
```

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Sinh

Sinh è una funzione derivata che calcola il seno iperbolico di un numero. Lingua standard di Gambas la sintassi per Sinh è:

```
Valore = Sinh (numero)
```

L'esempio seguente utilizza la funzione Exp per calcolare il seno iperbolico di un angolo:

```
STATICO PUBBLICO SUB Principale ()
  DIM MyAngle AS Float
  DIM MyHSin AS Galleggiante 'Definisci l'angolo in radianti

  MyAngle = 1.3    'Calcola il seno iperbolico
  MyHSin = (Exp (MyAngle) Exp (1 * MyAngle)) / 2
  PRINT MyHSin
FINE
```

La console risponde con:

```
1.698382437293
```

Mq

Sqr è una funzione intrinseca che restituisce la radice quadrata di un numero. L'argomento numero può essere qualsiasi espressione numerica valida maggiore o uguale a 0 (non negativo). La sintassi del linguaggio Gambas è:

```
Valore = Sqr (numero)
```

Ecco un esempio:

```
STAMPA Sqr (2)
1.414213562373
```

L'esempio seguente utilizza la funzione Sqr per calcolare la radice quadrata di un numero:

```
STATICO PUBBLICO SUB Principale ()

  DIM MySqr AS Float
```

This product is (C) 2005 by John W. Rittinghouse. It is released to the Gamasas User Community under an Open Content License (OCC) and may not be distributed under any other terms or conditions without the written consent of the author.

Una guida per principianti a

```
MySqr = Mq (4)      'Restituisce 2
PRINT MySqr & "è il risultato di Sqr (4)"
MySqr = Mq (23)     'Restituisce
4.795831523313 PRINT MySqr & "è il
risultato di Sqr (23)" MySqr = Sqr (0)
                           'Restituisce 0
PRINT MySqr & "è il risultato di Sqr (0)"

'MySqr = Sqr (4) 'Genera un errore di runtime
"PRINT MySqr &" è il risultato di Sqr (4) "
FINE
```

La console risponde con:

```
2 è il risultato di Sqr (4)
4.795831523313 è il risultato di Sqr
(23)
0 è il risultato di Sqr (0)
```

Tan

Tan è una funzione intrinseca che calcola la tangente di un angolo. L'argomento angolo può essere qualsiasi espressione numerica valida che esprime un angolo in radianti. Tan prende un angolo e restituisce il rapporto tra due lati di un triangolo rettangolo. Il rapporto è la lunghezza del lato opposto all'angolo divisa per la lunghezza del lato adiacente all'angolo. Per convertire i gradi in radianti, moltiplica i gradi per pi greco / 180. Per convertire i radianti in gradi, moltiplicare i radianti per 180 / pi. La sintassi standard del linguaggio Gamasas è:

```
Valore = Tan (angolo)
```

L'esempio seguente utilizza la funzione Tan per restituire la tangente di un angolo:

```
STATICO PUBBLICO SUB Principale ()
DIM MyAngle AS Float
DIM MyCotangent AS Float

MyAngle = 1.3 ' Definisci l'angolo in radianti.
MyCotangent = 1 / Tan (MyAngle) 'Calcola cotangente.
                                         STAMPA MyCotangent
PRINT Tan (Pi /
4) END
```

La console risponde con:

Una guida per principianti a

0.277615646541

Una guida per principianti a

1

Tanh

Tanh è una funzione derivata che calcola la tangente iperbolica di un numero. La sintassi standard del linguaggio Gambas per Tanh è:

```
Valore = Tanh (numero)
```

Esempio:

```
STATIC PUBLIC SUB Main
    () PRINT Tanh (1)
FINE
```

La console risponde con:

```
0.761594155956
```

Funzioni matematiche derivate

Le funzioni matematiche non intrinseche (derivate) mostrate nella tabella seguente sono fornite in Gambas dalle funzioni matematiche intrinseche incorporate.

Funzioni matematiche derivate da Gambas

| Funzione | Equivalenti derivati |
|-----------------------------|--|
| Seno inverso | $\text{ASin}(X) = \text{Atn}(X / \text{Sqr}(X * X + 1))$ |
| Coseno inverso | $\text{ACos}(X) = \text{Atn}(X / \text{Sqr}(X * X + 1)) + 2 * \text{Atn}(1)$ |
| Seno iperbolico | $\text{Sinh}(X) = (\text{Exp}(X) \text{Exp}(X)) / 2$ |
| Coseno iperbolico | $\text{Cosh}(X) = (\text{Exp}(X) + \text{Exp}(X)) / 2$ |
| Iperbolico Tangente | $\text{Tanh}(X) = (\text{Exp}(X) \text{Exp}(X)) / (\text{Exp}(X) + \text{Exp}(X))$ |
| Iperbolico inverso Sine | $\text{ASinh}(X) = \text{Log}(X + \text{Sqr}(X * X + 1))$ |
| Iperbolico inverso Coseno | $\text{ACosh}(X) = \text{Log}(X + \text{Sqr}(X * X + 1))$ |
| Tangente iperbolica inversa | $\text{ATanh}(X) = \text{Log}((1 + X) / (1 - X)) / 2$ |
| Logaritmo in base 10 | $\text{Log10}(X) = \text{Log}(X) / \text{Log}(N)$ |

Di seguito è riportata una tabella che elenca altre funzioni matematiche derivate che possono essere create utilizzando l'estensione formule riportate nella tabella seguente:

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content license (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Altre funzioni matematiche derivate

| | | |
|-------------------------------|--|---|
| Secante | $\text{Sec}(X) = 1 / \text{Cos}(X)$ | This product is (C) 2005 by John W. Ritter. It is released to the Gambas Community under a Creative Commons License and may not be distributed under any other terms or conditions without the consent of the author. |
| Cosecante | $\text{Cosec}(X) = 1 / \text{Sin}(X)$ | |
| Cotangente | $\text{Cotan}(X) = 1 / \text{Tan}(X)$ | |
| Secante inversa | $\text{Arcsec}(X) = \text{Atn}(X / \text{Sqr}(X * X 1)) + \text{Sgn}(X 1) * (2 * \text{Atn}(1))$ | |
| Cosecante inverso | $\text{Arccosec}(X) = \text{Atn}(X / \text{Sqr}(X * X 1)) + (\text{Sgn}(X 1) * (2 * \text{Atn}(1)))$ | |
| Cotangente inversa | $\text{Arccotan}(X) = \text{Atn}(X) + 2 * \text{Atn}(1)$ | |
| Iperbolico Secante | $\text{HSec}(X) = 2 / (\text{Exp}(X) + \text{Exp}(X))$ | |
| Iperbolico Cosecante | $\text{HCosec}(X) = 2 / (\text{Exp}(X) \text{Exp}(X))$ | |
| Iperbolico Cotangente | $\text{HCotan}(X) = (\text{Exp}(X) + \text{Exp}(X)) / (\text{Exp}(X) \text{Exp}(X))$ | |
| Secante iperbolica inversa | $\text{HArcsec}(X) = \text{Log}((\text{Sqr}(X * X + 1) + 1) / X)$ | |
| Cosecante iperbolico inverso | $\text{HArcosec}(X) = \text{Log}((\text{Sgn}(X) * \text{Sqr}(X * X + 1) + 1) / X)$ | |
| Cotangente iperbolica inversa | $\text{HArccotan}(X) = \text{Log}((X + 1) / (X 1)) / 2$ | |
| Logaritmo in base N | $\text{LogN}(X) = \text{Log}(X) / \text{Log}(N)$ | |

Poiché Gambas non fornisce direttamente le funzioni incorporate per queste altre funzioni derivate, costruiremo un modulo che estenderà le capacità di Gambas fornendo tutte le funzioni elencate nella tabella sopra come esercizio finale in questo capitolo. Il nostro modulo si chiamerà Derived e sarai in grado di accedere a tutte le funzioni nel tuo codice usando il formato Derived.function_name dove function_name sarà una delle funzioni che costruiremo nella prossima sezione. Avvia Gambas e crea un'applicazione terminale chiamata MyMath. Crea una nuova classe, Class1 dall'IDE e rendila una classe di avvio. Quindi crea un nuovo modulo e chiamalo Derivato. Apri la nuova finestra del codice per il file Class1.class e inserisci quanto segue:

```
"File di classe Gambas
STATICO PUBBLICO SUB
Principale ()
    DIM myVal AS Float

    myVal = 51,5
    MENTRE myVal <> 0.0

        PRINT "Immettere un numero> 1 o immettere 0 per
        uscire:"; LINE INPUT myVal
        SE myVal = 0.0 THEN BREAK

        SE myVal <= 1.0 ALLORA
            PRINT "Il numero era <=
            1.0" CONTINUA
```

FINISCI SE

Una guida per principianti a

Una guida per principianti a

```
PRINT Derived.Sec (myVal) & "è secante"
PRINT Derived.CoSec (myVal) & "è cosecante"
PRINT Derived.CoTan (myVal) & "è cotangente"
PRINT
PRINT Derived.ArcSec (myVal) & "è arcsecant" PRINT
Derived.Arccosec (myVal) & "è arccosecant" PRINT
Derived.Arccotan (myVal) & "è arccotangent" PRINT
PRINT Derived.HSec (myVal) & "è una secante iperbolica"
PRINT Derived.HCoSec (myVal) & "è la cosecante
iperbolica" PRINT Derived.HCoTan (myVal) & "è una
cotangente iperbolica" PRINT
PRINT Derived.HArcSec (myVal) & "è arcsecant iperbolico" PRINT
Derived.HArcCoSec (myVal) & "è arccosecante iperbolico" PRINT
Derived.HArcCoTan (myVal) & "è arccotangente iperbolico" PRINT
PRINT Derived.LogN (myVal, 2) & "è LogN di:" & myVal & "in base 2" PRINT
"di myVal:" & myVal
WEND
END
```

Ora apri la finestra del codice per il file Derived.module e inserisci questo codice:

```
"Gambas Derived.module file
FUNZIONE PUBBLICA Sec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = 1.0 / Cos
    (x) RETURN risultato
FINE

FUNZIONE PUBBLICA CoSec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = 1.0 / Sin
    (x) RETURN risultato
FINE

FUNZIONE PUBBLICA CoTan (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = 1.0 / Tan
    (x) risultato RETURN
FINE

FUNZIONE PUBBLICA ArcSec (x AS Float) AS
    Float DIM risultato AS Float
    risultato = Atn (x / Sqr (x * x1)) + (Sgn (x) 1) *
    (2 * Atn (1)) RETURN risultato
FINE
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
FUNZIONE PUBBLICA ArcCosec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = Atn (x / Sqr (x * x)) + (Sgn (x) 1) *
    (2 * Atn (1)) RETURN risultato
FINE

FUNZIONE PUBBLICA ArcCotan (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = Atn (x) + (2 *
    Atn (1)) RETURN risultato
FINE

FUNZIONE PUBBLICA HSec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = 2.0 / (Exp (x) + Exp
    (x)) RETURN risultato
FINE

FUNZIONE PUBBLICA HCoSec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = 2.0 / (Exp (x) Exp
    (x)) RETURN risultato
FINE

FUNZIONE PUBBLICA HCoTan (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = (Exp (x) + Exp (x)) / (Exp (x) Exp
    (x)) RETURN risultato
FINE

'questa routine attualmente produce un errore in Gambas
'se si tenta di utilizzare la formula risultato = Log ((Sqr (x *
(x + 1)) + 1) / x)' perché la funzione Log funziona solo con
numeri non negativi 'per impostazione predefinita. Questa è una
soluzione alternativa per ottenere gli stessi risultati

FUNZIONE PUBBLICA HArcsec (x AS Float) AS
    Float Risultato DIM AS Float
    DIM Temp1 AS Float
    DIM Temp2 AS Float
    DIM Temp3 AS Float

    Temp1 = ((x * 1) * (x + 1)) +
    1 Temp2 = Sqr (Abs (Temp1))
    Temp3 = Log (Temp2)
    risultato = Temp3 *
    1

'questa chiamata genererà un errore perché la'
'funzione Sqr non funzionerà con valori negativi
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
'risultato = Log ((Sqr (x * (x + 1)) + 1) / x)

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

```
RETURN
risultato END

FUNZIONE PUBBLICA HARccosec (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = Log ((Sgn (x) * Sqr (x *
    x + 1) +1) / x) RETURN risultato
FINE

FUNZIONE PUBBLICA HARccotan (x AS Float) AS
    Float Risultato DIM AS Float
    risultato = Log ((x + 1) /
    (x1)) / 2 RETURN risultato
FINE

FUNZIONE PUBBLICA LogN (x AS Float, n AS Float) AS
    Float Risultato DIM AS Float
    risultato = Log (x) /
    Log (n) RETURN
    risultato
FINE
```

Quando si esegue il programma, verrà richiesto di immettere un numero> 1 o 0 per uscire. Inserisci 4 e dovresti vedere i seguenti risultati:

```
Immettere un numero> 1 o immettere 0 per uscire: 4
1.529885656466 è secante
1.321348708811 è cosecante
0.863691154451 è cotangente

0.85707194785 è arcosecante
0.801529994232 è arccosecante
2.896613990463 è arccotangente

0.036618993474 è secante iperbolica
0.036643570326 è la cosecante
iperbolica
1.000671150402 è cotangente iperbolica

1.472219489583 è arcosecante iperbolico
0,247466461547 è arccosecante iperbolico
0,255412811883 è arccotangente iperbolico

2 è LogN di: 4 in base 2
di myVal: 4
Immettere un numero> 1 o immettere 0 per uscire:
```

Ora dovresti essere in grado di eseguire quasi ogni tipo di operazione

Una guida per principianti a matematica necessaria per la maggior parte dei programmi. Hai assorbito molto materiale in

Una guida per principianti a

questo capitolo ed è ora di fare una pausa. Nel prossimo capitolo, andremo per coprire i concetti di base della programmazione orientata agli oggetti (OO) e mostrare quanto può essere potente Gambas quando si traggono vantaggio da queste funzionalità OO.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Capitolo 11 - Concetti orientati agli oggetti

Nel campo dell'informatica, la programmazione orientata agli oggetti (OOP) è considerata un paradigma di programmazione per computer. L'idea generale è che un programma per computer sia composto da una raccolta di singole unità (chiamate oggetti), ciascuna delle quali è in grado di ricevere messaggi, elaborare dati e inviare messaggi ad altre unità o oggetti. Ciò differisce dalla visione tradizionale della programmazione in cui un programma è visto come un elenco di istruzioni che vengono eseguite in ordine sequenziale dal computer.

I sostenitori della programmazione orientata agli oggetti hanno affermato che l'approccio offre ai programmatore maggiore flessibilità e facilita le modifiche ai programmi quando ce n'è bisogno. Il paradigma è molto popolare nelle aziende e negli enti governativi che conducono l'ingegneria e lo sviluppo del software come una cosa ovvia. Inoltre, molte persone credono che l'OOP sia più facile da imparare per i programmatore inesperti rispetto ad altri linguaggi come Java, C++, ecc. OOP, affermano, è più semplice da sviluppare e mantenere e che si presta a una migliore comprensione di situazioni complesse e procedure rispetto ad altri metodi di programmazione.

Ci sono quasi tanti critici del paradigma OO quanti sono i sostenitori. Alcuni degli argomenti più comuni sollevati dai critici sono che i linguaggi basati su OO come Smalltalk, Lisp e altri hanno avuto il loro progresso bloccato con l'avvento di linguaggi più recenti come C++, C# e Java. Linguaggi piccoli e interpretati come Tcl, Perl e Python hanno sviluppato un seguito abbastanza ampio ma non stanno facendo alcun progresso nello sviluppo di un vero paradigma OO. I critici hanno anche affermato che l'approccio orientato agli oggetti non affronta adeguatamente i futuri requisiti di elaborazione. Ritengono che i linguaggi orientati agli oggetti abbiano sacrificato la semplicità, proprio ciò che rende la programmazione accessibile a un pubblico più ampio. Java, ad esempio, è un'implementazione estremamente complessa dei concetti OO. Altri sostengono concetti come encapsulamento ed ereditarietà, originariamente pensati per "salvare" i programmatore da se stessi durante lo sviluppo del software, hanno fallito perché implementano proprietà globali. La modularità è un altro modo per mantenere l'ambito locale in modo che le persone possano comprendere meglio il codice. Questo era ciò che avrebbe dovuto realizzare l'incapsulamento. Oggi, il software open source sembra aver sviluppato un approccio praticabile al problema della manutenzione del codice, sebbene non

Una guida per principianti a
sia privo di problemi.

Gli oggetti hanno promesso di essere riutilizzati e la comunità dei
programmatori non ha visto

Una guida per principianti a

molto successo con quello. La capacità di mantenere un programma una volta che è stato scritto e di eseguire il debug localizzato e di fare sforzi di sviluppo molto più grandi sono anche ragioni citate come vantaggi significativi ottenuti con l'uso di linguaggi di programmazione orientati agli oggetti (OOPL). Negli anni '90, l'eccessivo ottimismo riguardo ai vantaggi della programmazione OO ha portato le aziende ad aspettarsi miracoli. Quando gli sviluppatori di software non sono stati in grado di mantenere gli oltraggiosi piani aziendali basati su quelle promesse, il 2001 ha visto le aziende cadere come un domino, lasciando l'intero settore tecnologico in recessione. Aziende defunte disseminavano la Silicon Valley come tante ossa sbiancate in un deserto.

Fortunatamente, Gambas adotta un approccio molto pragmatico all'implementazione dei concetti OO, sfruttando ciò che funziona e, a differenza del C++, evitando ciò che non funzionerà mai bene. L'implementazione di Gambas è un elegante compromesso tra la vera teoria OO e le realtà aziendali con cui i programmatore devono confrontarsi oggi.

L'OOP è spesso chiamato paradigma piuttosto che uno stile o un tipo di programmazione per sottolineare il punto che L'OOP può cambiare il modo in cui il software viene sviluppato cambiando il modo in cui i programmatore e gli ingegneri del software pensano al software. Gran parte dell'evoluzione di qualsiasi lingua dipende dal sistema operativo che supporta quella lingua. Con l'avvento di Linux e l'enorme espansione del supporto per il software open source, il futuro sembra molto luminoso per Gambas. Ora, diamo uno sguardo ad alcuni dei concetti basilari di OO supportati da Gambas.

Fondamenti di programmazione orientata agli oggetti

La programmazione orientata agli oggetti si basa su diversi concetti chiave: oggetti, astrazione dei dati, encapsulamento, polimorfismo ed ereditarietà. Una caratteristica distintiva di OOP sono i metodi utilizzati per la gestione dei tipi di dati. I dati del tipo di oggetto sono generalmente richiesti per soddisfare i vincoli definiti dal programmatore. Un tipo di dati limitato a un oggetto specifico è considerato un tipo di dati privato. Questo tipo di vincolo imposto dal linguaggio che limita i tipi di dati a oggetti specifici costringe i programmatore (in base alla progettazione) a codificare senza fare affidamento sulla creazione di una lista di variabili visibili a ogni oggetto nel programma (variabili globali). Le variabili possono essere utilizzate ed eliminate all'interno dell'oggetto o della classe in cui sono necessarie e non influenzano altre parti di un programma. Le azioni

Una guida per principianti a intraprese dal programma in cui vengono utilizzate tali variabili sono denominate metodi. Nei linguaggi più tradizionali, i metodi equivalgono a funzioni o subroutine. In OOP,

Una guida per principianti a

I linguaggi orientati agli oggetti forniscono meccanismi interni per garantire che variabili e metodi siano visibili e disponibili solo nell'ambito della classe o il metodo in cui vengono dichiarati. Anche se alcune variabili e metodi sono progettati per essere visibili a tutti gli altri metodi o classi di un progetto. In questi casi, vengono indicati come variabili e metodi pubblici. In Gambas, le variabili e i metodi pubblici hanno un ambito globale rispetto alla classe in cui sono dichiarati. Le variabili destinate ad essere visibili solo ai metodi in cui sono dichiarate come private all'interno di una data classe vengono create con la parola chiave DIM in Gambas.

Oggetti

Oggetti sono costituiti da tutti i dati e le funzionalità che esistono all'interno di unità distinte in un programma per computer in esecuzione (in esecuzione). Gli oggetti sono il fondamento della modularità e della struttura in un OOPL. Hanno due attributi di base: sono autocontenuti e sono identificabili in modo univoco. La presenza di questi due attributi consente a tutti i vari componenti del programma di correlare le variabili in un programma con vari aspetti del mondo reale di un problema per il quale un programma intende risolvere.

Astrazione dei dati

Astrazione dei dati L'astrazione dei dati è una metodologia utilizzata negli OOPL che consente a un programmatore di specificare come viene utilizzato un oggetto dati. L'astrazione consente a un programmatore di isolare gli oggetti dati dai dettagli sottostanti di come vengono creati i dati (di solito utilizzando oggetti dati ancora più primitivi). In altre parole, il file l'uso di un oggetto dati, composto da uno o più oggetti dati primitivi, consente a un programmatore di strutturare il proprio codice per utilizzare oggetti dati che operano su dati astratti.

Gli oggetti dati che sono costituiti da uno o più oggetti dati primitivi sono chiamati oggetti di dati composti. Con l'astrazione, il programma può utilizzare i dati senza fare ipotesi su di essi (cioè i dettagli sottostanti di come o da dove provengono i dati) che non siano necessari per eseguire una determinata attività. Viene definita una rappresentazione concreta dei dati indipendentemente da tutti i programmi che utilizzano i dati. L'interfaccia tra i dati concreti e astratti è un insieme di procedure, chiamate costruttori, che implementano dati astratti in termini di rappresentazione concreta. Anche i metodi (subroutine e funzioni) possono essere astratti nel paradigma

Una guida per principianti a
OOP.

Una guida per principianti a

Incapsulamento

Incapsulamento Garantisce che gli utenti di un oggetto non possano modificare l'interno stato dell'oggetto in modi inaspettati; solo i metodi definiti internamente dell'oggetto possono accedere al suo stato (o attributi). Ogni oggetto espone un'interfaccia che specifica come altri oggetti possono interagire con esso. Altri oggetti non conosceranno i metodi interni di questo oggetto e dovranno quindi fare affidamento sull'interfaccia di questo oggetto per interagire con esso.

Polimorfismo

Polimorfismo significa la capacità di assumere più di una forma. In OOPL, essosi riferisce alla capacità di un linguaggio di programmazione di elaborare gli oggetti in modo diverso a seconda del tipo di dati o della classe. Più specificamente, il polimorfismo è la capacità di ridefinire i metodi per le classi derivate. Un'operazione può mostrare comportamenti diversi in circostanze diverse. Il comportamento dipende dal tipo di dati utilizzato nell'operazione. Ad esempio, data una classe base denominata shape, il polimorfismo consente al programmatore di definire diversi metodi di calcolo dell'area per un numero qualsiasi di classi derivate, come cerchi, rettangoli e triangoli. Indipendentemente dalla forma di un oggetto, l'applicazione del metodo dell'area restituirà i risultati corretti. Il polimorfismo è ampiamente utilizzato negli OOPL durante l'implementazione dell'ereditarietà ed è considerato un requisito di qualsiasi vero OOPL.

Eredità

Eredità è il processo mediante il quale gli oggetti possono acquisire le proprietà degli oggetti che esistono in un'altra classe. In Gambas, tutti i controlli della casella degli strumenti ereditano le proprietà della classe denominata Control. In OOP, l'ereditarietà fornisce riusabilità ed estensibilità (aggiungendo funzionalità aggiuntive a una classe esistente senza la necessità di modificarla). Ciò si ottiene derivando una nuova classe da una classe esistente e apportando modifiche alla nuova istanza di quella classe. La nuova classe avrà combinato le caratteristiche di entrambe le classi. L'ereditarietà consente di definire e creare oggetti che sono implementazioni specializzate di oggetti già esistenti. Possono condividere (ed estendere) il comportamento dell'oggetto esistente senza doverlo reimplementare.

Una guida per principianti a

L'approccio di Gambas all'OOP

Un programma OO è generalmente progettato identificando tutti gli oggetti che esisteranno in un sistema. Il codice che esegue effettivamente il lavoro è irrilevante per il fileoggetto dovuto all'incapsulamento. La sfida più grande in OOP è progettare un sistema di oggetti che sia comprensibile e gestibile. Va notato che ci sono paralleli distinti tra il paradigma orientato agli oggetti e la teoria dei sistemi tradizionali. Mentre OOP si concentra sugli oggetti come unità in un sistema, la teoria dei sistemi si concentra sul sistema stesso. Da qualche parte nel mezzo, si possono trovare modelli di progettazione del software o altre tecniche che utilizzano classi e oggetti come elementi costitutivi per componenti più grandi. Tali componenti possono essere viste come un passaggio intermedio dal paradigma orientato agli oggetti verso i modelli più pragmaticamente orientati della teoria dei sistemi. Questo sembra essere l'approccio di Gambas all'OOP. È una soluzione elegante e pragmatica che prende il meglio da entrambi gli approcci e rende la programmazione facile e semplice.

Gambas ha cercato di mantenere la semplicità in prima linea in un approccio che sfrutta i concetti di OO e compromette con la teoria dei sistemi dove fabuon senso. A questo proposito, Gambas ha adottato un approccio di classe all'OOP. L'approccio OOP basato sulla classe è un paradigma OOP in cui il concetto di classe è centrale. In questo approccio, l'incapsulamento impedisce agli utenti di rompere le invarianti della classe. Le invarianti di classe vengono stabilite durante la costruzione e vengono costantemente mantenute tra le chiamate ai metodi pubblici. È possibile interrompere temporaneamente l'invarianza di classe tra chiamate a metodi privati, sebbene non sia incoraggiata. A volte, è utile perché consente di modificare l'implementazione di una classe di oggetti per aspetti non visibili o esposti all'interno dell'interfaccia della classe senza avere un impatto sul codice utente che potrebbe aver implementato le funzionalità di quella classe.

La definizione di encapsulamento in un paradigma OOP basato sulla classe si concentra sul raggruppamento e sul confezionamento delle informazioni correlate (coesione). I linguaggi OOP normalmente non offrono restrizioni di sicurezza formali allo stato dell'oggetto interno. L'utilizzo di un metodo di accesso è una questione di convenzione per la progettazione dell'interfaccia. L'ereditarietà viene in genere eseguita raggruppando gli oggetti in classi e definendo le classi come estensioni di classi esistenti. Questo raggruppamento di classi a volte è organizzato in albero ostrutture reticolari, che spesso riflettono un insieme comune di comportamenti.

Una guida per principianti a
Classi di Gambas

In Gambas, puoi definire il tuo classi. Ricorda che una classe è

Una guida per principianti a

semplicemente modello che viene utilizzato dal programma in fase di esecuzione. La classe stessa non lo è usato direttamente. Per usare una classe, devi creare una variabile dichiarazione del tipo di dati della classe, creare una copia della classe per il programma da utilizzare, quindi istanziare la copia. La copia istanziata è chiamata oggetto. Poiché una classe è un tipo di dati reale, la dichiarazione dell'oggetto assomiglia a qualsiasi altra normale dichiarazione di variabile:

```
nome_oggetto AS nome_classe
```

Creare un'istanza di un oggetto significa creare una nuova istanza del modello e assegnarlo quell'istanza il nome sul lato sinistro del segno di uguale. In GB, ciò si ottiene utilizzando la parola chiave NEW:

```
nome_oggetto = NUOVO nome_classe
```

Quando object_name viene istanziato come una nuova copia del modello definito da *Nome della classe*, il linguaggio di programmazione consente a object_name di ereditare tutti i metodi e gli attributi di Class_name. Quando il nuovo oggetto viene creato, richiama automaticamente un costruttore per inizializzarsi con le variabili predefinite. È importante notare che l'attuale versione di Gambas non supporta il concetto di invio virtuale. In altre parole, quando si richiama un metodo (o si imposta / si ottiene un valore di proprietà) per un riferimento a un oggetto utilizzando la notazione del punto (cioè, class.object.method), il metodo che viene effettivamente eseguito è il metodo appartenente al tipo di dati di la variabile che possiede il riferimento all'oggetto, non il metodo del tipo di dati reale dell'oggetto. In Gambas2, questo problema è stato corretto.

Programma di esempio: contatti

Per illustrare l'uso di una classe in Gambas, creeremo un programma di esempio che implementa un libro di contatti. Sarà abbastanza semplice ma sarà utile per mantenere i contatti. Per iniziare, crea un nuovo progetto denominato Classes e quando arrivi all'IDE, crea una classe (non una classe di avvio) denominata Contact. È qui che inizieremo con il nostro programma, costruendo una definizione di classe per un contatto.

La classe Contact

```
"Gambas Contact.class file
PUBLIC FirstName AS String
```

Una guida per principianti a

```
PUBLIC LastName AS String  
PUBLIC Initial AS String  
PUBLIC Suffix AS String  
PUBLIC Street1 AS String
```

Una guida per principianti a

```
PUBLIC Street2 AS String  
PUBLIC City AS String  
PUBLIC State AS String  
PUBLIC Zip5 AS String  
PUBLIC Zip4 AS String  
PUBLIC Areacode AS String  
PUBLIC Prefix AS String  
PUBLIC Last4 AS String  
PUBLIC RecordID AS Integer
```

Vogliamo che la nostra classe fornisca due metodi, uno per recuperare i nostri dati di contatto dal file viene memorizzato in e uno per inserire i dati in quel file. In modo appropriato, i due metodi verranno denominati Contact.GetData e Contact.PutData.

Metodo Contact.GetData

Cominciamo con GetData , che viene annullato come funzione perché prenderà una variabile di raccolta come parametro e restituirà una raccolta piena di dati dal file contacts.data se tali dati esistono. Dichiareremo le variabili locali all'interno dei nostri metodi utilizzando la parola chiave DIM. Nota che potremmo dichiarare queste variabili come PRIVATE al di fuori delle definizioni dei metodi e le renderebbe visibili solo ai metodi all'interno della classe contact. Tuttavia, le dichiarazioni locali al metodo stesso restringono ulteriormente l'ambito e rendono l'intero file di classe più pulito in apparenza dal punto di vista dell'utente. Vedranno tutti i metodi e le variabili pubblici e nient'altro.

La variabile di raccolta viene passata come parametro dalla chiamata al costruttore in Form1.class quindi non è necessario dichiararlo separatamente.La funzione restituirà una variabile di raccolta, quindi aggiungiamo AS Collection alla fine della dichiarazione della funzione. Mostreremo come la variabile di raccolta cTheData viene passata a questa funzione quando viene chiamata dal file Form1.class.

```
FUNZIONE PUBBLICA GetData (cTheData AS Collection) AS Collection  
  
'questa stringa var conterrà ogni riga di dati letta dal file DIM  
sInputLine AS String  
  
'abbiamo bisogno di un handle di file e due variabili di stringa  
per la creazione di nomi di file DIM hFileIn AS File  
Nome file DIM AS  
String DIM fullname AS  
String
```

This product is (C)2003 by John W. Gambs User Company under an Open Content License (OCL) and cannot be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

'questa variabile di classe viene utilizzata per convertire i dati della stringa che abbiamo letto' dal file in un record di contatto.
Dichiariamo la variabile qui e

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'ne creerà un'istanza di seguito
DIM MyContactRecord AS Contact

'un numero intero locale var per tenere traccia di quanti record
abbiamo aggiunto DIM RecordNum AS Integer

'questa variabile prenderà la stringa letta e convertirà tutti i campi
che sono commadelimitati in un array usando l'istruzione split
DIM arsInputFields AS String []

"questo contatore viene utilizzato per scorrere la matrice e
assegnare valori" ai record del contatto
Contatore DIM AS Integer

filename = "contacts.data" 'hardcoded perché non cambierà

'useremo la directory corrente in cui si trova l'app per creare il
percorso completo fullpath = Application.Path & / filename

'ora, controlla se il file esiste già, in tal caso aprilo per
LEGGERE. IF Exist (fullpath) THEN
APRI fullpath PER LEGGERE COME #hFileIn

'inizia con il
record 1 RecordNum =
1

'eseguiremo un ciclo su ogni riga di dati fino a raggiungere la fine
del file MENTRE NON Eof (hFileIn)
    "ogni contatto aggiunto alla raccolta deve essere istanziato"
    ogni volta che un record viene aggiunto alla raccolta.
    Tuttavia, PUOI riutilizzare la stessa classe var per ogni
    reinstanziazione.MyContactRecord = NUOVO contatto 'qui è dove
    si rinnova

'legge la riga dal file e la memorizza nella stringa var
sInputLine LINE INPUT #hFileIn, sInputLine

'utilizzato per il debug della
console PRINT "Lettura:" &
sInputLine

'ora, riempi il nostro array di stringhe usando Split per
convertire la riga' dei dati di stringa letti in campi in un
array
arsInputFields = Split (sInputLine, ",")

'per il debug solo per mostrare che tutti i campi vengono
convertiti correttamente Contatore FOR = da 0 a 14
PRINT "Caricamento in corso:" & arsInputFields [contatore]
```

Una guida per principianti a
IL PROSSIMO

'ora, inseriamo i dati del campo nella nostra classe di contatto
locale var

Una guida per principianti a

```
'e aggiungerà il record del contatto alla raccolta dei contatti
MyContactRecord.FirstName = arsInputFields [0]
MyContactRecord.Initial = arsInputFields [1]
MyContactRecord.LastName = arsInputFields [2]
MyContactRecord.Suffix = arsInputFields [3]
MyStreetRecordselds [3] MyContactRecordseldselds Città
arsInputFields [6]
MyContactRecord.State = arsInputFields [7]
MyContactRecord.Zip5 = arsInputFields [8]
MyContactRecord.Zip4 = arsInputFields [9]
MyContactRecord.Areacode = arsInputFields [10]
MyContactRecputseceldsecelds12 RecordID = CInt
(arsInputFields [13]) MyContactRecord.Deleted = CBool
(arsInputFields [14])

'per l'output di debug sulla console
PRINT "Aggiunta di dati per il record:" & RecordNum & "";
STAMPA MyContactRecord.FirstName & "" & MyContactRecord.LastName

'a tutti i campi dei record vengono assegnati dati, quindi
aggiungiamo alla raccolta cTheData.Add (MyContactRecord, CStr
(RecordNum))

'per l'output di debug sulla console
PRINT cTheData [CStr (RecordNum)]. LastName & "stored".

'incrementa il nostro contatore di record e cancella la
variabile sInputLine INC RecordNum
sInputLine = ""

WEND 'loop per più record

'chiude il file quando non esistono più record nel file
CHIUDI # hFileIn
ALTRO 'nessun file trovato, invia un messaggio all'utente
    Message.Info ("Nessun file dei contatti.data presente.",
    "OK")
FINISCI SE

'solo per il debug, elencheremo tutti i record nella raccolta
PRINT "Ecco cosa è memorizzato in cTheData:"
PER OGNI MyContactRecord IN cTheData
    STAMPA MyContactRecord.FirstName & "" & MyContactRecord.LastName
AVANTI
RETURN cTheData 'restituisce la raccolta (ora piena) con i
dati FINE 'di GetData
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under the Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

THIS DOCUMENT IS PROVIDED UNDER THE TERMS OF THE UNLICENSE. YOU MAY USE IT FOR ANY PURPOSE UNLESS RESTRICTED BY APPLICABLE LAW. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Una guida per principianti a

Metodo Contact.PutData

Il reciproco routine per GetData è PutData, dove memorizzeremo la raccolta di dati che è considerata la più attuale. Le modifiche o le modifiche apportate verranno riflesse in questa operazione di salvataggio. Salveremo la data in un file, memorizzando il record come una stringa delimitata dalla commessa. Ogni riga di output rappresenterà un record di contatto. Il metodo PutData viene dichiarato come subroutine poiché non restituisce una variabile. Richiede un parametro, l'oggetto della raccolta che desideriamo memorizzare. cTheData è la raccolta passata a questa subroutine.

```
PUBLIC SUB PutData (cTheData AS Collection)

'questa stringa var conterrà ogni riga di dati letta dal file DIM
sInputLine AS String

'abbiamo bisogno di un handle di file e due variabili di stringa
per la creazione di nomi di file DIM hFileOut AS File
Nome file DIM AS
String DIM fullname AS
String

'questa variabile di classe viene utilizzata per convertire i
dati della stringa in' un record di contatto. Dichiariamo la
variabile qui e
'ne creerà un'istanza di seguito
DIM MyContactRecord AS Contact

'un numero intero locale var per tenere traccia di quanti record
abbiamo aggiunto DIM RecordNum AS Integer

'crea un'istanza di un nuovo record vuoto. Si noti che poiché non
stiamo "aggiungendo questo record a una raccolta, non è necessario
rinnovarlo" ripristinandolo ogni volta.
MyContactRecord = NUOVO contatto

'inizia dal record
1 RecordNum = 1

'abbiamo un nome di file hardcoded quindi abbiamo solo bisogno del
percorso del programma' perché memorizzeremo i dati in cui risiede il
programma.
filename = "contacts.data"
fullname = Application.Path & / filename

'dobbiamo specificare WRITE CREATE quando apriamo il file. Se il
file "esiste, verrà aperto per la scrittura. Se non esiste, creeremo
un nuovo file e vi scriveremo.
```

Una guida per principianti a
OPEN fullpath FOR WRITE CREATE AS #hFileOut

'ora, passiamo in rassegna ogni record nella raccolta. Nota che noi

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is distributed to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

'usa il costrutto FOR EACH per farlo.
PER OGNI MyContactRecord IN cTheData

'concateniamo ogni campo all'ultimo, inserendo virgole'
da utilizzare come delimitatori di campo per l'operazione
di lettura

```
sInputLine = sInputLine & MyContactRecord.FirstName & ","
sInputLine = sInputLine & MyContactRecord.Initial & ","
sInputLine = sInputLine & MyContactRecord.LastName & ","
sInputLine = sInputLecord & Street1 & "," sInputLine =
sInputLine & MyContactRecord.Street2 & "," sInputLine =
sInputLine & MyContactRecord.City & "," sInputLine =
sInputLine & MyContactRecord.State & "," sInputLine = sInput5
& " = sInputLine & MyContactRecord.Zip4 & "," sInputLine =
sInputLine & MyContactRecord.Areacode & "," sInputLine =
sInputLine & MyContactRecord.Prefisso & "," sInputLine =
sInputLine & MyContactRecord.Last4 & "," sInputLine =
sInputLine & MyContactRecord.RecordID & "," sInputLine =
sInputLine & CStr (MyContactRecord.Deleted) & "','" solo per il
debug, per mostrare cosa viene memorizzato
PRINT "Memorizzazione:" & sInputLine
```

'questa chiamata è in realtà ciò che inserisce i dati nel
file **PRINT #hFileOut, sInputLine**

'cancella la nostra stringa var da riutilizzare nella prossima
iterazione **sInputLine = ""**

IL PROSSIMO "iterazione del ciclo per enumerare
gli oggetti" ora li ha scritti tutti, quindi
chiudi il file **CHIUDI # hFileOut**
RITORNO 'senza restituire una variabile, quindi una
subroutine **FINE**

File Form1.class

Salvare il file Contact.class e chiuderlo. La nostra definizione della classe di contatto è completa, quindi ora dovremo creare un modulo denominato Form1. Form1 fungerà da interfaccia principale tra l'utente e il programma e i suoi dati. Ecco come apparirà il nostro modulo finito:

Una guida per principianti a



Figura 74 Programma Contatti finiti.

Il nostro programma consentirà a un utente di aggiungere, modificare o eliminare un contatto e memorizzare i dati in un file di dati. I dati di contatto vengono gestiti nel nostro programma come una raccolta di oggetti di tipo Contact, che abbiamo definito nel file Contact.class sopra. Consentiremo all'utente di spostarsi al primo elemento nella raccolta, scorrere agli elementi successivi o precedenti, andare all'ultimo elemento, aggiornare i dati nel modulo corrente, salvare i dati correnti su file e uscire. Tutto sommato, fa la maggior parte delle basi necessarie per la gestione dei contatti. Aggiungeremo anche una funzione di ricerca (utilizzando un modulo) che troverà la prima istanza di un contatto cercando un cognome. Sorprendentemente, per realizzare tutto questo ci vuole pochissimo codice in Gambas!

Ora, diamo un'occhiata al programma che utilizzerà la classe contact. Il programma richiederà quattro variabili pubbliche, come mostrato di seguito. La nostra routine del costruttore lo farà cercare e carica automaticamente il file di contatto predefinito, che abbiamo chiamato contacts.data. Non è necessario che l'utente conosca o utilizzi questo nome di file, quindi è codificato nei metodi in cui viene utilizzato. Se la routine del costruttore _new () non può caricare un file, creerà un record fittizio in modo che l'utente non veda mai una schermata di avvio vuota.

```
'File di classe Gambas Form1

'Contatti rappresenterà la raccolta di record per il libro addr Contatti
PUBBLICI come raccolta

'rappresenta un singolo record di contatto utilizzato da tutte
le subroutine Contatto PUBBLICO Registra come contatto

'indice che punta al record corrente
RecordPointer PUBBLICO AS Integer
```

This product is (C) 2005 or later. All rights reserved. It is released to the public domain under the terms of the GNU General Public License (GPL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
'avremo bisogno di una stringa var da usare quando l'utente cerca per
cognome **SearchKey** PUBLIC AS String

Costruttore Form1

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'la routine del costruttore chiamata quando viene chiamata
form1_open () PUBLIC SUB _new ()

'istanziare la raccolta che conterrà i nostri record di indirizzo
Contatti = NUOVA Collezione

'crea un'istanza di un nuovo contatto (record)
vuoto ContactRecord = NUOVO contatto

'solo per il debug
PRINT "Richiamo di GetData ..."

'chiama il metodo GetData definito nella definizione della classe di
contatto ContactRecord.GetData (Contatti)

'imposta il puntatore del record sull'ultimo record trovato nel set di
dati RecordPointer = Contacts.Count

'solo per il debug
PRINT "In _new (), rtn fm GetData con:" & RecordPointer & "records"

"il costruttore predefinito garantirà che esista almeno un
record" se la proprietà count della raccolta di contatti mostra
<1 record.
SE RecordPointer <1
    ALLORA 'aggiungi i
    nostri dati fittizi
    ContactRecord.FirstName = "John"
    ContactRecord.Initial = "Q"
    ContactRecord.LastName = "Pubblico"
    ContactRecord.Suffix = "II"
    ContactRecord.Street1 = "12345 Gambas Drive"
    ContactRecord.Street2 = "Apt 101"
    ContactRecord.City = "Ballerino"
    ContactRecord.State = "TX"
    ContactRecord.Zip5 = "77929"
    ContactRecord.Zip4 = "0101"
    ContactRecord.Areacode = "888"
    ContactRecord.Prefix = "666"
    ContactRecord.Last4 = "4444"
    ContactRecord.RecordID = 1
    ContactRecord.Deleted = FALSE
    RecordPointer = ContactRecord.RecordID
    Contacts.Add (ContactRecord, CStr (ContactRecord.RecordID))
ENDIF

'controlla la nostra raccolta per assicurarti
che i record esistano IF Contacts.Exist (CStr
(RecordPointer)) THEN
```

Una guida per principianti a

' solo per il debug

Una guida per principianti a

```
PRINT "I dati del record esistono per" & CStr (RecordPointer) &
"record."  
  
'questo ciclo for è solo per il
debug PER OGNI contatto, registra
nei contatti
    STAMPA ContactRecord.FirstName & "" & ContactRecord.LastName
NEXT
ALTRO  
  
'se non ci sono record, metti un MessageBox per informare l'utente
    Message.Info ("I record non esistono!", "Ok")
ENDIF
END
```

Form_Open Subroutine

Nostro il programma aprirà automaticamente il modulo utilizzando la subroutine Form_Open () di seguito. Poiché Form1.form è definito come un form di avvio, non è necessario chiamare il metodo Form1.Show. Tutto ciò che faremo qui è impostare la didascalia da visualizzare nella parte superiore della finestra quando il programma viene eseguito e ottenere il record puntato dal puntatore del record corrente. In questo caso, il puntatore dovrebbe puntare all'ultimo record caricato dal file dei contatti dopo che il flusso del programma ritorna dal costruttore.

```
PUBLIC SUB Form_Open ()
    'imposta la didascalia
    del programma
    Form1.Caption = "Gestione contatti"  
  
'inizializza con il record puntato dopo la chiamata del
    costruttore' che dovrebbe essere l'ultimo record della
    raccolta ContactRecord = Contatti [CStr (RecordPointer)]
    'questa subroutine aggiornerà tutti i campi con i dati del record
    corrente UpdateForm
FINE
```

Aggiunta di controlli a Form1.Form

È ora di mettere tutti i controlli sul modulo e dare loro i nomi. Useremo nove ToolButtons, un Button normale, 14 etichette e 13 TextBoxes. Dovremo anche usare nove icone, che possiamo ottenere dalla cartella usr / share / icons / defaultkde / 32x32 (NOTA: il tuo sistema potrebbe essere diverso, quindi naviga per trovare dove sono memorizzate le icone, se necessario). Le icone che ho usato per questo programma sono le seguenti:

Una guida per principianti a

| | | | | |
|-----------------|------|-------------|---------------|----------------|
| | | | | |
| Inizio dei dati | Prec | Il prossimo | Fine dei dati | Cancella campi |

| | | | |
|----------------|------------------|------------------|---------------|
| | | | |
| Aggiorna campi | Aggiungere nuova | Elimina corrente | Salva su file |

I ToolButtons

I nove ToolButtons che utilizzano queste icone nella loro proprietà Picture sono di nome:

| | | |
|-----------------------|------------------------|----------------------|
| FirstRecordBtn | PrevRecordBtn | NextRecordBtn |
| LastRecordBtn | ClearBtn | Aggiornare |
| AddBtn | DeleteRecordBtn | SaveAllBtn |

Creare nove ToolButtons e disponili nel modulo come mostrato nella figura all'inizio di questo capitolo. Assegna i nomi mostrati sopra ai pulsanti degli strumenti e imposta le proprietà delle immagini dei rispettivi pulsanti sui nomi delle icone che hai scelto per il tuo programma, in base a ciò che fornisce la distribuzione del tuo sistema. Lo scenario peggiore è che puoi utilizzare l'editor di icone integrato in Gambas per creare le tue icone.

Il pulsante Esci

Dopo aver completato i ToolButtons, aggiungeremo il pulsante normale come il nostro pulsante "Esci". Selezionare un pulsante dalla casella degli strumenti e posizionarlo sul modulo nell'angolo inferiore destro, il più vicino possibile alla Figura 74. Impostare la proprietà Text per questo pulsante su "Esci". Finché siamo qui, possiamo codificare la routine di uscita del programma, che verrà chiamata quando si fa clic su questo pulsante. Fai doppio clic sul pulsante Esci e sarai inserito nell'editor del codice. Aggiungi questo codice:

```

PUBLIC SUB QuitBtn_Click ()

    'richiesta di salvare i dati prima di uscire
    SELEZIONA Message.Question ("Salva prima di uscire?", "Sì", "No",
    "Annulla")
    CASO 1
        'se l'utente dice di sì,

```

Una guida per principianti a
salviamo **SaveAllBtn_Click**

CASO 2 'non salviamo, chiudiamo

This product is (C) 2005 by J. L. V. G. All rights reserved. It is released under the terms of the GNU General Public License (GPL), Version 2, or at the express written consent of the author.

Una guida per principianti a

CASO 3 'hanno annullato, quindi torniamo al programma **RITORNO**

CASO ALTRO 'non facciamo altro che

chiudere **FINE SELEZIONA**

ME.Chiudi 'qui chiudiamo la finestra **FINE** 'il programma si interrompe

Aggiunta di etichette e caselle di testo

Probabilmente la parte più noiosa dello sviluppo di questo programma è la successiva, l'aggiunta dei controlli Label e TextBox al form. Prova a disporli come vedi nella Figura 74 sopra. Per i controlli Etichetta, puoi prendere i nomi delle variabili di default forniti da Gambas per tutte le etichette tranne quella in fondo al modulo dove visualizzeremo il nostro Record n di n dati. Quell'etichetta dovrebbe essere denominata StatusLabel. I campi della casella di testo, come mostrato da in alto a sinistra a in basso a destra nel modulo, dovrebbero essere denominati come segue:

```
FirstName  
MI  
LastName  
Suffix  
Street1  
Street2  
City  
State  
Zip5  
Zip4  
AreaCode  
DialPrefix  
DialLast4
```

Si spera che i nomi debbano essere autoesplicativi (cioè, codice autodocumentante) e che tu non abbia problemi a creare il modulo. Fai attenzione a usare le maiuscole e le minuscole e digita i nomi esattamente come mostrato sopra, altrimenti rintracerai variabili non dichiarate quando proverai a eseguire il programma. A questo punto, il tuo modulo, in modalità progettazione, dovrebbe essere simile a questo:

This product is (C) 2005 by John W. K. Gammie, all rights are reserved. It is released to the Gambas User Community under an open source license (OCL) and may not be distributed under any other terms or conditions without express written consent of the author.

Una guida per principianti a

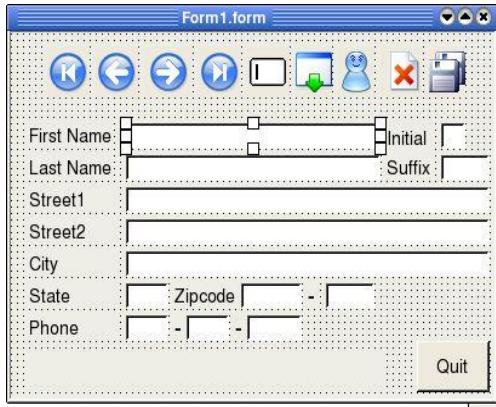


Figura 75 Form1 visto in modalità progettazione.

Subroutine UpdateForm ()

La routine UpdateForm () verrà chiamata ogni volta che i dati nel modulo sono stati modificati e devono essere aggiornati in modo che l'utente vedrà gli aggiornamenti man mano che si verificano. Questa routine inserisce ogni campo del modulo con i dati correnti trovati nella raccolta in base al puntatore del record corrente. Inoltre, l'etichetta StatusLabel viene aggiornata con il numero di record corrente di n record nella raccolta da qui. Esaminiamo il codice:

```
PUBLIC SUB UpdateForm ()  
  
'controlla per assicurarti che ci siano record prima di  
fare qualsiasi cosa IF Contacts.Exist (CStr  
(RecordPointer)) THEN  
  
'abbiamo trovato i record, quindi ora passiamo al record al puntatore  
corrente ContactRecord = Contatti [CStr (RecordPointer)]  
  
'solo per il debug  
PRINT "In UpdateForm con RecordPointer di:" & RecordPointer  
  
'assegna Textbox.Text valori qualunque sia memorizzato nel record del  
contatto FirstName.Text = ContactRecord.FirstName  
MI.Text = ContactRecord.Initial  
LastName.Text = ContactRecord.LastName  
Suffix.Text = ContactRecord.Suffix  
Street1.Text = ContactRecord.Street1  
Street2.Text = ContactRecord.Street2  
City.Text = ContactRecord.City  
State.Text = ContactRecord.State Zip5  
.Text = ContactRecord.Zip5 Zip4.Text =  
ContactRecord.Zip4 AreaCode.Text =  
ContactRecord.Areacode
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the Gamba User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
DialPrefix.Text = ContactRecord.Prefix
DialLast4.Text = ContactRecord.Last4

'aggiorna il campo dello stato del record corrente con i dati
correnti StatusLabel.Text = "Record:" & CStr (RecordPointer)
StatusLabel.Text = StatusLabel.Text & " of" & CStr (Contacts.Count)

'chiama il metodo Refresh per ridipingere il contenuto
del modulo Form1.Refresh

'seleziona il contenuto del campo FirstName
FirstName.Select

'imposta lo stato attivo del cursore sul campo FirstName
FirstName.SetFocus
ALTRO 'non esistono
record! 'solo per il
debug
PRINT "In UpdateForm, Record:" & RecordPointer & "not found".
FINISCI SE
'solo per il debug
PRINT "Uscita da UpdateForm
..." END
```

Tasti degli strumenti di codifica: primo, precedente, successivo e ultimo

Il primo ToolButton che codificheremo è quello che ci porterà all'inizio dei dati. Fai doppio clic sul primo ToolButton e dovrassi essere portato alla finestra del codice dove attende la routine FirstRecordBtn_Click (). Aggiungi questo codice:

```
PUBLIC SUB FirstRecordBtn_Click
() 'passa al primo record
RecordPointer = 1

'recuperare quel record dalla raccolta
ContactRecord = Contatti [CStr (RecordPointer)]

'aggiorna la visualizzazione del modulo
con i nuovi dati UpdateForm
FINE
```

Come puoi vedere, passare al primo record è stato piuttosto semplice. Ora codificheremo i pulsanti Prev e Next. Fai doppio clic sul secondo ToolButton (←) e sarai portato alla subroutine PrevRecordBtn_Click (). Aggiungi questo codice:

```
PUBLIC SUB PrevRecordBtn_Click ()
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gamasas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

```
'solo per il debug
PRINT "in Prec: il puntatore corrente è:" & RecordPointer

'ci stiamo spostando su un record inferiore, quindi diminuiamo il
puntatore del record DEC RecordPointer

'se siamo passati al record 1, reimpostare al
record 1 IF RecordPointer = 0 ALLORA
    RecordPointer = 1
ENDIF

'aggiorna la nostra variabile di contatto con i nuovi dati del record
(precedente) ContactRecord = Contatti [CStr (RecordPointer)]

'se siamo già all'inizio dei dati avvisare l'utente IF
RecordPointer = 1 ALLORA
    'solo per il debug
    PRINT "Già al primo contatto!"

ALTRO
    'solo per il debug
    PRINT "in Prev ora il puntatore corrente è:" & RecordPointer
ENDIF
'aggiorna il nostro modulo con i
nuovi dati UpdateForm
FINE
```

Ora codificheremo il pulsante Avanti. Fai doppio clic sul terzo ToolButton (→) e sarai portato alla subroutine NextRecordBtn_Click (). Aggiungi questo codice:

```
PUBLIC SUB NextRecordBtn_Click

() 'solo per il debug
PRINT "in Next: il puntatore corrente è:" & RecordPointer

'ci spostiamo in avanti nella raccolta, quindi incrementiamo il
puntatore INC RecordPointer

'se ci siamo spostati oltre l'ultimo record, ripristina
l'ultimo record IF RecordPointer> Contacts.Count THEN
    RecordPointer = Contacts.Count
ENDIF

'aggiorna il record del contatto con i nuovi dati
ContactRecord = Contatti [CStr (RecordPointer)]

'se siamo già all'ultimo record, dillo all'utente IF
RecordPointer = Contacts.Count ALLORA
```

This product is (C) 2000, John W. Rittinghouse, all rights reserved. It is released to the
Gambar User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'solo per il debug
STAMPA "Finalmente un contatto
già!" ALTRO
'solo per il debug
PRINT "in Next, il puntatore corrente è:" & RecordPointer
ENDIF

'aggiorna nuovamente il nostro modulo
con i nuovi dati UpdateForm
FINE 'prossima mossa da record
```

L'ultimo ToolButton di cui abbiamo bisogno per codificare per spostarci nei dati è per la subroutine LastRecordBtn_Click (). Questa subroutine ci posizionerà alla fine della raccolta dei dati, all'ultimo record. Fare doppio clic sul quarto ToolButton e aggiungere questo codice:

```
PUBLIC SUB LastRecordBtn_Click ()
'imposta il puntatore del record all'ultimo record utilizzando la
proprietà collection.count.
RecordPointer = Contacts.Count
'aggiorna il nostro record var con i nuovi dati per l'ultimo record
ContactRecord = Contatti [CStr (RecordPointer)]
'aggiorna il modulo con i nuovi
dati UpdateForm
FINE
```

Pulsanti degli strumenti di codifica: aggiunta di un record

Ora vediamo come aggiungere un nuovo contatto alla raccolta. Fai doppio clic sul quinto ToolButton, l'icona della piccola persona, e verrai inserito nella finestra del codice nella subroutine AddBtn_Click (). Esaminiamo il codice:

```
PUBLIC SUB AddBtn_Click ()

'dichiara un contatto locale variabile solo per l'uso in questa
subroutine DIM MyContactRecord AS Contact

'istanzia il record del contatto in modo da poterlo
utilizzare MyContactRecord = NUOVO contatto

'se non c'è il nome, non aggiungeremo un record IF
FirstName.Text <> "" THEN
'solo per il debug
STAMPA "Aggiunta" & FirstName.Text & "alla raccolta."

'siamo qui quindi abbiamo trovato un nome. Assegna i dati della
casella di testo ai campi memorizzati nel record:
MyContactRecord.FirstName = FirstName.Text
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
MyContactRecord.LastName = LastName.Text
MyContactRecord.Initial = MI.Text
MyContactRecord.Suffix = Suffix.Text
MyContactRecord.Street1 = Street1.Text
MyContactRecord.Street2 = Street2.Text
MyContactRecord.City = City.Text
StateContate .Zip5 = Zip5.Text
MyContactRecord.Zip4 = Zip4.Text
MyContactRecord.Areacode = AreaCode.Text
MyContactRecord.Prefix = DialPrefix.Text
MyContactRecord.Last4 = DialLast4.Text

'incrementa la chiave del record in modo che ogni chiave sia univoca quando viene aggiunta alla raccolta. Gambas richiede che questa venga convertita in una variabile stringa quando la aggiungiamo di seguito.
MyContactRecord.RecordID = Contacts.Count + 1

'se aggiungiamo un record non viene etichettato come cancellato MyContactRecord.Deleted = FALSE

'imposta il puntatore del record globale al valore della chiave incrementata RecordPointer = MyContactRecord.RecordID

'ora, chiama il metodo Collection.Add per aggiungere il record Contacts.Add (MyContactRecord, CStr (RecordPointer))

'questo serve solo per il debug per mostrare cosa è stato aggiunto IF Contacts.Exist (CStr (RecordPointer)) THEN
    'solo per il debug
    STAMPA "Record" & CStr (RecordPointer) &
"aggiunto!" ALTRO
    'solo per il debug
    PRINT "Il record non esiste!"
FINISCI SE

ALTRO 'Non è possibile aggiungere un contatto senza un nome, informa l'utente Message.Info ("Impossibile aggiungere senza un nome", "Ok")
FINISCI SE

'aggiorna il modulo con i nuovi dati UpdateForm

'questo è per scopi di debug, mostra tutti i record nella raccolta PER OGNI MyContactRecord IN contatti
    STAMPA "Record" & CStr (RecordPointer) & ":";
    STAMPA MyContactRecord.FirstName & " " & MyContactRecord.LastName
AVANTI
FINE 'aggiungi routine
```

Una guida per principianti a

Pulsanti degli strumenti di codifica: cancellazione dei dati

Il sesto ToolButton viene utilizzato per cancellare i dati nel modulo per consentire agli utenti di iniziare con un nuovo modulo vuoto per inserire i dati. Cancella semplicemente ogni campo della casella di testo. Fai doppio clic su ClearBtn e inserisci questo codice:

```
PUBLIC SUB ClearBtn_Click
() FirstName.Text = ""
MI.Text = ""
LastName.Text = ""
Suffix.Text = ""
Street1.Text = ""
Street2.Text = ""
City.Text = ""
State.Text = ""
Zip5.Text = ""
Zip4.Text = ""
AreaCode.Text = ""
DialPrefix.Text = ""
DialLast4.Text = ""
FINE
```

Convalida dell'input dell'utente

Quando l'utente inserisce i dati nel campo Iniziale e anche nei campi Suffix, State, Zip5, Zip4, AreaCode, DialPrefix e DialLast4, vogliamo assicurarci che i dati si adattino e che non possano inserire lettere o caratteri estranei. Ciascuna delle routine seguenti viene attivata in un evento di modifica. In altre parole, se il testo nel campo TextBox.Text cambia, andrà alla rispettiva routine _Change () di seguito. Ogni routine funziona in modo quasi identico per limitare il numero di caratteri che l'utente può digitare. Se ne digitano di più, i caratteri extra vengono troncati alla lunghezza specificata dal programma. Ad esempio, un prefisso non può contenere più di tre cifre. Potremmo anche aggiungere un codice per assicurarci che solo le cifre siano inserite nei campi numerici, ecc., Ma non è questo il motivo per cui stiamo creando questo programma. Lo lascerò a voi da aggiungere come esercizio alla fine di questo capitolo. Per ogni controllo di seguito, avremo bisogno di una stringa temporanea var, sTemp. Se la lunghezza dei dati TextBox.Text supera la lunghezza consentita, utilizziamo MID \$ per copiare le prime n lettere in sTemp e quindi copiare nuovamente quel valore nella proprietà TextBox.Text.

```
PUBLIC SUB MI_Change
() DIM sTemp AS
String
SE Len (MI.Text)> 1 ALLORA
```

Una guida per principianti a

```
sTemp = Mid $ (MI.Text, 1,  
1) MI.Text = sTemp
```

Una guida per principianti a

```
ENDIF
END

PUBLIC SUB Suffix_Change
() DIM sTemp AS String

IF Len (Suffix.Text)> 3 THEN sTemp
= Mid $ (Suffix.Text, 1, 3)
Suffix.Text = sTemp
ENDIF
END

PUBLIC SUB State_Change
() DIM sTemp AS
String

IF Len (State.Text)> 2 THEN
sTemp = Mid $ (MI.Text, 1,
2) State.Text = sTemp
ENDIF
END

PUBLIC SUB Zip5_Change
() DIM sTemp AS String

IF Len (Zip5.Text)> 5 THEN
sTemp = Mid $ (MI.Text, 1,
5) Zip5.Text = sTemp
ENDIF
END

PUBLIC SUB Zip4_Change
() DIM sTemp AS String

IF Len (Zip4.Text)> 4 THEN
sTemp = Mid $ (MI.Text, 1,
4) Zip4.Text = sTemp
ENDIF
END

PUBLIC SUB AreaCode_Change
() DIM sTemp AS String

IF Len (AreaCode.Text)> 3 THEN sTemp
= Mid $ (AreaCode.Text, 1, 3)
AreaCode.Text = sTemp
ENDIF
END

PUBLIC SUB DialPrefix_Change ()
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

This product is (C) 2005 by John D. C. Bollinger. All rights reserved. It is released to the Gamasas User Community under a Gamasas Content License. It may not be distributed under any other terms or conditions.

Una guida per principianti a

```
DIM sTemp AS String

IF Len (DialPrefix.Text)> 3 THEN sTemp
    = Mid $ (DialPrefix.Text, 1, 3)
    DialPrefix.Text = sTemp
ENDIF
END

PUBLIC SUB DialLast4_Change
() DIM sTemp AS String

IF Len (DialLast4.Text)> 4 THEN sTemp
    = Mid $ (DialLast4.Text, 1, 4)
    DialLast4.Text = sTemp
ENDIF
END
```

Come puoi vedere, molto codice ripetitivo ma necessario. Puoi tornare più tardi e "a prova di proiettile" l'input di questo programma aggiungendo ulteriori controlli di convalida per ogni possibile cosa a cui puoi pensare ma ti mostrerò più avanti in questo libro un modo migliore, costruendo un InputBox personalizzato che eseguirà il processo di convalida automaticamente.

Aggiunta di una funzione di ricerca

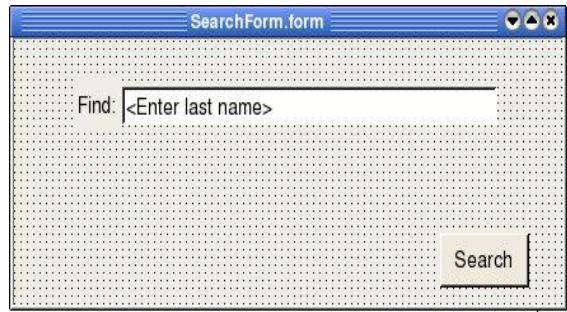
Vogliamo che l'utente sia in grado di trovare un contatto cercando un cognome. Ci sono così tanti controlli sul nostro modulo ora che aggiungerne un altro sminuirebbe l'estetica generale e lo farebbe apparire meno che elegante. Una soluzione migliore è utilizzare un clic del mouse in un punto qualsiasi del modulo stesso. Genera un evento e ci consentirà di catturare quell'evento e avviare un processo di ricerca. Quando ciò accade, faremo apparire un modulo di ricerca personalizzato e chiederemo all'utente di inserire un cognome che useremo come chiave per la nostra ricerca. Crea un evento per MouseDown nel modulo. Ecco il codice per l'evento MouseDown:

```
PUBLIC SUB Form_MouseDown
() SearchForm.ShowModal
Message.Info ("Torna al modulo principale con:" & SearchKey,
"Ok") DoFind
FINE
```

Nota che mostriamo il form usando il metodo ShowModal invece di chiamare solo Show. Questo perché vogliamo che l'utente inserisca qualcosa e faccia clic sul pulsante Cerca. Ecco come appare il modulo stesso in modalità progettazione:

This product is (C) 2001 W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a



Abbastanza semplice, solo un'etichetta, una casella di testo e un pulsante. Dall'IDE creare un nuovo modulo, denominato SearchForm e aggiungere l'etichetta e il pulsante come mostrato di seguito. Assegna un nome al pulsante SearchBtn e la casella di testo dovrebbe essere denominata SearchInput. Ecco tutto il codice necessario per il SearchForm:

File di classe Gambas

```
SearchForm PUBLIC SUB

SearchBtn_Click ()

'questa è una chiamata al nostro modulo personalizzato,
ricerca, che ha' un unico metodo, SearchOn, che è
spiegato di seguito Search.SearchOn (SearchInput.Text)

'solo per il debug
PRINT "restituendo:" & SearchInput.Text

'assegna la chiave di ricerca alla var globale che abbiamo nominato in
Form1 Form1.SearchKey = SearchInput.Text

'ora chiudi il nostro
modulo di ricerca
SearchForm.Close
FINE

PUBLIC SUB Form_Open ()
'imposta la didascalia per la finestra di ricerca
SearchForm.Caption = "Trova per cognome"

'evidenzia e seleziona la casella di testo SearchInput
'poiché è presente un solo campo di input, è già attivo
SearchInput.Select
FINE
```

A questo punto, ti starai chiedendo cosa sta succedendo esattamente.

Una guida per principianti a
Fondamentalmente, in Gambas, le variabili pubbliche sono pubbliche in un file di classe. Quando vogliamo passare parametri tra classi, possiamo usare il metodo di passare parametri con

Una guida per principianti a

funziona come quando abbiamo avviato il programma e passato una variabile di raccolta vuota al metodo GetData.

Un altro modo per ottenere ciò è utilizzare i moduli. Il punto chiave da ricordare è che le variabili pubbliche nei moduli sono pubbliche per tutte le classi. In sostanza, tutto ciò che abbiamo fatto è creare un condotto per passare la nostra chiave di ricerca dal modulo di ricerca al programma principale tramite il modulo. Notare che l'unico codice necessario nel modulo è un'istruzione che prende il parametro passato dal modulo di ricerca e lo restituisce al programma principale. Non appena ritorna dal modulo, viene assegnato alla variabile pubblica della classe Form1 SearchKey. Tieni inoltre presente che abbiamo qualificato completamente il nome della variabile, specificando Form1.SearchKey per assicurarci che si risolva correttamente. Ciò è necessario perché la subroutine SearchButton_Click () che ha richiamato il modulo si trova in un file di classe diverso dal file di classe Form1 che lo ha chiamato.

```
'File del modulo Gamas

FUNZIONE PUBBLICA SearchOn (sKey AS String) AS
    String RETURN sKey
FINE
```

La subroutine DoFind

A questo punto, abbiamo aperto il modulo, ottenuto l'input dell'utente di un cognome da cercare e abbiamo restituito quella variabile al nostro programma utilizzando il metodo conduit del modulo descritto sopra. Ora siamo pronti per cercare effettivamente la raccolta e trovare la voce. Ecco il codice per farlo:

```
PUBLIC SUB DoFind ()

    'dichiara un oggetto contatto per uso
    locale DIM MyContactRecord AS Contact

    'istanziarlo MyContactRecord
    = NUOVO contatto

    'imposta il puntatore del record sul primo record della raccolta
    RecordPointer = 1

    'solo per il debug
    PRINT "In DoFind con:" & SearchKey

    'usa FOR EACH per scorrere ogni oggetto nella raccolta PER OGNI
    MyContactRecord IN contatti
```

This product is (C) 2005 by John W. Rittig. All rights are reserved. It is released to the Gambas User Community under an OpenContent License (OCL) and may not be distributed under any other terms or conditions without express written consent of the author.

Una guida per principianti a

```
'assegna ogni record temporaneo al nostro oggetto di  
contatto globale ContactRecord = Contatti [CStr  
(RecordPointer)]  
  
'se il cognome corrisponde alla chiave di ricerca, aggiorna il  
modulo e esci IF ContactRecord.LastName = SearchKey THEN  
    'solo per il debug  
    PRINT "Found:" & MyContactRecord.LastName  
    'ora aggiorna il modulo  
    UpdateForm  
    ROMPERE 'forza il ritorno con questa  
dichiarazione FINISCI SE  
'nessuna corrispondenza, quindi incrementiamo il puntatore del  
record e passiamo al successivo INC RecordPointer  
FINE  
SUCSES  
SIVA
```

Di nuovo ToolButtons: Aggiornamento di un record

Se l'utente dovesse modificare i dati su un record esistente quando viene visualizzato, desideriamo che i dati vengano salvati nello stesso record. Per ottenere ciò, l'utilizzo di un oggetto di raccolta è complicato dal fatto che ogni oggetto della raccolta deve avere una chiave univoca. Per aggirare questo problema, elimineremo l'oggetto con la chiave corrente usando il metodo integrato Remove e leggeremo l'oggetto usando la stessa chiave ma con dati aggiornati. Un po' subdolo, ma porta a termine il lavoro senza troppi sforzi. Ecco come funziona:

```
PUBLIC SUB Update_Click ()  
    'dichiara un oggetto di contatto locale con cui  
    lavorare DIM MyContactRecord AS Contact  
  
    'istanziarlo MyContactRecord  
    = NUOVO contatto  
    'ora rimuovi il record esistente nella raccolta  
    Contacts.Remove (CStr (RecordPointer))  
  
    'quindi popoliamo il nostro record di lavoro con i  
    dati del modulo MyContactRecord.FirstName =  
    FirstName.Text MyContactRecord.LastName =  
    LastName.Text MyContactRecord.Initial = MI.Text  
    MyContactRecord.Suffix = Suffix.Text  
    MyContactRecord.Street1 = Street1.Text  
    MyContactRecont.Street2 = City .State = State.Text  
    MyContactRecord.Zip5 = Zip5.Text  
    MyContactRecord.Zip4 = Zip4.Text  
    MyContactRecord.Areacode = AreaCode.Text
```

This product is (C) 2005 by John W. Rittinger. All rights reserved. It is released to the Gamas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
MyContactRecord.Prefix = DialPrefix.Text
MyContactRecord.Last4 = DialLast4.Text

"aggiungi la copia di lavoro alla raccolta con la
stessa" chiave che apparteneva al record che abbiamo
eliminato in precedenza Contacts.Add
(MyContactRecord, CStr (RecordPointer))

IF Contacts.Exist (CStr (RecordPointer))
    THEN 'solo per il debug
        STAMPA "Record" & CStr (RecordPointer) & "aggiornato!"
    ALTRO
        STAMPA "Record non aggiornato!" 'vogliamo questo output se
fallito FINISCI SE
'aggiorna il modulo con i nuovi
dati UpdateForm
FIN
E
```

Di nuovo i pulsanti degli strumenti: eliminazione di un record

Se l'utente desidera eliminare un record in una raccolta, dobbiamo anche pensare a come gestirlo. Gestire quali chiavi sono disponibili e quali no potrebbe essere l'incubo di un programmatore, quindi un approccio più semplice è semplicemente contrassegnare il record come cancellato e cancellarlo. In questo modo, è disponibile per il riutilizzo con la stessa chiave. Cancella i dati e, quando si verifica il salvataggio, non memorizza i vecchi dati, eliminandoli di fatto. L'unica differenza è che possiamo riutilizzare la chiave facendo semplicemente inserire i dati in un record vuoto e fare clic sul pulsante di aggiornamento. Ecco come si ottiene:

```
PUBLIC SUB DeleteRecordBtn_Click ()

'dichiara il nostro oggetto di
contatto locale DIM
MyContactRecord AS Contact

'istanziarlo MyContactRecord
= NUOVO contatto

'rimuove il record corrente usando il metodo Remove incorporato
Contacts.Remove (CStr (RecordPointer))

'cancella tutti i campi del
modulo ClearBtn_Click

'imposta il primo campo con un indicatore che il record è stato
cancellato MyContactRecord.FirstName = "<Record eliminato>"
```

Una guida per principianti a

'lascia il resto vuoto semplicemente assegnandoli dopo la
cancellazione **MyContactRecord.LastName = LastName.Text**

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

```
MyContactRecord.Initial = MI.Text
MyContactRecord.Suffix = Suffix.Text
MyContactRecord.Street1 = Street1.Text
MyContactRecord.Street2 = Street2.Text
MyContactRecord.City = City.Text
MyContactRecord.State = State.Text5
MyZcontact5Record .Zip4 = Zip4.Text
MyContactRecord.Areacode = AreaCode.Text
MyContactRecord.Prefix = DialPrefix.Text
MyContactRecord.Last4 = DialLast4.Text

' contrassegna il record come eliminato
MycontactRecord.Deleted = TRUE

' aggiunge il record vuoto ed eliminato così come esiste alla raccolta
Contacts.Add (MyContactRecord, CStr (RecordPointer))

IF Contacts.Exist (CStr (RecordPointer))
    THEN ' solo per il debug
        STAMPA "Record" e CStr (RecordPointer) e "contrassegnato
come cancellato!" ALTRO
        ' se fallisce, vogliamo un messaggio Message.Info
        ('Record non contrassegnato come cancellato!",
        "Ok")
    FINISCI SE
    ' aggiorna il
FIN modulo
E   UpdateForm
```

Di nuovo ToolButtons: Salvataggio dei dati

L'ultima cosa di cui abbiamo bisogno per codificare è il pulsante di salvataggio. Questo è davvero semplice perché abbiamo già creato un metodo nel nostro file di classe Contact per farlo. Qui, tutto ciò che facciamo è chiamare il metodo PutData:

```
PUBLIC SUB SaveAllBtn_Click ()

' invoca il metodo PutData
ContactRecord.PutData
(Contatti) ' solo per il debug
STAMPA "Contatti salvati nel file
contact.data." FINE
```

Questo è tutto ciò che ci resta da fare. Esegui il programma e inserisci alcuni dati. Assicurati di provare tutte le opzioni e di creare alcuni record. Salva i tuoi dati su un file e ricaricalo. Quando sei soddisfatto che funziona come pianificato, puoi tornare indietro e commentare tutto l'output della console (le istruzioni PRINT) in

Una guida per principianti a
modo che non lo faranno

Una guida per principianti a

visualizzare in fase di esecuzione. Per tutti gli altri programmi che abbiamo creato, ci siamo accontentati di eseguirli semplicemente dall'IDE. Poiché questo programma può essere un'utile aggiunta al tuo desktop, ora ti mostreremo come creare un eseguibile autonomo e consentirgli di funzionare indipendentemente dall'ambiente di programmazione Gambas.

Creazione di un eseguibile autonomo

È molto facile creare l'eseguibile. Assicurati semplicemente che il tuo programma sia pronto per il rilascio (provalo prima!). Quindi vai al menu Progetto e scegli Rendi eseguibile. Gambas IDE creerà uno script di esecuzione che richiamerà Gambas Interpreter e ti consentirà di eseguire il tuo codice. Ho creato un collegamento desktop al programma e l'ho usato per il percorso dell'applicazione sul mio sistema:

```
"/ rittingj / I miei documenti / Gambas per principianti / Classi /  
Classi"
```

Ovviamente il tuo sistema sarà diverso. Dovrai puntare il collegamento sul desktop all'applicazione nel punto in cui l'IDE ha salvato il file eseguibile. Sulla distribuzione Linux che ho usato per questo programma, non avrebbe funzionato fino a quando non avessi racchiuso il percorso completo con segni di spunta, come mostrato sopra. Poiché Gambas Interpreter viene invocato come una chiamata di sistema, i segni di spunta sono necessari per impedire al sistema di pensare che stia passando parametri a / rittingj / My (dove ricorre il primo carattere spazio). Ecco il risultato finale, il nostro programma in esecuzione in modalità standalone sul desktop:



This product is (C) 2004 Jonathan W. Rittinger. It is released to the public under an Open Content License (OCL) and may not be distributed or modified without the express written consent of the author.

Figura 76 Gestione contatti in esecuzione autonoma sul desktop.

Capitolo 12 - Imparare a disegnare

La classe Draw è la classe utilizzata in Gambas per disegnare su un oggetto. L'oggetto può essere di tipo Picture, Window, Printer, Drawing o DrawingArea. Questa classe è statica. È importante ricordare che prima di iniziare a disegnare qualsiasi cosa, è necessario chiamare il metodo Begin passandogli un handle al tipo di oggetto che si desidera disegnare. Una volta fatto ciò, puoi chiamare uno qualsiasi dei metodi di disegno forniti da GB per disegnare punti, linee, testi, immagini, ecc. Tuttavia, non è chiaramente documentato che devi anche impostare la proprietà Cache di DrawingArea su True se vuoi vedere il tuo disegno appare nel modulo. Al termine del disegno, è necessario chiamare il metodo End.

Proprietà di disegno

Ogni elemento che desideri disegnare in Gambas è supportato con diversi attributi comuni, come il colore di sfondo e il colore di primo piano, un colore di riempimento, lo stile di riempimento, ecc. Alcune proprietà in Gambas, come Clip, sono di sola lettura e ti restituiscono semplicemente i dati per utilizzare nel codice.

BackColor / Background e ForeColor / Foreground

Come abbiamo spiegato nel Capitolo 4, molte delle proprietà dei controlli sono comuni a tutti i controlli, quindi in questo capitolo spiegheremo solo le proprietà che sono univoche per un dato controllo.

Colore di sfondo è definito come PROPRIETÀ BackColor AS Integer
ForeColor è definito come PROPRIETÀ ForeColor AS Integer

Questo valore intero rappresenta il colore utilizzato per lo sfondo del disegno o il colore di primo piano corrente. È sinonimo delle proprietà Background / Foreground. Tieni presente che PROPRIETÀ è un tipo di dati predefinito utilizzato internamente in Gambas. È possibile utilizzare le costanti predefinite di Gambas per il colore per impostare il valore del colore:

```
NeroBluCianoBlu scuro  
DarkCyanDarkGrayDarkGreenDarkMagenta  
DarkRedDarkYellow  
DefaultGray  
Verde Grigio Chiaro Magenta Arancione  
Rosa Rosso Trasparente Viola Bianco  
Giallo
```

This document is licensed by John W. Hartman, all rights are reserved. It is based to the
Gambas Documentation under an Creative Commons Attribution-NonCommercial-
ShareAlike license. It may be distributed
under different conditions without the express written consent of the author.

Una guida per principianti a

Per impostare la proprietà BackColor su rosso, dovresti usare questo codice:

```
Draw.BackColor = Color.Red
```

In alternativa, come abbiamo affermato in precedenza, se conosci i valori RGB o HSV per un colore specifico, GB fornisce un mezzo per convertire quei valori in un valore intero che può essere passato alla proprietà BackColor (o altro colorrelated). La classe Color fornisce due metodi, RGB e HSV che puoi utilizzare. La funzione RGB restituisce un valore di colore dai suoi componenti rosso, verde e blu. HSV restituisce un valore di colore dai suoi componenti di tonalità, saturazione e valore. Per utilizzare una di queste funzioni per impostare il colore di sfondo del pulsante, ecco cosa potresti codificare:

```
Draw.BackColor = Color.RGB (255,255,255)
```

Questo imposterebbe il colore di sfondo dell'oggetto Draw su bianco. Ciò è particolarmente utile quando si tenta di impostare colori i cui valori non rientrano nei valori delle costanti predefinite forniti in GB.

Clip

La classe Clip è statica. Clip è una proprietà di sola lettura che restituisce un oggetto virtuale (.DrawClip) utilizzato per gestire l'area di ritaglio di un disegno. La classe virtuale .DrawClip viene utilizzata per definire l'area di ritaglio di un disegno. I metodi di disegno Gambas non disegnano mai al di fuori dei confini dell'area di ritaglio definita. Non è possibile utilizzare questa classe virtuale come tipo di dati. La sintassi standard del linguaggio Gambas è:

PROPRIETÀ STATICA LEGGERE Clip COME .DrawClip

Questa classe può essere utilizzata come funzione. Per richiamare la funzione per definire un'area di ritaglio, utilizzare questa sintassi del linguaggio Gambas:

```
STATIC SUB .DrawClip (X AS Integer, Y AS Integer, W AS Integer, H AS Integer)
```

Le proprietà restituite da questa funzione includono Enabled, H, Height, W, Width, X e Y.

FillColor, FillStyle, FillX, FillY

La proprietà FillColor restituisce o imposta il colore utilizzato dai metodi di disegno

This product is covered by U.S. Patent and Trademark Office Patent No. 6,535,712. © 2002 by John W. Ringhouse, all rights reserved. This release of the source code does not constitute an express or implied license to sell, copy, or distribute it without the express or implied consent of the author.

Una guida per principianti a

in grado di riempire un poligono con un colore o motivo o entrambi. Questa classe è statica. La proprietà FillStyle restituisce o imposta lo stile utilizzato dai metodi di disegno in grado di riempire un poligono con un colore o motivo o entrambi. La classe Fill supporta diverse costanti FillStyle predefinite utilizzate per rappresentare i modelli di riempimento per il disegno:

| | | | |
|--------------------------|---------------------|----------------------|--------------------|
| IndietroDiagonale | Attraversare | CrossDiagonal | Denso12 |
| Denso37 | Denso50 | Denso6 | Denso63 |
| Denso88 | Denso94 | Diagonale | Orizzontale |
| Nessuna | Solido | Verticale | e |

La sintassi del linguaggio Gambas per FillColor e FillStyle è la seguente:

```
PROPRIETÀ STATICA FillColor AS
Intero PROPRIETÀ STATICA FillStyle
AS Intero
```

La proprietà FillX e la proprietà FillY vengono utilizzate per restituire o impostare l'origine orizzontale / verticale dei pennelli utilizzati dai metodi di disegno in grado di riempire un poligono con un colore o motivo o entrambi. La sintassi del linguaggio Gambas per FillX e FillY è:

```
PROPRIETÀ STATICA FillX AS
Intero PROPRIETÀ STATICA FillY
AS Intero
```

Font

La classe Font restituisce o imposta il carattere utilizzato per il rendering del testo sulla superficie di disegno. La sintassi del linguaggio Gambas è:

```
PROPRIETÀ STATICA Font AS Font
```

Per impostare gli attributi dei caratteri di controllo, è possibile utilizzare questo codice:

```
Draw.Font.Name = "Lucida"
Draw.Font.Bold = VERO
Draw.Font.Italic = FALSO
Draw.Font.Size = "10"
Draw.Font.StrikeOut = FALSO
Draw.Font.Underline = FALSO
```

Una guida per principianti a Invertire

La proprietà Inverti viene utilizzata per stabilire che tutte le primitive di disegno combineranno i propri colori pixel con i colori pixel della destinazione utilizzando un'operazione XOR. La sintassi del linguaggio Gambas è:

Una guida per principianti a

PROPRIETÀ STATICHE Inverti come booleano

LineStyle / LineWidth

La proprietà LineStyle restituisce o imposta lo stile utilizzato per disegnare le linee. LineWidth restituisce o imposta la larghezza utilizzata per disegnare le linee. La classe Line è statica e definisce le costanti utilizzate dalla proprietà Draw.LineStyle. Queste costanti includono:

| | | |
|-------------------|----------------|----------------|
| Dash | Punto | DashDot |
| DashDotDot | Nessuna | Solido |

La sintassi standard del linguaggio Gambas è:

```
STATIC PROPERTY LineStyle AS Integer
STATIC PROPERTY LineWidth AS Integer
```

Trasparente

La proprietà Transparent indica che alcuni metodi di disegno come Draw.Text non riempiono lo sfondo con nulla (quindi sono trasparenti). La sintassi del linguaggio Gambas è:

```
PROPRIETÀ STATICHE Trasparente AS Booleano
```

Metodi di disegno

Uno dei modi migliori per imparare a utilizzare i metodi di disegno di Gambas è scrivere codice che dimostri l'uso di ciascun metodo. In questo modo potrai vedere come programmare il metodo e ottenere una gratificazione immediata vedendone i risultati. Costruiremo un'applicazione demo che ti mostrerà come utilizzare quasi tutti i metodi supportati dalla classe Draw.

Inizieremo a creare una nuova applicazione di interfaccia utente grafica. Assegna un nome al progetto gfxDemo e quando viene visualizzato l'IDE, crea un modulo, Form1. Assicurati di specificarlo come classe di avvio e assicurati che i controlli siano pubblici. Ecco come apparirà il layout di Form1 in fase di esecuzione:

Una guida per principianti a



Figura 77 layout gfxDemo Form1.

Imposta la larghezza del modulo su 530 e l'altezza su 490. Le impostazioni predefinite per tutte le altre proprietà dovrebbero andare bene per i nostri scopi. Crea otto pulsanti e etichettali come mostrato nell'immagine sopra. I nomi dei pulsanti sono TextBtn, InvRectBtn, EllipseBtn, FillRectBtn, PolygonBtn, PolyLineBtn, TileBtn e QuitBtn. Ogni pulsante dovrebbe avere una larghezza di 60 e un'altezza di 25. Posizionarli il più in basso possibile sul modulo senza tagliare la parte inferiore del pulsante. Per ogni pulsante che hai creato, dovremo creare un evento clic facendo doppio clic sul pulsante.

Man mano che passiamo attraverso le spiegazioni di ciascuno dei seguenti metodi, aggiungeremo codice ove appropriato. Prima di arrivare ai metodi stessi, ci sono un paio di cose che dobbiamo fare con il nostro programma. Innanzitutto, dobbiamo aggiungere un controllo DrawingArea, denominato da al form. Posizionalo nell'angolo in alto a sinistra di Form1 e fallo circa 1 pollice per 1 pollice. Lo ridimensioneremo dinamicamente nel nostro programma. Successivamente, utilizzeremo un costruttore per avviare il processo di disegno, invocando il metodo Draw.Begin e impostando i parametri che verranno utilizzati per il nostro programma. Ecco la subroutine del costruttore che devi aggiungere:

```
PUBLIC SUB _new ()  
    'stabilisce la larghezza dell'area  
    di disegno da.W = form1.W - 10  
  
    'stabilisce l'altezza dell'area di  
    disegno da.H = form1.H 45  
  
    'effettuare la chiamata obbligatoria per avviare le  
    operazioni di sorteggio Draw.Begin (da)  
    'imposta una larghezza di linea predefinita di 2 pixel
```

This product is (C) 2005 by John V. Gambas User Community under all rights reserved released to the postazioni strisci scopi nomi de MyLineBtn, altezza d' inferiore vento click i seguenti i stessi, c'anzitutto al form 1 pollice avamente il metodo il nostro the author.

Una guida per principianti a

```
Draw.LineWidth = 2  
END
```

Quando il modulo si apre all'avvio del programma, chiamerà prima il costruttore sopra, quindi eseguirà il codice nella subroutine Form_Open. Tutto quello che faremo per questa subroutine è impostare la didascalia nella parte superiore del modulo:

```
PUBLIC SUB Form1_Open ()  
    Form1.Text = "Esempi di disegno"  
END
```

L'ultima routine che creeremo prima di esplorare i metodi Draw è il codice necessario per fermare il nostro programma. Fai doppio clic sul pulsante Esci e aggiungi questo codice:

```
PUBLIC SUB QuitBtn_Click ()  
    Disegna.Fine 'chiudere le  
    operazioni di disegno ME.Chiudi  
    'chiudi Form1  
FINE
```

Ora siamo pronti per iniziare ad aggiungere codice per esplorare i nostri vari strumenti di disegno, i metodi della classe Draw.

Text / TextHeight / TextWidth

Inizieremo con Draw.Text e le due proprietà di sola lettura, TextHeight e TextWidth. La proprietà TextHeight restituisce l'altezza di un disegno di testo mentre la proprietà TextWidth restituisce la larghezza di un disegno di testo. La sintassi standard del linguaggio Gambas è:

```
FUNZIONE STATIC TextHeight (Text AS String) AS Integer  
FUNZIONE STATIC TextWidth (Text AS String) AS Integer
```

Fai doppio clic sul pulsante Testo e aggiungi questo codice:

```
PUBLIC SUB TextButton_Click ()  
  
    'questa var verrà utilizzata per il nostro  
    contatore di loop Contatore DIM AS Integer  
  
    'cancella l'area di disegno  
    corrente da.Clear
```

Una guida per principianti a
'imposta il colore di primo
piano sul nero **Draw.ForeColor**
= color.Black

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambar User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'imposta il carattere
corrente su Arial
draw.Font.Name = "Arial"

'attiva il grassetto
draw.Font.Bold = TRUE

'disattiva corsivo
draw.Font.Italic = FALSE

'disattiva la
sottolineatura
draw.Font.Underline = FALSE

'imposta la dimensione del
testo a 16 punti
draw.Font.Size = "16"

'avvia un ciclo e continua fino a quando non iteriamo
10 volte Contatore FOR = da 0 a 9

'testo di output e incremento della posizione y di 40 *
contatore' i parametri di larghezza e altezza opzionali
sono forniti come 100, 60 draw.Text ("Sample Text", 10, 10
+ 40 * counter, 100,60)
IL PROSSIMO 'iterazione del ciclo

'ora imposta il carattere
su 24 punti draw.Font.Size
= "24"
Draw.ForeColor = color.Blue 'e cambia il colore in blu

Contatore FOR = da 0 a 9 'loop altre 10 volte
'tutto ciò che cambiamo è la posizione x per spostare il testo ad
destra di 200 pixel draw.Text ("Altro testo di esempio", 200,
contatore 10 + 40 *, 100,60)
IL PROSSIMO 'iterazione

'aggiorna l'area di disegno per vedere cosa è stato
fatto da.Refresh
FINE 'abbiamo finito con le cose di testo!
```

Dopo aver aggiunto il codice sopra, salva il tuo lavoro e fai clic sul pulsante ESEGUI.
Ecco cosa dovresti vedere:

Una guida per principianti a

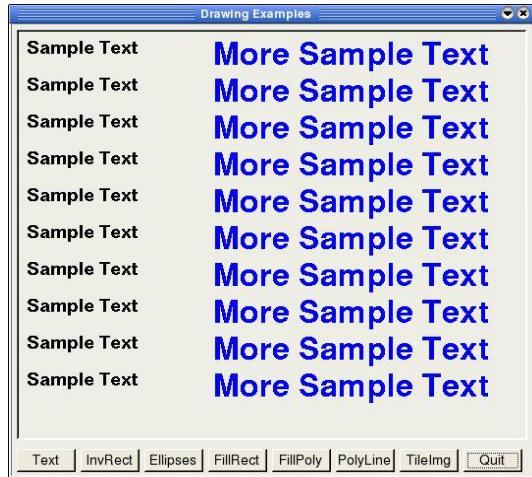


Figura 78 Risultati del clic sul pulsante Testo.

Disegna primitive: Punto / Retto / Ellisse / Linea

Il metodo Draw.Point disegna un singolo pixel mentre il metodo Draw.Rect disegna un rettangolo. La sintassi del linguaggio Gambas per questi due metodi è:

```
STATIC SUB Punto (X AS Integer, Y AS Integer)
STATICO SUB Retto (X AS Inter, Y AS Inter, Larghezza AS Inter, Altezza AS Inter)
```

Utilizzeremo il pulsante InvRect per illustrare come vengono utilizzati questi metodi. Fare doppio clic sul pulsante InvRect e immettere il codice seguente:

```
PUBLIC SUB InvRectBtn_Click ()
    'usa questa var per il nostro
    contatore di loop Contatore DIM AS
    Integer

    'cancella l'area di
    disegno da.Clear

    'assicurati che lo stile di
    riempimento sia impostato su Nessuno
    draw.FillStyle = fill.None

    'imposta il colore su ciano
    draw.ForeColor = color.cyan

    'questo farà apparire rosso il rettangolo
    ciano draw.Invert = TRUE
    Contatore FOR = da 0 a 15 'loop 16 volte

    'disegna un rettangolo, inc le posizioni startx, ye end x, y 5 *
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an OpenContent License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
counter val  
Draw.Rect (contatore 5 + 5 *, contatore 5 + 5 *, contatore da.W / 2 + 5 *,  
contatore da.H / 2 + 5 *)
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

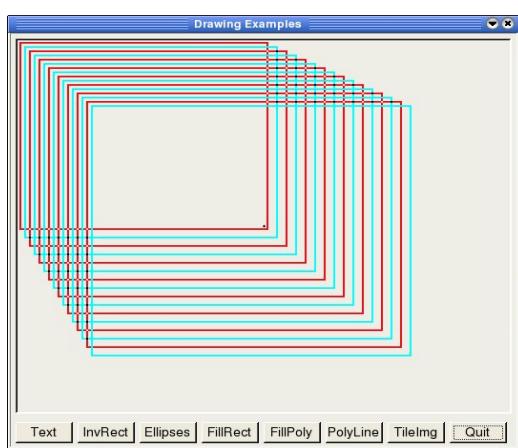
```
'se raggiungiamo un numero pari, invoca l'attivazione /
disattivazione della proprietà Inverti SE contatore MOD 2 =
0 ALLORA
    draw.Invert = FALSE
ELSE
    draw.Invert = TRUE
ENDIF
IL PROSSIMO 'iterazione del ciclo

'assicurarsi che invert sia
impostato su off draw.Invert
= FALSE
'imposta il colore fgd corrente
su nero draw.Forecolor =
color.Black 'imposta un punto a
metà schermo draw.Point (da.W /
2, da.H / 2)

'ora faremo un minuscolo mirino dal primo punto
'spostando un pixel in alto, in basso, a sinistra ea destra
del punto centrale' abbiamo impostato con la prima chiamata
draw.point draw.Point (da.W / 21, da.H / 2)
draw.Point (da.W / 2 + 1,
da.H / 2) draw.Point (da.W /
2, da.H / 21) draw.Point
(da.W / 2, da.H / 2 + 1)

'aggiorna il display per vedere cosa
abbiamo fatto da.Refresh
FINE 'del codice demo_rect / point
```

Salva il tuo lavoro ed esegui il programma. Quando fai clic sul pulsante InvRect, dovresti vedere qualcosa di simile a questo:



Una guida per principianti a
Figura 79 Risultati del clic del pulsante
InvRect. Notare il minuscolo schermo centrale
con mirino nero.

Una guida per principianti a

Il metodo Draw.Line traccia una linea. I punti x1, y1 rappresentano il vertice iniziale della linea, x2, y2 rappresentano il vertice finale della linea. Il metodo line utilizza le proprietà della classe Line per stabilire lo stile e lo spessore della linea. Le costanti utilizzate dalla proprietà DrawLineStyle sono Dash, DashDot, DashDotDot, Dot, None e Solid. Lo spessore della linea viene impostato utilizzando la proprietà DrawLineWidth e la larghezza viene impostata in pixel. La sintassi del linguaggio Gambas per questo metodo è:

```
STATIC SUB Riga (X1 AS Integer, Y1 AS Integer, X2 AS Integer, Y2 AS Integer)
```

Il metodo Draw.Ellipse disegnerà un'ellisse o un cerchio. Ricorda, un cerchio è un'ellisse con parametri di larghezza e altezza uguali, motivo per cui non vedi la funzione Draw.Circle in Gambas. La sintassi del linguaggio Gambas per Draw.Ellipse è:

```
STATIC SUB Ellisse (X AS Inter, Y AS Inter, Larghezza AS Inter, Altezza AS Inter [, Inizio AS Float, Lunghezza AS Float])
```

X e Y rappresentano il punto centrale del cerchio o dell'ellisse. La larghezza può essere considerata come un raggio lungo il piano orizzontale, l'altezza è un raggio lungo il piano verticale. Se si specificano i parametri opzionali di inizio e lunghezza, questi rappresentano gli angoli iniziali (in gradi) in cui il disegno inizierà e terminerà. Lo illustreremo nel nostro esempio di seguito utilizzando il pulsante Ellipses. Fai doppio clic sul pulsante Ellissi e inserisci questo codice:

```
PUBLIC SUB EllipseBtn_Click ()  
    ' dichiara alcune variabili con cui  
    lavorare  
    DIM x1 AS Inter  
    DIM x2 AS Inter  
    DIM y1 AS Inter  
    DIM y2 AS Inter  
  
    ' avremo bisogno di una  
    controvariabile DIM i AS Integer  
  
    ' e una variabile per mantenere il valore della larghezza  
    della linea mentre lo cambiamo DIM lwidth AS Integer  
  
    ' cancella il nostro display prima  
    di iniziare da.Clear  
  
    ' imposta i vettori  
    iniziali x1 = 50  
    y1 = 50  
    x2 = 100
```

Una guida per principianti a
y2 = 100

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gamas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'cambia il colore in blu scuro
Draw.ForeColor = color.DarkBlue

'impostato per un riempimento solido. Qualsiasi contenitore
compilabile chiamato dopo questa chiamata verrà riempito con il
modello specificato di seguito
draw.FillStyle = fill.Solid

'imposta un ciclo che creerà 12 incrementi
PER i = da 0 a 360 PASSO 30
  'se cadiamo su un incremento di 90 gradi, cambia i
  colori SE io MOD 45 = 0 ALLORA
    draw.FillColor = color.Yellow
  ELSE
    draw.FillColor = color.Red
  ENDIF

'disegna un'ellisse piena di 12 segmenti, partendo da ogni valore
di i' e spostandosi di 30 gradi, come specificato nei parametri
opzionali Draw.Ellipse (x1, y1, x2, y2, i, 30)
IL PROSSIMO

'ora, disattiva lo stile di
riempimento draw.FillStyle =
fill.None

'imposta il nostro colore fgd su nero
draw.ForeColor = color.Black

'tracciamo delle linee, iniziando x con l'asse orizzontale medio
25 pixel x1 = da. W / 225

'inizia la nostra y a metà schermo sull'asse
verticale y1 = da.H / 2

'inizia con una larghezza di linea di un
singolo pixel lwidth = 1

'loop 8 volte, spostandosi verso il basso lungo l'asse y con
incrementi di 25 pixel PER i = da 10 a 200 PASSO 25

'disegna una linea
draw.Line (da.W / 2, i, da.W / 2 + 150, i)

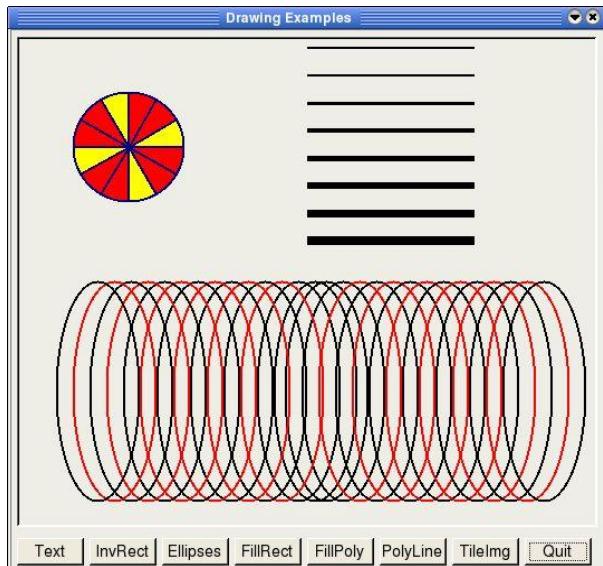
'aumentare lo spessore della
linea variabile Larghezza INC

'imposta la nuova larghezza
draw.LineWidth = lwidth
NEXT 'iterazione del ciclo
```

Una guida per principianti a

```
'fuori dal ciclo, ripristina un valore predefinito di 2  
pixel di larghezza draw.LineWidth = 2  
  
'ora, iniziamo un altro ciclo e disegniamo alcune  
ellissi PER i = da 1 a 40 FASE 3 'facciamo 13 di loro  
  
'spostando questo a sinistra lungo l'asse  
orizzontale Draw.Ellipse (x1 * 5, y1, 75,  
200)  
  
'e spostando questo lungo l'asse orizzontale  
Draw.Ellipse (x1 + i * 5, y1, 75, 200)  
  
'se raggiungiamo un numero pari nel nostro ciclo  
cambia i colori SE i MOD 2 = 0 ALLORA  
    draw.ForeColor = color.Red  
ELSE  
    draw.ForeColor = color.Black  
ENDIF  
IL PROSSIMO 'iterazione del ciclo  
  
'aggiorna l'area di disegno per vedere il  
nostro lavoro da.Refresh  
FINE 'fine del codice dimostrativo con puntini di sospensione e righe
```

Salva il tuo lavoro ed esegui il programma. Quando fai clic sul pulsante **Ellissi**, dovresti vedere qualcosa di simile a questo:



La Figura 80 **Ellissi** mostra il disegno di linee ed ellissi.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Ambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Il nostro prossimo segmento di codice utilizzerà il pulsante FillRect. Stiamo rivisitando il metodo Draw.Rect per mostrarvi come utilizzare Draw.FillStyle per creare oggetti riempiti. Crea un clickevent e inserisci il seguente codice:

```
PUBLIC SUB FillRectBtn_Click ()  
    'creare alcune variabili locali con  
    'cui lavorare DIM x1 AS Integer  
    DIM x2 AS Interio  
    DIM y1 AS Interio  
    DIM y2 AS Interio  
  
    'cancella l'area di  
    'disegno da.Clear  
  
    'imposta l'angolo retto in  
    'alto a sinistra x1 = 20  
    y1 = 80  
  
    'imposta l'angolo destro del  
    'bot rect x2 = x1 + 50  
    y2 = y1 + 50  
  
    'imposta il colore di fgd su  
    'rosso draw.FillColor = color.Red  
  
    'specifica un motivo di riempimento  
    'in stile orizz draw.FillStyle =  
    fill.Horizontal  
  
    'disegna un rettangolo, se viene specificato un motivo di riempimento,  
    'viene riempito automaticamente Draw.Rect (x1, y1, x2, y2)  
  
    'imposta l'angolo retto in  
    'alto a sinistra x1 = 220  
    y1 = 180  
  
    'imposta l'angolo destro del  
    'bot rect x2 = x1 + 50  
    y2 = y1 + 50  
  
    'imposta il colore di fgd su blu  
    draw.FillColor = color.Blue  
  
    'specifica un motivo di riempimento  
    'a croce draw.FillStyle = fill.Cross  
  
    'disegna un rettangolo, se viene specificato un motivo di riempimento,  
    'viene riempito automaticamente Draw.Rect (x1, y1, x2, y2)
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'disattiva il riempimento  
draw.FillStyle = fill.None  
  
'aggiorna per vedere il  
nostro lavoro da.Refresh  
FINE
```

Salva il tuo lavoro ed esegui il programma. Quando fai clic sul pulsante FillRect, dovresti vedere qualcosa di simile a questo:

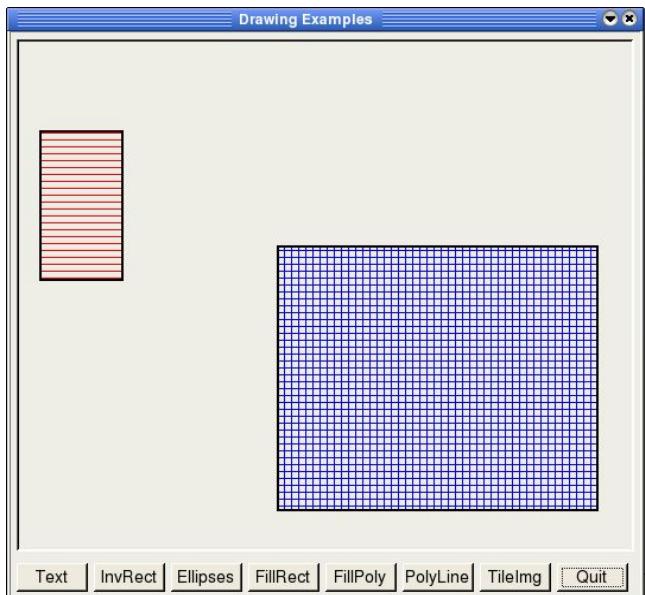


Figura 81 Output dopo aver fatto clic sul pulsante FillRect.

Disegna primitive: poligono e polilinea

Il metodo Draw.Polygon disegna un poligono con n vertici. I vertici sono specificati in un array Integer usando questa sintassi del linguaggio Gambas:

```
STATIC SUB Polygon (Points AS Integer [])
```

Il parametro Points è un array di Integer contenente le coordinate dei vertici del poligono. Si presume che le coordinate siano in formato x, y quindi devono esserci due numeri interi nell'array per ogni vertice del poligono.

Il metodo Draw.Polyline funziona quasi come la routine Polygon tranne per il fatto che con una polilinea la figura che si sta disegnando non ha bisogno di essere chiusa. Questo a volte è indicato come un poligono vuoto. Una polilinea è una serie di vertici

Una guida per principianti a

uniti dall'uno all'altro da una linea. La linea verrà tracciata dall'ultimo vertice al vertice successivo specificato nell'array. La sintassi del linguaggio Gambas per Polyline è:

```
STATIC SUB Polyline (Points AS Integer [])
```

Il parametro Points è un array di Integer contenente le coordinate dei vertici del poligono. Quindi devono esserci due numeri interi nell'array per ogni vertice. Useremo il pulsante FillPoly sul nostro Form1 per dimostrare l'uso del metodo Draw.Polygon. Fare doppio clic sul pulsante FillPoly per creare un evento clic e quando viene visualizzata la finestra del codice, immettere questo codice:

```
PUBLIC SUB PolygonBtn_Click ()  
    'dichiara alcune variabili di lavoro per creare  
    'un poligono DIM x1 AS Integer  
    DIM y1 AS Interger  
    DIM x2 AS Interger  
    DIM y2 AS Interger  
    DIM x3 AS Interger  
    DIM y3 AS Interger  
  
    'creeremo un triangolo per il nostro poligono' e  
    'memorizzeremo i vertici nel triangolo dell'array  
    'intero Triangolo DIM AS Integer []  
  
    'cancella l'area di  
    'visualizzazione  
    da.Clear  
  
    'imposta i vertici per ogni gamba del triangolo'  
    iniziando con la parte superiore di un triangolo  
    equilatero x1 = da.w / 2 'imposta l'asse orizzontale a  
    metà schermo  
    y1 = 10 'imposta l'asse verticale in alto + 10 pixel  
  
    'ora, faremo la gamba sinistra del triangolo'  
    iniziando dall'estremo bordo sinistro più 10 pixel  
    x2 = da.sinistra + 10  
  
    'e impostando la nostra posizione y su drawingarea altezza 200  
    pixel y2 = da.H 200  
  
    'la gamba destra imposta la posizione orizzontale  
    all'estrema destra di 20 pixel in x3 = da.W 20  
  
    'e l'asse verticale sarà lo stesso della seconda  
    tappa y3 = y2
```

Una guida per principianti a
'tutti i vertici definiti, carica l'array intero con il triangolo

This product is (C) 2005 by John W Kittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
triangolo = Array (x1, y1, x2, y2, x3,
y3) 'imposta uno stile di riempimento
draw.FillStyle = fill.Dense63

'e un colore di riempimento
draw.FillColor = color.DarkMagenta

'e traccia il poligono utilizzando i dati della matrice
triangolare draw.Polygon (triangolo)

'ricorda di impostare lo stile di
riempimento su Nessuno
draw.FillStyle = fill.None

'chiama refresh per vedere il
nostro lavoro da.Refresh
FINE 'e abbiamo finito con la routine del poligono
```

Salva il tuo lavoro ed esegui il programma. Quando fai clic sul pulsante FillPoly, dovrà vedere qualcosa di simile a questo:

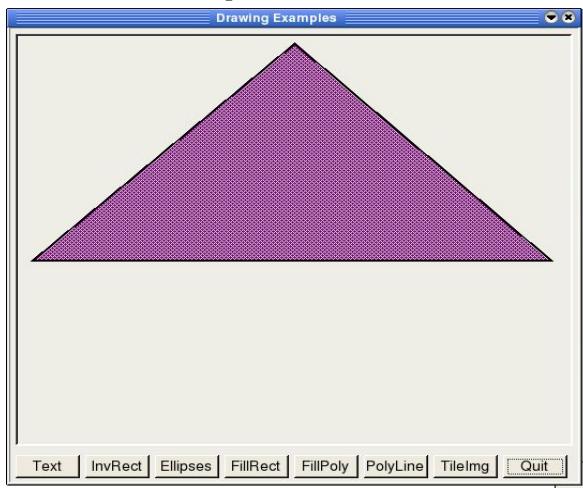


Figura 82 Utilizzo di Draw.Polygon per disegnare un triangolo.

Vediamo ora come funziona il metodo Draw.Polyline. Fare doppio clic sul pulsante Polilinea e creare un evento clic. Quando viene visualizzata la finestra del codice, inserisci questo codice:

```
PUBLIC SUB PolyLineBtn_Click ()
'dichiara alcune variabili per creare i nostri
vertici DIM x1 AS Integer
DIM y1 AS Integer
DIM x2 AS Integer
```

Una guida per principianti a

```
DIM y2 AS Interio
DIM x3 AS Interio
DIM y3 AS Interio
DIM x4 AS Interio
DIM y4 AS Interio

'dichiara un array intero per contenere i dati della
polilinea Righe DIM AS Integer []

'cancella l'area di
disegno da.Clear

'avvia il nostro primo punto a metà schermo a circa 10 pixel
dall'alto x1 = da.w / 2
y1 = 10

'il punto successivo sarà di 10 pixel dall'estrema sinistra, verso
il basso di 200 pixel x2 = da.sinistra + 10
y2 = da.H - 200

'il nostro 3 ° vertice sarà il punto morto dell'area di disegno per
sull'asse orizzontale' e molto al di sotto dell'area di disegno per
l'asse verticale
x3 = da.W 20
y3 = da.H 20

'e il 4 ° vertice sarà il punto morto xey dell'area di disegno x4 =
da.W / 2
y4 = da.H / 2

'carica i vertici nell'array
'se volessimo chiudere il poligono, tutto ciò che dobbiamo fare è
aggiungere i vertici' x1, y1 alla fine della matrice
'rimuovi il commento dalla riga successiva per provarlo e vedere
'righe = Array (x1, y1, x2, y2, x3, y3, x4, y4, x1, y1)
linee = Array (x1, y1, x2, y2, x3, y3, x4, y4)

'assicurati che tutti gli stili di
riempimento siano disattivati
draw.FillStyle = fill.None

'imposta il colore sul verde scuro
draw.ForeColor = color.DarkGreen

'traccia le linee
draw.Polyline
(linee)

'aggiorna per vedere il
nostro lavoro da.Refresh
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gamas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
FINE 'della routine demo della polilinea

Una guida per principianti a

Salva il tuo lavoro ed esegui il programma. Quando fai clic sul pulsante Polilinea, dovresti vedere qualcosa di simile a questo:

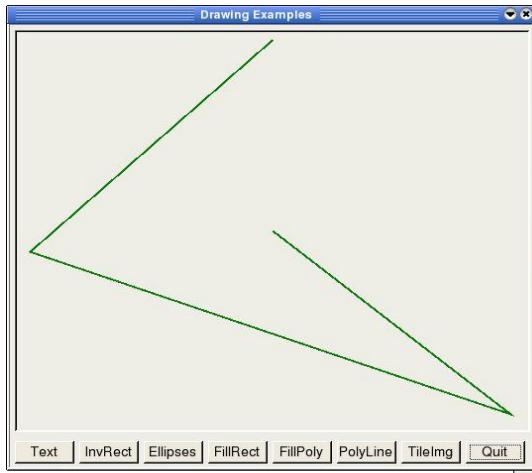


Figura 83 Utilizzo di Draw.Polyline per disegnare linee.

Immagine / Immagine / Piastrella

La classe Image disegna un'immagine o parte di essa. Il contenuto dell'immagine viene archiviato nella memoria di processo, non nel server di visualizzazione come un'immagine. Questa classe è creabile. Se Larghezza e Altezza non sono specificate, la nuova immagine è nulla. La sintassi del linguaggio Gambas è:

```
STATIC SUB Image (Image AS Image, X AS Integer, Y AS Integer [, SrcX AS Integer, SrcY AS Integer, SrcWidth AS Integer, SrcHeight AS Numero intero ] )
```

Il codice seguente creerebbe una nuova immagine:

```
DIM hImage AS Image  
hImage = NEW Image ([Width AS Integer, Height AS Integer])
```

Questa classe si comporta come un array di pixel. Pensa a X come righe e Y come colonne di pixel dell'immagine. Ecco come funzionerebbe il codice per ottenere o impostare un pixel di un'immagine:

```
DIM hImage AS Image  
DIM anInteger AS Integer  
  
anInteger = hImage [X AS Integer, Y AS Integer]
```

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Il codice sopra restituirà il colore di un pixel dell'immagine su X, Y mentre il seguente

Una guida per principianti a

code imposterà il colore di un pixel dell'immagine sul valore di un intero:

```
hImage [X AS Integer, Y AS Integer] = anInteger
```

L'immagine consente di utilizzare le seguenti proprietà: Profondità, Altezza, Immagine e Larghezza.

Profondità restituisce o imposta la profondità di immagine Quando impostando i valori di profondità, è necessario utilizzare solo i seguenti valori: 2, 8, 16 e 24. Viene dichiarato come:

PROPRIETÀ Profondità AS Integer

Altezza è un attributo di sola lettura che restituisce l'altezza dell'immagine come numero intero. È dichiarato come:

PROPRIETÀ LETTURA Altezza come numero intero

Larghezza è un attributo di sola lettura che restituisce la larghezza in pixel dell'immagine. È dichiarato come:

PROPRIETÀ LETTURA Larghezza AS Integer

Immagine (sola lettura) converte l'immagine in un'immagine e restituisce esso esso è dichiarato come:

IMMOBILE LEGGERE Picture AS Picture

La classe Image supporta diversi metodi. Questi includono: Cancella, Copia, Riempì, Capovolgi, Carica, Specchia, Sostitisci, Ridimensiona, Ruota, Salva e Allunga.

Chiaro cancellerà l'immagine, impostando tutti i valori dei pixel su zero.

copia restituisce una copia dell'immagine o una copia di una parte di essa. Il metodo di copia è dichiarato come funzione, come mostrato di seguito:

```
FUNZIONE Copia ([X AS Inter, Y AS Inter, Larghezza AS Inter, Altezza AS Inter]) AS Immagine
```

Riempire riempie l'immagine con un colore specificato. La sintassi del linguaggio Gamasas è:

Una guida per principianti a

SUB Fill (Colore come numero intero)

Una guida per principianti a

Flip restituisce una copia speculare dell'immagine. La copia speculare viene creata utilizzando l'asse orizzontale dell'immagine.

FUNZIONE Capovolgi () COME Immagine

Carica e salva carica o salva un'immagine da un file o in un file rispettivamente. L'estensione del file Path specificherà il formato del file grafico utilizzato con l'immagine salvata. I formati di file attualmente supportati sono JPEG, PNG, BMP, GIF e XPM. La sintassi del linguaggio Gambas è:

```
SUB Load (Path AS String)  
SUB Save (Path AS String)
```

Specchio restituisce una copia speculare dell'immagine. La copia speculare viene creata utilizzando l'asse verticale dell'immagine.

FUNZIONE Specchio () COME Immagine

Sostituire scambia un valore OldColor esistente con il colore specificato da NewColor, come mostrato di seguito:

```
SUB Sostituisci (OldColor AS Integer, NewColor AS Integer)
```

Ridimensiona Ridimensiona l'immagine alla dimensione specificata dal parametro di larghezza e altezza fornito:

```
SUB Resize (larghezza come numero intero, altezza come numero intero)
```

Ruotare restituisce una copia ruotata dell'immagine. La sintassi del linguaggio Gambas è la seguente:

FUNZIONE Ruota (Angolo AS Fluttuante) COME Immagine

La funzione ruoterà l'immagine di n gradi. L'angolo è specificato in gradi, non in radianti.

Allungare restituisce una copia allungata dell'immagine. Se il parametro facoltativo Smooth è specificato come True, al risultato restituito viene applicato un algoritmo di smoothing. La sintassi del linguaggio Gambas è:

```
FUNZIONE Allunga (larghezza come numero intero, altezza come numero  
intero [, liscia come valore booleano]) come immagine
```

Una guida per principianti a

La classe Picture disegna un'immagine o parte di un'immagine. Il contenuto dell'immagine viene archiviato nel server di visualizzazione, non nella memoria di processo come un'immagine. Anche se XWindow non gestisce ancora la trasparenza, ogni immagine può avere una maschera. Questa funzione può essere impostata esplicitamente durante l'istanza dell'immagine o implicitamente durante il caricamento di un file immagine con trasparenza come PNG. Quando si disegna su un'immagine con una maschera, l'immagine e la maschera vengono modificate di conseguenza. La sintassi del linguaggio Gambas è:

```
STATIC SUB Picture (Picture AS Picture, X AS Integer, Y AS Integer [,  
SrcX AS Integer, SrcY AS Integer, SrcWidth AS Integer, SrcHeight AS  
Integer])
```

Questa classe è creabile. Il codice seguente mostra come creare un'immagine:

```
DIM hPicture AS Picture  
  
hPicture = NEW Picture ([Width AS Integer, Height AS Integer, Transparent  
AS Boolean])
```

Se i parametri opzionali Larghezza e Altezza non sono specificati, la nuova immagine è nulla. È possibile specificare se l'immagine ha una maschera con il parametro Trasparente. Questa classe si comporta come un array.

```
DIM hPicture AS Picture  
  
hPicture = Immagine [Path AS String]
```

Il codice precedente restituirà un oggetto immagine dalla cache immagini interna. Se l'immagine non è presente nella cache, viene caricata automaticamente dal file specificato. Per inserire un'immagine nella cache delle immagini interna, utilizzare questo codice:

```
DIM hPicture AS Picture  
Immagine [Path AS String] = hPicture
```

Profondità restituisce o imposta la profondità dell'immagine. Quando si impostano i valori di profondità, è necessario utilizzare solo i seguenti valori: 2, 8, 16 e 24. Viene dichiarato come:

```
PROPRIETÀ Profondità AS Integer
```

Una guida per principianti a
Altezza è un attributo di sola lettura che restituisce l'altezza dell'immagine come numero intero. È dichiarato come:

Una guida per principianti a

PROPRIETÀ LETTURA Altezza come numero intero

Larghezza è un attributo di sola lettura che restituisce la larghezza in pixel dell'immagine. È dichiarato come:

PROPRIETÀ LETTURA Larghezza AS Integer

Immagine è una proprietà di sola lettura che converte l'immagine in un'immagine e la restituisce. È dichiarato come:

IMMOBILE LEGGI Immagine COME Immagine

Useremo un esempio molto semplice per mostrarti come funziona. Ti mostreremo come fare uno screenshot con il codice e salvare l'immagine. L'immagine del desktop viene presa come un oggetto immagine quando si fa clic sul pulsante nel modulo. Questo oggetto immagine viene successivamente convertito in un oggetto immagine e visualizzato come immagine in una DrawingArea. Avvia Gambas e crea un nuovo progetto di interfaccia utente grafica, denominato gfxDemo2. Rendi il progetto traducibile con controlli pubblici e quando viene visualizzato l'IDE, crea un modulo di classe di avvio denominato Form1. Dalla finestra delle proprietà per Form1, impostare la proprietà resize su True in modo da poter allungare il modulo per vedere l'immagine quando viene aggiunto. Aggiungere un pulsante denominato Button1 e un DrawingArea denominato DrawingArea1. Posiziona il DrawingArea nell'angolo in alto a sinistra del modulo e fallo largo circa un pollice e alto un pollice. Adesso, metti il pulsante nell'angolo in basso a destra del modulo. Aggiungi la didascalia "Instantanea" al pulsante. Fare doppio clic sul pulsante per creare un evento clic e aggiungere questo codice:

```
"File di classe Gambas

Pulsante PUBLIC SUB1_Click ()
'dichiara una var per la
nostra immagine p COME NUOVA
Immagine

'e uno per la nostra
immagine i COME NUOVA
immagine

'ottiene l'immagine object dal desktop usando il metodo
.Grab p = Desktop.Grab ()

'assegna l'immagine utilizzando la proprietà
dell'immagine per' convertire lo screenshot in
un'immagine
```

Una guida per principianti a
i = p. immagine
'specifica che il contenuto dell'area di disegno viene memorizzato nella
cache **Drawingarea1.Cached = TRUE**

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'ridimensiona l'area di disegno alle dimensioni dell'immagine fm  
convertita DrawingArea1.Resize (i.Width, i.Height)  
  
'cancella l'area di  
disegno  
DrawingArea1.Clear ()  
  
'chiama il metodo Draw.Begin (obbligatorio) per iniziare a  
disegnare Draw.Begin (DrawingArea1)  
  
'posiziona l'immagine nell'area di disegno a partire  
dall'origine 0,0 Draw.Image (i, 0, 0)  
  
'chiama il metodo Draw.End (obbligatorio) per smettere di  
disegnare Disegna Fine  
  
'rendi visibile l'immagine  
DrawingArea1.Visible = TRUE  
  
'aggiorna l'area di disegno per vedere cosa è stato fatto  
DrawingArea1.Refresh  
FINE
```

Salva il tuo lavoro ed esegui il programma. Fare clic sul pulsante e uno screenshot del desktop verrà posizionato nell'area di disegno del modulo. Abbastanza facile usare i metodi Gambas, non è vero?

Trasparente è un flag booleano che può essere letto o impostato. Indica se l'immagine ha o meno una maschera. La sintassi del linguaggio Gambas è:

PROPRIETÀ Trasparente AS Booleano

I metodi supportati da Picture includono Clear, Copy, Fill, Flush, Load, Resize e Save.

Chiaro cancellerà l'immagine, impostando tutti i valori dei pixel su zero.

copia restituisce una copia dell'immagine o una copia di una parte di essa. Il metodo di copia è dichiarato come funzione, come mostrato di seguito:

**FUNZIONE Copia ([X AS Intero, Y AS Intero, Larghezza AS Intero,
Altezza AS Intero]) AS Immagine**

Riempire riempie l'immagine con un colore specificato. La sintassi del linguaggio Gambas è:

Una guida per principianti a

```
SUB Fill (Colore come numero intero)
```

Sciacquone Svuota la cache delle immagini interna. Il metodo non accetta parametri e viene chiamato utilizzando questa convenzione:

```
STATIC SUB Flush ()
```

Carica e salva carica o salva un'immagine da un file o in un file rispettivamente. L'estensione del file Path specificherà il formato del file grafico utilizzato con l'immagine salvata. I formati di file attualmente supportati sono JPEG, PNG, BMP, GIF e XPM. La sintassi del linguaggio Gambas è:

```
SUB Load (Path AS String)  
SUB Save (Path AS String)
```

Ridimensiona ridimensiona l'immagine alla dimensione specificata dal parametro di larghezza e altezza fornito:

```
SUB Resize (larghezza come numero intero, altezza come numero intero)
```

Il metodo Tile disegna un'immagine affiancata nel file DrawingArea.Gambas la sintassi del linguaggio è la seguente:

```
STATIC SUB Tile (Picture AS Picture, X AS Integer, Y AS Integer,  
Larghezza AS Integer, Altezza AS Integer)
```

Torniamo ora al nostro programma gfxDemo originale e finiamo di codificare l'ultimo pulsante. Avvia Gambas e carica il progetto gfxDemo. Apri Form1 e fai doppio clic sul pulsante TileImg. Quando viene visualizzata la finestra del codice, aggiungi questo codice:

```
PUBLIC SUB TileBtn_Click ()  
    ' dichiara qui i nostri vars  
    locali  
    ' avremo bisogno di una variabile immagine per contenere  
    ' l'immagine che carichiamo DIM mypic AS Picture  
  
    ' abbiamo bisogno di un counter var per  
    ' il nostro lavoro sul loop Contatore DIM  
    AS Integer  
  
    ' avremo anche bisogno di alcune variabili  
    ' intere DIM i AS Integer  
    ' DIM j AS Numero intero
```

Una guida per principianti a
'imposta i e j a zero per
iniziare **io = 0**

Una guida per principianti a

```
j = 0

'istanzia la variabile dell'immagine, imposta la larghezza,
l'altezza in modo che non sia nulla mypic = NUOVA immagine (170,
105, FALSE)

'cancella l'area di
disegno da.Clear

'ora carica l'immagine usando il metodo di caricamento
mypic.Load ("Gambas Shrimp.png")

'affiancheremo le colonne dell'immagine i per j righe verso
il basso' nel nostro caso avremo 4 righe di 3 immagini per
riga
'quindi dovremo ripetere 12 volte
Contatore FOR = da 0 a 11

'disegna la tessera
draw.Tile (mypic, i * 175, j, 170,105)

'incrementa il nostro
contatore di colonne INC i

'controlla per vedere se ne abbiamo
tre SE io MOD 3 = 0 ALLORA

'in tal caso, sposta il valore dell'immagine j (il nostro asse
y). altezza + 5 pixel verso il basso j = j + 110

'reimposta il contatore della
colonna a zero io = 0
FINISCI SE 'il nostro controllo per
vedere se 3 attraverso IL PROSSIMO
'iterazione del ciclo

'aggiorna il display per vedere il
lavoro da.Refresh
FINE 'del codice del pulsante TileImg
```

L'ultima cosa che dobbiamo fare è ottenere un'immagine da utilizzare per il nostro programma. Per il nostro programma, ho scelto la mascotte Gambas, che abbiamo utilizzato da un progetto precedente. Puoi copiare questa immagine dal nostro FirstProject. Dalla finestra del progetto, nella cartella TreeView Data, selezionare New Image. Quando appare la finestra, puoi selezionare la scheda Esistente e navigare fino alla cartella FirstProject (supponendo che tu abbia chiamato i tuoi progetti come specificato nel libro) e scegliere l'immagine "Gambas mascot.png". Salvalo nel progetto gfxDemo come "Gambas shrimp.png" e questo è

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a tutto quello che c'è da fare. Se, per qualche motivo, non riesci a trovare l'immagine sul tuo sistema, vai semplicemente al sito web di Gambas e usa il tasto destro del mouse sull'immagine della mascotte per salvarne una copia nella cartella del nostro progetto gfxDemo. Ora salva il tuo lavoro

Una guida per principianti a

ed esegui il file programma Fare clic su il pulsante TileImg e dovresti vedere qualcosa di simile a questo:

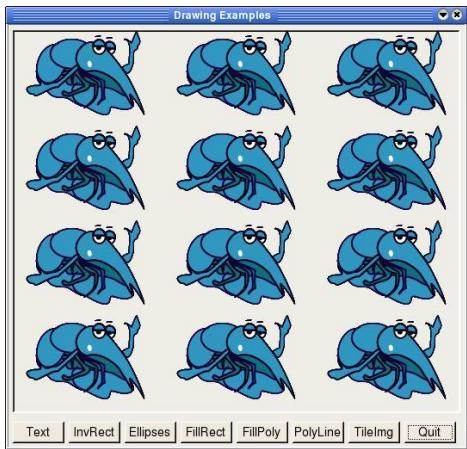


Figura 84 Utilizzo di immagini affiancate con il pulsante TileImg del nostro programma demo.

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. This software is released under the terms of the GNU General Public License (GPL). It is not build with the autoconf tool.

Disegnare con un oggetto Drawing

In Gambas, un oggetto di disegno si basa sull'uso di Scalable Vector Graphics, o SVG. Secondo la specifica SVG16, "SVG è un linguaggio per descrivere la grafica bidimensionale in XML. SVG consente tre tipi di oggetti grafici: forme grafiche vettoriali (ad esempio, tracciati costituiti da linee rette e curve), immagini e testo. Gli oggetti grafici possono essere raggruppati, stilizzati, trasformati e composti in oggetti renderizzati in precedenza. Il set di funzionalità include trasformazioni nidificate, tracciati di ritaglio, maschere alfa, effetti filtro e oggetti modello. I disegni SVG possono essere interattivi e dinamici. Le animazioni possono essere definite e attivate in modo dichiarativo (ovvero incorporando elementi di animazione SVG nel contenuto SVG) o tramite script."

Inkscape¹⁷ è un popolare programma utilizzato nella comunità Linux per creare immagini SVG ed è ciò che questo autore ha scelto di usare con questo libro poiché è disponibile gratuitamente su Internet. Inkscape è un editor di grafica vettoriale open source, con funzionalità simili a Illustrator, Freehand, CorelDraw o Xara X utilizzando il formato di file SVG (Scalable Vector Graphics) standard W3C.

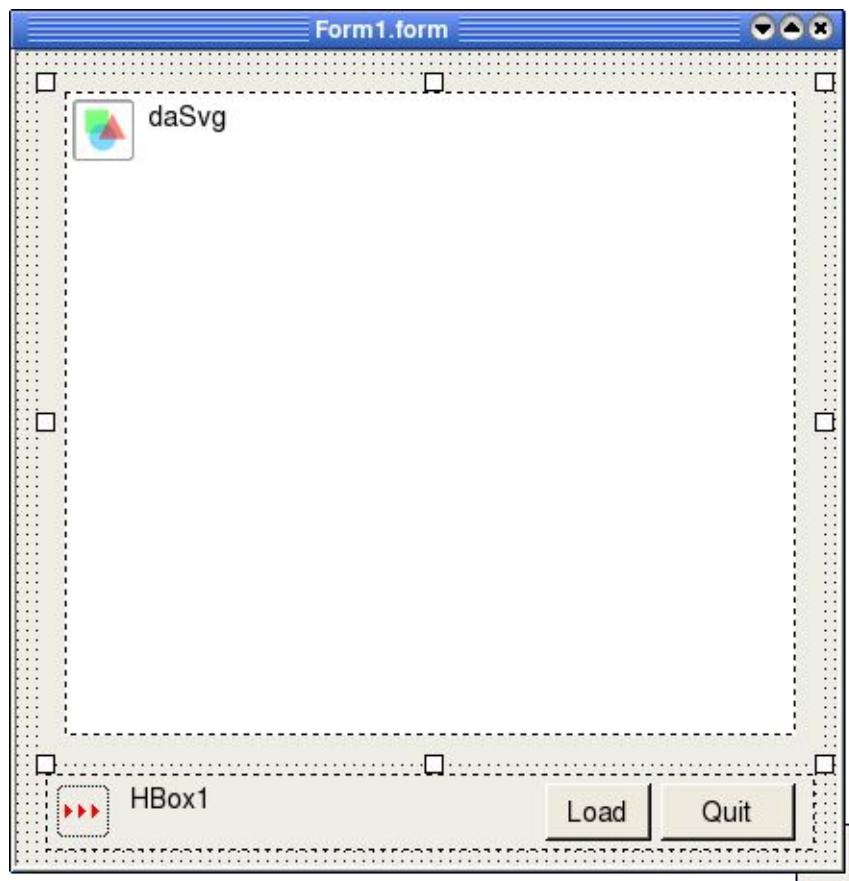
In Gambas puoi caricare, salvare, cancellare o copiare un SVG immagine

16 See URL <http://www.w3.org/TR/SVG/intro.html> for mori dettagli on SVG.

17 Vedi URL <http://www.inkscape.org/> per maggiori dettagli su Inkscape.

Una guida per principianti a

le proprietà che hai a disposizione sono le più elementari, ovvero larghezza, altezza e profondità. Gambas non è progettato per essere un editor SVG. Ha semplicemente delle disposizioni per usare le immagini SVG nel tuo programma. Una volta che hai il disegno nel tuo programma, sta a te farci qualcosa. Ad esempio, creeremo un nuovo programma per caricare semplicemente un'immagine SVG e visualizzarla. Per prima cosa, dovremo trovare un'immagine SVG. Ho scaricato un'immagine con licenza di dominio pubblico disponibile gratuitamente da Internet all'indirizzo <http://openclipart.org> da utilizzare per la nostra demo. Puoi scegliere qualsiasi immagine desideri per questa demo, purché sia un file in formato SVG. Ora creeremo un nuovo progetto di interfaccia utente grafica in Gambas chiamato gfxDemo3. Rendi pubblici e traducibili i controlli del progetto. Successivamente, crea un nuovo modulo di classe di avvio, denominato Form1. Aggiungeremo diversi controlli: Scrollview, DrawingArea, HBox e due pulsanti. Il tuo modulo dovrebbe assomigliare a questo in modalità progettazione:



© 2006 W. Rittinghouse, all rights are reserved. It is released to the public domain under an Open Content License (OCL) and may not be distributed or modified without the express written consent of the author.

Fare doppio clic sul modulo quando si apre e viene visualizzata la finestra del codice. Inizia con il controllo ScrollView, nominandolo ScrollView1.Next, aggiungere un

Una guida per principianti a

DrawingArea nella parte superiore del controllo ScrollView. Assegna un nome a DrawingArea daSVG. Aggiungi l'HBox, denominato HBox1 ei due pulsanti denominati LoadBtn e QuitBtn. Non dimenticare di aggiungere le didascalie di testo ai pulsanti. Quindi, fai doppio clic sul pulsante Esci e aggiungi questo codice:

```
PUBLIC SUB QuitBtn_Click  
    () ME.Close  
FINE
```

Ora, fai doppio clic su LoadBtn e aggiungi questo:

```
PUBLIC SUB LoadBtn_Click ()  
  
    'dichiara il nostro oggetto di disegno per il  
    file SVG DIM dra AS Drawing  
  
    'istanziarlo dra  
    = NUOVO disegno  
  
    'ora caricalo usando il metodo  
    Load dra.Load ("floppy.svg")  
  
    'imposta la proprietà memorizzata nella cache su TRUE per  
    memorizzarla nella cache daSvg.Cached = TRUE  
  
    'imposta la proprietà visible su TRUE così possiamo vedere  
    l'immagine daSvg.Visible = TRUE  
  
    'sposta l'oggetto dell'area di disegno in modo che inizi dall'origine in  
    alto a sinistra (0,0) dasvg.Move (0,0, Dra.Width, Dra.Height)  
  
    'chiama il sorteggio obbligatorio, inizia a iniziare a  
    disegnare draw.Begin (daSvg)  
  
    'fai il sorteggio  
    draw.Drawing (dra,  
    0,0)  
  
    'chiama il metodo Draw.End per uscire dal  
    disegno draw.end  
  
    'aggiorna il nostro modulo per vedere  
    cosa è stato fatto Form1.Refresh  
FINE
```

Dopo aver scaricato o creato il file floppy.svg, salvalo nella directory del progetto. Questo è tutto il codice che dobbiamo aggiungere per visualizzare un file

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the public under the terms of the Microsoft Public License (MSPL). You may not redistribute this product in whole or in part. This includes the source code, object code, and any derivative works, but does not include those documents which are separate from the source code, such as help files.

Una guida per principianti a
SVG. Salva

Una guida per principianti a

il tuo lavoro ed esegui il programma. All'avvio del programma, fai clic sul pulsante Carica e dovresti vedere un'immagine simile a questa:

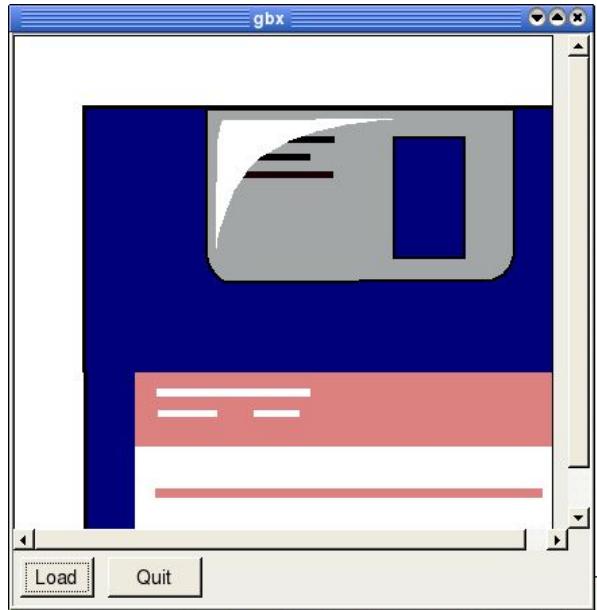


Figura 85 Caricamento di un file SVG in Gambas.

Spero che abbiate apprezzato il nostro breve "tour de force" delle capacità di disegno di Gambas. Questo capitolo ha presentato tutte le basi necessarie per usare Gambas per quasi tutte le esigenze legate alla grafica. Alla fine di questo libro, nell'Appendice A, presentiamo uno speciale programma di grafica che puoi usare per disegnare oggetti di rendering 2D / 3D ancora più avanzati usando una classe grafica GKStype creata in Gambas.

This product is (C) 2005 by John W. Rittinghouse, a trademark reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Contenuto del file SVG "floppy.svg" scaricato da <http://openclipart.org>:

```
<? xml version = "1.0" encoding = "UTF-8" standalone = "no"?>
<! DOCTYPE Svg PUBBLICO - // W3C // DTD SVG 20010904 // EN "
"http: //www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd ">
<! - Creato con Sodipodi ("http://www.sodipodi.com/") -> <svg height = "400pt" id = "svg548" sodipodi: docbase
= "/ home / nicu / svg_gal / computers / sodipodi: docname = "/ home / nicu / svg_gal / computers /
floppy.svg" sodipodi: version = "0.32" width = "400pt" xmlns = "http://www.w3.org/2000/svg" xmlns: sodipodi
="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd" xmlns: xlink = "http://www.w3.org/1999/xlink">
<metadata>
<rdf: RDF xmlns: cc = "http://web.resource.org/cc/" xmlns: dc =
"http://purl.org/dc/elements/1.1/" xmlns: rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#
" />
<cc: Work rdf: about = "">
<dc: title> Clipart by Nicu Buculei - unntenna </ dc: title>
<dc: description> </ dc: description>
<dc: subject>
<rdf: Borsa>
<rdf: li> hash </ rdf: li>
<rdf: li> </ rdf: li>
<rdf: li> computer </ rdf: li>
</ rdf: Borsa>
</ dc: subject>
<dc: publisher>
<cc: agente rdf: about = "http://www.openclipart.org">
<dc: title> Nicu Buculei </ dc: title>
</ cc: agente>
</ dc: publisher>
<dc: creator>
<cc: agente>
<dc: title> Nicu Buculei </ dc: title>
</ cc: agente>
</ dc: creator>
<dc: rights>
<cc: agente>
<dc: title> Nicu Buculei </ dc: title>
</ cc: agente>
</ dc: rights>
<dc: date> </ dc: date>
<dc: format> image / svg + xml </ dc: format>
<dc: type rdf: resource = "http://purl.org/dc/dcmitype/StillImage"/>
<cc: licenza rdf: resource = "http://web.resource.org/cc/PublicDomain"/>
<dc: language> it </ dc: language>
</ cc: Work>
<cc: Licenza rdf: about = "http://web.resource.org/cc/PublicDomain">
<cc: permessi rdf: risorsa = "http://web.resource.org/cc/Reproduction"/>
<cc: permessi rdf: risorsa = "http://web.resource.org/cc/Distribution"/>
<cc: permessi rdf: risorsa = "http://web.resource.org/cc/DerivativeWorks"/>
</ cc: License>
</ rdf: RDF>
</metadata>
<defs id = "defs550" />
<sodipodi: namedview id = "base" showgrid = "true" />
<path d = "M 52.5712 53.9929 L 407.003 53.9929 L 443.463 89.6056 L 441.768 447.429 L 53.4191 445.733 L 52.5712
53.9929 z" id = "path551" style = "fill: # 00007b; fill-rule: evenodd; stroke: black; stroke-opacity: 1; stroke-width: 3.75;
stroke-linejoin: mitre; stroke-linecap: butt; riempimento-opacità: 1; tratto-dasharray: nessuno; "/>
<path d = "M 371,39 53,9929 L 371,39 145,569 C 371,39 145,569 368,846 159,135 355,279 163,375 C 341,713 164,222
156,866 164,223 156,866 164,223 C 156,018 165,071 144,995 157,439 144,147 152,352 C 143,299 147,264 142,451
145,569 142,451 145,569 144,147 53,145 143,299 53,145 C 142,451 53,145 372,238 53,9929
371,39 53,9929 z" iod = "path552" sodipodi: nodetypes = "cccccccc" style = "fill: # a3a6a6; fill-rule: evenodd; stroke:
black; stroke- opacity: 1; stroke-width: 2.32865; stroke-linejoin: mitre; stroke-linecap: butt; fill-opacity: 1;
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the
Gambas User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
stroke-dasharray: nessuno; "

Una guida per principianti a

```
transform = "matrix (1.000000,0.000000,0.000000,1.152582,0.000000, -6.413116)" />
<rect height = "185.69497" id = "rect553" style = "font-size: 12; fill: #ffffff; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "321.362518" x = "89.879768" y = "249.862976" />
<rectheight = "9.32714844" id = "rect554" style = "font-size: 12; fill: # 808080; fill-rule: evenodd; stroke-width: 1pt;" larghezza = "0.00000000" x = "410.394348" y = "240.535797" />
<path d = "M 334.929 76.0389 C 334.081 76.0389 282.358 76.0389 282.358 76.0389 L 282.358 165.071 L 334.929 165.071 L 334.929 76.0389 z" id = "path555" Style = "fill: # 00007b; fill-rule: evenodd; stroke: black; stroke-opacity: 1; stroke-width: 2.5; stroke-linejoin: mitre; stroke-linecap: butt; fill-opacity: 1; stroke-dasharray: nessuno;" />
<rect height = "55.1149292" id = "rect556" style = "font-size: 12; fill: # dd8080; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "321.362549" x = "89.8797607" y = "249.862946" />
<rectheight = "6.783386" id = "rect557" style = "font-size: 12; fill: # dd8080; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "287.445648" x = "105.142365" y = "335.503052" />
<rectheight = "5.935455" id = "rect558" style = "font-size: 12; fill: # dd8080; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" width = "287.445648" x = "104.294426" y = "371.115822" />
<rectheight = "5.935455" id = "rect559" style = "font-size: 12; fill: # df8080; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" width = "286.597717" x = "105.142365" y = "406.728516" />
<rectheight = "5.08750916" id = "rect567" style = "font-size: 12; fill: # 000000; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" width = "78.8567657" x = "159.409393" y = "75.1909637" />
<rectheight = "5.08754730" id = "rect568" style = "font-size: 12; fill: # 000000; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "58.5066528" x = "161.105225" y = "87.9097748" />
<rectheight = "5.08750916" id = "rect569" style = "font-size: 12; fill: # 180003; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "76.3130035" x = "161.105225" y = "100.628624" />
<rectheight = "5.93548584" id = "rect570" style = "font-size: 12; fill: #ffffff; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" width = "114.469490" x = "105.990288" y = "261.733826" />
<rectheight = "5.08752441" id = "rect571" style = "font-size: 12; fill: #ffffff; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "44.0919495" x = "106.838211" y = "277.844360" />
<rectheight = "5.087512" id = "rect572" style = "font-size: 12; fill: #ffffff; fill-rule: evenodd; stroke-width: 1pt; fill-opacity: 1;" larghezza = "33.916901" x = "178.063675" y = "277.844386" />
<path d = "M 150.317 158.228 C 151.899 153.481 148.734 63.2911 155.063 63.2911 C 161.392 63.2911 292.722 61.7089 292.722 61.7089 C 232.595 68.038 164.557 66.4557 150.317 158.228 z" id = "path566" sodipodi:nodetypes = "cccc" style = "fill: #ffffff; fill-rule: evenodd; stroke: none; stroke-opacity: 1; stroke-width: 1pt; stroke-linejoin: mitre; stroke-linecap: butt; fill-opacity: 0.5;" />
</svg>
```

Una guida per principianti a

Capitolo 13 - Gestione degli errori

La gestione degli errori è un'area chiave per gli sviluppatori che svolgono qualsiasi tipo di sviluppo di applicazioni. La gestione degli errori è un concetto molto potente e può rendere il lavoro di sviluppo molto più semplice se utilizzato in modo efficiente. In qualità di sviluppatore Gambas, ti consigliamo vivamente di adottare una strategia appropriata per la gestione degli errori che ti consentirà di creare applicazioni robuste e di alta qualità. La gestione impropria degli errori può e ridurrà le prestazioni dell'applicazione, la sua percezione da parte della comunità di utenti e, in definitiva, la sua accettazione complessiva. Per comprendere la gestione degli errori, è importante conoscere i tipi di errori che possono verificarsi in un programma. Spiegheremo brevemente le categorie di errore più comuni nelle sezioni seguenti e proveremo a sottolineare alcune cose che dovresti osservare mentre scrivi il tuo codice.

Concetti generali di gestione degli errori

Gli errori possono avere molte cause. A seconda della causa, l'errore potrebbe essere recuperabile. Un errore recuperabile è quello per cui l'applicazione può identificare la causa dell'errore e intraprendere alcune azioni per risolvere o mitigare il problema. Alcuni errori potrebbero essere recuperabili possono essere gestiti senza la necessità di indicare all'utente che si è verificato un errore. Ad esempio, un errore generato quando un file richiesto non esiste è recuperabile. L'applicazione può essere programmata per visualizzare un messaggio di errore in tale evento. Errori come errori di convalida, per i quali l'applicazione non può continuare l'elaborazione ma può fornire un feedback specifico dell'errore possono anche essere considerati recuperabili se gestiti correttamente. Per esempio,

Gestione degli errori

Ci sono tre fattori chiave da considerare quando si programma per evitare errori: prevenzione, rilevamento e ripristino. L'adozione di misure adeguate per prevenire gli errori contribuirà sicuramente a garantire una comunità di utenti felice. Tuttavia, come si evitano gli errori se non possono essere rilevati dal programma prima che l'utente ne subisca le conseguenze? Tu e il tuo team avete impiegato del tempo per indagare e tenere conto di ogni possibile sequenza di tasti, input e tipo di dati che un utente potrebbe inserire? Pensaci davvero mentre sei ancora nella fase di progettazione e di nuovo

Una guida per principianti a

durante lo sviluppo iniziale dei piani di test perché può tornare a perseguitarti per molte, molte ripetizioni di build e test. Infine, se l'errore si verifica, cosa si può fare per riportare il programma allo stato di usabilità in cui si trovava prima che si verificasse l'errore?

La maggior parte degli errori è il risultato di una progettazione inadeguata, pratiche di programmazione inadeguate e test inadeguati. Le cattive pratiche di programmazione includono la codifica solo del metodo per ottenere i risultati desiderati senza tenere conto di errori dell'utente, errori di input, errori di convalida, ecc. Tutti i programmatore sanno come sviluppare controlli di convalida dei dati e di solito cercano di rimandarli fino a quando il codice funziona . Rendere la convalida dei dati un requisito aumenta la priorità di eseguire questo lavoro di una tacca nell'elenco delle cose che il programmatore deve fare. I test inadeguati sono rappresentati dal fatto che i team di test non tengono conto di questi tipi di fattori nella progettazione e costruzione dei piani di test del prodotto e dei casi di test.

BoundaryRelated Errori

Quando un programma viene eseguito, il cursore del programma si sposta da una parte del programma a quella successiva eseguendo sequenzialmente le istruzioni codificate. Il passaggio da un'area all'altra nel programma è definito dall'uscita di una condizione al contorno e dall'ingresso a quella successiva. Questo stato di transizione, o confine, può includere limiti numerici, di testo e operatori. Se all'utente viene chiesto di inserire un numero da 0 a 99 in un programma, cosa succede se l'utente inserisce la lettera "B" o digita invece 123? Cosa succede se l'utente inserisce il numero 84 e il sistema ti dice che il numero è troppo grande, chiedendo all'utente di provare a inserire di nuovo un numero da 0 99? Il programma non riesce a convalidare l'input e tenta di elaborare un input errato? Tutte le possibili condizioni al contorno sono state prese in considerazione nella progettazione e nei test? Veramente?

Errori di calcolo

Esistono un numero infinito di condizioni che possono causare errori di calcolo. La logica del programmatore potrebbe essere stata errata o la sua formula per il calcolo potrebbe essere codificata in modo errato. Gli errori di arrotondamento o troncamento possono alterare in modo significativo i risultati desiderati. Un buon processo di progettazione dovrebbe incorporare uno pseudocodice per tutti i risultati calcolati per mostrare come queste formule devono essere implementate o utilizzate nella produzione. Le revisioni del codice controllerebbero che siano effettivamente implementate come previsto. I casi di test

Una guida per principianti a eseguono tutte le condizioni limite rispetto al codice e verificano che l'implementazione sia quanto più possibile solida prima di passare alla produzione per uso generale.

Una guida per principianti a

Stati iniziali e successivi

Quando un programma esegue subroutine e utilizza variabili, queste variabili spesso devono essere inizializzate con valori predefiniti. In Gambas, tali valori predefiniti assumono generalmente la forma di un valore zero o un valore NULL. Se l'inizializzazione non viene eseguita specificatamente da uno sviluppatore, nessuno può immaginare quale sarà il valore di quella variabile quando viene chiamata per la prima volta, a meno che la variabile non sia uno dei tipi di dati che Gambas inizializza automaticamente. Ricorda, questi tipi di errori sono il risultato di programmatore che non eseguono correttamente il lavoro. Ad esempio, supponiamo che nell'esempio precedente non sia stata eseguita alcuna inizializzazione di variabile quando una routine richiede l'incremento di una variabile. Il programma ha elaborato 50 record, incrementando il valore della variabile ogni volta che viene eseguita la routine. Quando il valore viene emesso alla fine dell'elaborazione,

Per dimostrare errori di stato successivi, si consideri la stessa situazione di cui sopra, ma invece di passare il valore atteso di 50 a un altro modulo da utilizzare per l'elaborazione, viene passato il valore di 172. Questi errori sono un po' più difficili da rilevare perché l'output dal modulo chiamato è dove l'utente pensa che si sia verificato l'errore. I programmatore incaricati di eseguire il debug di questo problema devono tracciare il percorso di esecuzione e seguire il cambiamento dello stato della variabile fino a quando non si scopre che è un problema nel modulo chiamante, non in quello chiamato. Il risultato di ciò è una grande quantità di tempo e impegno per scoprire un problema che era evitabile. Se l'utente ha segnalato ciò da un sistema di produzione, la credibilità del team di progetto subisce nuovamente un colpo. In qualità di manager del programma, devi assicurarti che questi tipi di situazioni siano prevenute facilitando

Errori di flusso di controllo

In precedenza, abbiamo affermato che quando un programma viene eseguito, il cursore del programma si sposta da una parte all'altra del programma, eseguendo le istruzioni codificate in sequenza. Alcune istruzioni di esecuzione costringono il programma a passare da una routine all'altra. Questo movimento in esecuzione è indicato come programma o flusso di controllo. Quando un programma fa qualcosa di sbagliato nel flusso di controllo, è generalmente un problema con la logica del programma o un errore del programmatore nella codifica del progetto. Tali errori logici vengono spesso scoperti durante le revisioni del codice nella fase di implementazione. A volte, durante i test, vengono scoperti costruendo test

Una guida per principianti a

casi per convalidare il flusso del programma sulla base delle specifiche di progetto costruite in fase di progettazione e dei piani di test costruiti nella fase di implementazione. Uno qualsiasi di questi tipi di errori scoperti da un utente quando un sistema è in produzione riflette un fallimento del team di test nel testare adeguatamente il prodotto. In qualità di responsabile del progetto, sei responsabile in ultima analisi di impedire che questa situazione si verifichi. PROVA a lavorare con il team di progettazione e gli sviluppatori dei casi di test il più presto possibile nel processo per stabilire le aspettative adeguate per il tipo di test e il livello di accuratezza che ti aspetti.

Errori nella gestione o interpretazione dei dati

Ci sono molti modi in cui i dati possono diventare pieni zeppi di errori in un processo di sviluppo. Tuttavia, quasi tutte queste miriadi di modi in cui i dati possono andare storte sono attribuite all'errore umano. Questi tipi di errori possono essere il risultato di calcoli errati, errori nella logica, errori di passaggio di parametri, utilizzo di tipi di dati errati, ecc. Quasi tutto ciò che un programmatore può fare per commettere un errore può causare un errore nei dati. Il modo migliore per prevenire tali errori è un controllo continuo della logica, la revisione dei valori attesi ed effettivi, la revisione dei dati di input e output, ecc. Questo è un processo continuo e coinvolge sia l'utente che lo sviluppatore.

Dal punto di vista dell'utente, i dati dovrebbero essere confrontati con i mezzi attuali di produzione dei dati. L'utente dovrebbe rivedere continuamente i risultati dell'output per assicurarsi che qualcosa non sia andato in conflitto e abbia causato il deterioramento dei dati. Dal punto di vista dello sviluppatore, dovrebbe essere applicata la disciplina di base della codifica; è necessario controllare le condizioni dell'intervallo e al contorno, i valori parametrici devono essere convalidati, deve avvenire la convalida del tipo di dati e l'uso dei dati di prova deve essere convalidato per essere del formato corretto, corrente e in sincronia con i dati di produzione e utilizzabile per l'output di produzione. È sempre positivo per i responsabili del team rafforzare questi concetti con gli sviluppatori e garantire che vengano eseguite revisioni periodiche del codice. È essenziale per il successo della squadra.

Condizioni di gara e carico

Una condizione di gara è meglio definita come un evento che precede un altro. Il trucco è assicurarsi che la prima condizione sia la prima o che l'ultima sia l'ultima. Pochi programmati pensano o addirittura controllano le condizioni di gara. Queste condizioni a volte vengono identificate solo dopo che è stata impiegata una grande manodopera per fare tracce in codice eseguibile, passo dopo passo, per

Una guida per principianti a vedere cosa sta succedendo. Quando senti le parole irreproducibili mentre i tester parlano di questo tipo di bug, interpretale come una possibile condizione di gara e chiedi a qualcuno se

Una guida per principianti a
l'hanno verificato.

Le condizioni di carico sono comuni e sono il risultato di più lavoro sul computer di quanto possa sopportare. Un semplice punto di illustrazione è avere un computer che elabora milioni di record. È il lavoro perfetto per un computer ma, quando il tempo necessario per elaborare ogni record supera i pochi millisecondi, il milione di record da elaborare può richiedere molto tempo. Ad esempio, supponiamo una situazione ideale in cui la tua applicazione esegua una convalida complessa, necessaria per elaborare correttamente un record e che siano necessari 10 millisecondi per elaborare ciascuno di questi record in un ambiente di test perfettamente gestito. Ciò si traduce in una velocità di elaborazione di 100 record al secondo. Non male, ma quando dividi 100 in 1.000.000 di record, scopri rapidamente che questo lavoro richiederà 10.000 secondi o 2,7 ore per essere completato. Ancora, accettabile nella maggior parte dei casi, tranne per il fatto che è necessario ricordare che la macchina è completamente impegnata con l'elaborazione. Non è possibile produrre report, eseguire query, ecc. Nell'ambiente informatico moderno di oggi, ciò non è accettabile.

Diamo un altro punto di vista in cui vengono applicate le condizioni del mondo reale. In un ambiente di test, dove i 10 millisecondi sono condizioni di calcolo ideali, aggraveremo il problema con l'interazione dell'utente e il sovraccarico del sistema. Esaminiamo i tempi di recupero dei record e li calcoliamo nell'equazione. I tuoi 10 millisecondi ora sono 14,5 millisecondi per record. Le query degli utenti e altre interazioni con gli utenti potrebbero facilmente aggiungere altri 10 o più di lì millisecondi per attività e ora siamo a 25 millisecondi di tempo di elaborazione per un solo record. Ora stiamo valutando 6,75 ore per eseguire lo stesso lavoro eseguito in 2,7 ore nel laboratorio di test. Il punto qui è che i test dovrebbero tenere conto di tali condizioni di carico prima del rilascio in produzione. Se un utente sa prima di avviare un lavoro di produzione che ci vorranno 7 ore per essere eseguito, possono andarsene e fare altre cose o eseguire il lavoro di notte. Tuttavia, se l'aspettativa è un lavoro di 2,5 ore, saranno sempre insoddisfatti delle prestazioni perché non è quello che si aspettavano.

Problemi di piattaforma e hardware

I problemi hardware sono tra i più frustranti che un utente possa incontrare. Un esempio potrebbe essere la selezione di un utente per stampare un rapporto e guardarla scorrere sullo schermo, a un milione di parole al secondo. Il dispositivo di stampa non è stato specificato correttamente o non è stato specificato affatto. L'utente pensava che una stampante avrebbe stampato il rapporto ma gli errori del dispositivo o la mancanza di gestione del dispositivo hanno causato la situazione di

Una guida per principianti a cui sopra. Esistono molti tipi di dispositivi di cui un programmatore deve tenere conto e il processo di test deve tenere conto dell'interazione dell'utente su tutti loro. A parte

Una guida per principianti a

la condizione sopra citata, i programmi possono inviare i codici sbagliati al dispositivo giusto, sovraccaricare il dispositivo con i dati, ecc. I programmatori e i tester sono responsabili di garantire che queste condizioni siano prese in considerazione e testate prima che la tua comunità di utenti ne sia esposta. Tutti questi tipi di errori possono fare è darti un brutto voto di credibilità mentre i tuoi utenti valutano le tue prestazioni.

Errori di controllo di origine, versione e ID

Questi tipi di errori sono generalmente dovuti a una cattiva gestione della configurazione e alla mancanza interna di applicazione delle politiche e delle procedure di gestione della configurazione esistenti. Un esempio di ciò è avere un nuovo programmatore che inserisce una versione beta del codice in una versione stabile di produzione per una normale correzione di bug e quindi non dice a nessuno che è stata apportata una modifica in quel modulo. Il normale test della versione di manutenzione probabilmente non individuerà il problema e potrebbe passare inosservato per diversi rilasci successivi.

Come altro esempio, questa situazione potrebbe essere qualcosa di insidioso come un errore di dati che si manifesta solo quando vengono prodotti i dati finanziari trimestrali. Quando non si trova alcun motivo per spiegare perché il bilancio della società è sfasato di diverse centinaia di migliaia di dollari, viene speso molto impegno per far eseguire agli sviluppatori senior le tracce del codice per trovare un problema. Qualcuno alla fine scoprirà che c'è del codice beta in un modulo di produzione che ha arrotondato tutto in modo errato al dollaro più vicino o qualche altro stupido errore simile. Buone intenzioni a parte, la migliore cura per questi tipi di problemi è una buona disciplina del programmatore per la gestione della configurazione e per avere una forte applicazione delle politiche radicata nella cultura aziendale dell'organizzazione.

Errori di test

Anche i tester commettono errori. Possono interpretare la logica di un test case in modo improprio, possono interpretare erroneamente i risultati dei test, non riuscire a segnalare glitch, ecc. I tester possono dimenticare di eseguire test case o segnalare bug che in realtà non esistono. Anche queste persone sono umane. Ciò significa che anche i loro risultati sono soggetti a domande. Quando i tuoi team hanno lavorato duramente per sviluppare qualcosa e i risultati del test tornano distorti in un modo o nell'altro, metti in discussione il processo del test.

Revisioni del piano di test

Una guida per principianti a

Per tutti i tipi di progetto tranne i più semplici, è generalmente richiesto un processo di revisione formale. Ciò richiede anche la partecipazione degli utenti, il

Una guida per principianti a

sviluppatori, analisti di sistemi, analisti di dati e tester. È meglio per tutti gli interessati se la revisione formale del test include gli utenti e gli sviluppatori più esperti presenti in un'organizzazione. Alcuni utenti potrebbero non avere l'esperienza adeguata con un sistema per identificare correttamente i problemi prima che si verifichino e questa esperienza aggiuntiva può solo avvantaggiare un processo di revisione. Garantire che la partecipazione a una revisione formale del piano di test si basi meno sulla disponibilità di utenti e personale degli sviluppatori e più sui contributi che gli sviluppatori e gli utenti più qualificati possono portare al tavolo. Ora che abbiamo una buona comprensione del come e del perché degli errori, parliamo di come possiamo gestire tali errori in Gambas.

Gestione degli errori di Gambas

Gambas fornisce meccanismi incorporati per assistere il programmatore nella gestione degli errori. Tali meccanismi sono strumenti a disposizione di un programmatore che vengono spesso trascurati e, in molti casi, non utilizzati affatto. Gambas è stato progettato con in mente i concetti di semplicità e di evitare la logica errante. I programmi Gambas che scrivi dovrebbero cercare di aderire a questi concetti utilizzando tutti i mezzi disponibili per evitare errori. Ciò significa prendere il tempo per scrivere routine di convalida, lavorare sul codice per tenere traccia del flusso e fornire log di output, ecc. Non è molto più lavoro e farà un'enorme differenza a lungo termine. Gambas può solo fornire un ambiente in cui svilupparsi, ma, alla fine, sei tu, il programmatore, ad essere responsabile del tuo prodotto. Dedichiamo un po 'di tempo ora per imparare a utilizzare le funzionalità di Gambas per la gestione degli errori.

PROVARE *dichiarazione...* SE ERRORE

Il comando TRY fornisce al programmatore i mezzi per eseguire un controllo degli errori ovunque sospetti che possa verificarsi un errore. È, in effetti, un modo per testare l'acqua prima di saltare dentro. Puoi provare a eseguire l'istruzione e, se si verifica un errore, verrà registrato dalla classe Error. Questo viene fatto impostando i valori delle proprietà che puoi controllare. Questi valori di proprietà includono:

- ✓ Error.Class
- ✓ Codice di errore
- ✓ Error.Text
- ✓ Errore, dove

La proprietà Error.Class restituirà il nome della classe in cui si è verificato

Una guida per principianti a l'errore e la proprietà Error.Code indicherà qual era il codice di errore specifico. Il testo associato a Error.Code è memorizzato in Error.Text

Una guida per principianti a

e la proprietà Error.Where ti dirà la riga offensiva del codice sorgente (se possibile). La classe error fornisce anche due metodi, Error.Clear ed Error.Raise, che puoi utilizzare per rintracciare i problemi. Discuteremo l'uso di questi due metodi un po' più avanti in questo capitolo.

L'istruzione TRY viene utilizzata per verificare errori specifici mentre l'istruzione CATCH (descritta di seguito) è più un meccanismo di rilevamento degli errori di uso generale. La logica generale utilizzata per TRY è mostrata nello pseudocodice seguente:

```
PROVA una dichiarazione
  se ERRORE ALLORA
    Message.Error
    (error.text) endif
  fine del tentativo
```

Creiamo un rapido esempio per dimostrare come utilizzare try. Apri l'IDE Gambase crea un nuovo progetto di interfaccia utente grafica. Crea un modulo di classe di avvio e inserisci un pulsante Esci nel modulo, denominato ExitBtn. Fare doppio clic sul pulsante e aggiungere questo codice:

```
PUBLIC SUB ExitBtn_Click
  () DIM a AS Integer
  DIM b AS Interger

  a = 2
  b = 0

PROVA LA
  STAMPA a / b
  SE ERRORE
  ALLORA
    Message.Error ("CLASS:" & Str (Error.class) & ", CODE:" & error.code &
    ", =" & error.text & "AT:" & error.where, "OK")
  ENDIF
END
```

Salva il tuo lavoro ed esegui il programma. Fai clic sul pulsante Esci e dovresti vedere qualcosa di simile a questo:



Una guida per principianti a

Figura 86 Risultati di errore rilevati con TRY.

Una guida per principianti a

Dichiarazioni di cattura e infine

Catturare è una trappola di errore generica. Viene eseguito quando viene generato un errore in un punto qualsiasi tra il punto in cui inizia l'esecuzione e il punto in cui finisce. Catch fornisce una mens per affrontare gli errori che possono verificarsi in una data funzione e consentire comunque all'utente di continuare a utilizzare il programma. Un errore può essere generato dalla funzione stessa o da qualsiasi altra funzione che può chiamare durante l'esecuzione. Se la funzione chiamata ha un blocco Catch, più il blocco Catch è annidato, maggiore è la priorità. In altre parole, se si è verificato un errore nella sezione di codice annidata, verrà eseguito il Catch in quella sezione, non il Catch dalla funzione chiamante. Se viene generato un errore mentre il programma è in esecuzione all'interno del blocco catch stesso, l'errore verrà normalmente propagato al di fuori del blocco catch perché catch non ha mezzi per proteggersi. Se c'è una parte Finalmente nella funzione, deve precedere la parte catch. Creiamo un semplice programma per dimostrare questa caratteristica. Crea un nuovo progetto denominato Chap13 come progetto basato su GUI e quando viene visualizzato l'IDE, crea un modulo di classe di avvio denominato Form1. Aggiungi due pulsanti, Error e Quit, denominati ErrBtn e QuitBtn. Posiziona i pulsanti nell'angolo inferiore destro del modulo. Crea un TLabel e chiamalo TLabel1. Mettilo nell'angolo in alto a sinistra del modulo e allungalo completamente e fallo diventare alto circa un pollice. Fai doppio clic sul modulo e inserisci questo codice: denominato ErrBtn e QuitBtn. Posiziona i pulsanti nell'angolo inferiore destro del modulo. Crea un TLabel e chiamalo TLabel1. Mettilo nell'angolo in alto a sinistra del modulo e allungalo completamente e fallo diventare alto circa un pollice. Fai doppio clic sul modulo e inserisci questo codice: denominato ErrBtn e QuitBtn. Posiziona i pulsanti nell'angolo inferiore destro del modulo. Crea un TLabel e chiamalo TLabel1. Mettilo nell'angolo in alto a sinistra del modulo e allungalo completamente e fallo diventare alto circa un pollice. Fai doppio clic sul modulo e inserisci questo codice:

```
"File di classe

Gambas PUBLIC SUB

    _new ()
    TLabel1.Text = "<center> <strong> Nessun errore
    presente." FINE

    FUNZIONE PUBBLICA STATICA Run () AS Boolean
        DIM hForm AS Form
        hForm = NEW Form1
        RETURN hForm.ShowModal
    () END
```

Una guida per principianti a

```
PUBLIC SUB QuitBtn_Click
    () ME.Close (TRUE)
FINE

PUBLIC SUB ErrBtn_Click
    () DIM a AS Integer
    DIM b AS Interger

    a = 2
    b = 0
    STAMPA a / b 'genererà una divisione per zero errori
```

Una guida per principianti a

```
INFINE
    TextLabel1.Text = "CLASS:" & Str (Error.class) & ", CODE:" &
error.code & ", =" & error.text & "AT:" & error.where
    TextLabel1.Refresh

CATTURARE
    SE ERRORE ALLORA
        Message.Error ("CLASS:" & Str (Error.class) & ", CODE:" &
error.code & ", =" & error.text & "AT:" & error.where, "OK")
    FINISCI SE
    Message.Info ("Cleared the Error", "OK")
    TextLabel1.Text = "<center> <strong> L'errore è stato
cancellato." FINE
```

Quando esegui il programma, dovresti vederlo come la schermata di



Figura 87 Schermata di apertura del programma di test CATCH.

Ora, fai clic sul pulsante di errore per generare il nostro errore e dovresti

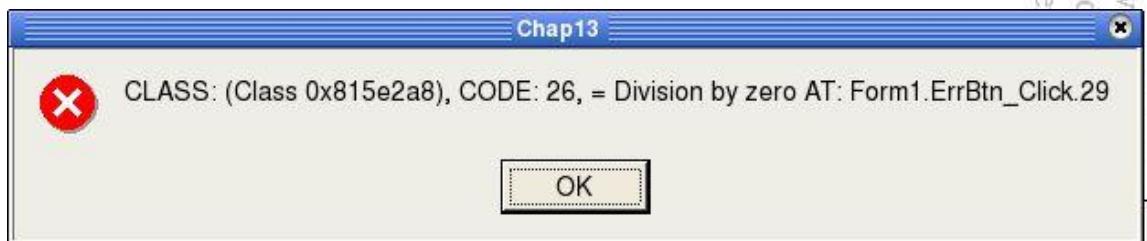


Figura 88 Errore Div per Zero rilevato da

Dovresti anche notare che la schermata principale TextLabel è cambiata in questo:

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved.
Gambas User Community under an Open Content License (OCL) and
under any other terms or conditions without the express written conser-

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a



Figura 89 L'etichetta di testo aggiornata con informazioni sull'errore.

Dopo aver fatto clic sul pulsante Ok per la finestra di dialogo Messaggio di errore, vedrai questo:

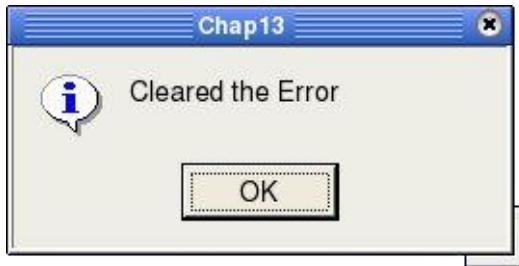


Figura 90 Messaggio informativo che indica che l'errore è stato cancellato.

Ora, fai clic sul pulsante OK sul messaggio Info e TextArea viene aggiornato in questo modo:



Figura 91 Schermata principale dopo la cancellazione dell'errore.

Questo dovrebbe darti un'idea abbastanza chiara di come utilizzare le funzionalità di gestione degli errori di Gambas per aiutare a gestire gli errori quando si verificano nel tuo codice (e si verificheranno inevitabilmente). Non gestire errori come il div per zero che abbiamo usato nel nostro esempio sarà un "ostacolo" nella

Una guida per principianti a maggior parte delle circostanze. Come rapido esempio,

Una guida per principianti a

torna al nostro codice, nella subroutine ErrBtn_Click e commenta queste righe:

```
' INFINE
"TextLabel1.Text =" CLASS: "& Str (Error.class) &", CODE: "&
error.code &", = "& error.text &" AT: "& error.where
TextLabel1.Refresh
'CATTURARE
'SE ERRORE ALLORA
'Message.Error ("CLASS:      "& Str (Error.class) &", CODE: "&
error.code &", = "& error.text &" AT: "& error.where," OK ")
'ENDIF
'Message.Info ("Cleared the Error", "OK")
```

Ora, esegui di nuovo il programma e fai clic sul pulsante Errore. Ecco cosa ottieni:

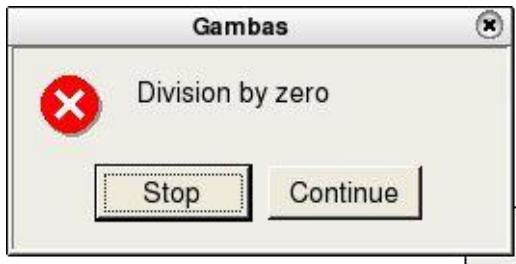


Figura 92 Finestra di dialogo di errore predefinita di Gambas.

Puoi provare a continuare, ma non c'è alcuna garanzia che il tuo programma funzioni correttamente o che continuerà. Questo non è sicuramente quello che vorresti che i tuoi utenti vedessero. È semplicemente logico utilizzare gli strumenti disponibili in Gambas per evitare che ciò accada quando è possibile. Ora, passiamo a vedere come possiamo usare gli eventi per aiutare a scrivere un codice migliore.

Gestione eventi Gambas

Gambas si basa su un paradigma di programmazione basato sugli eventi. I linguaggi di programmazione per computer tradizionali sono stati progettati per seguire il proprio flusso di controllo che cambia la direzione del flusso (corso) nei punti di diramazione. In un paradigma basato sugli eventi, il flusso di controllo è in gran parte guidato da eventi esterni come i movimenti del mouse, i clic dei pulsanti, ecc. qualsiasi tipo di evento, anche il passare del tempo). Se viene rilevato un evento, attiva un flag che può essere verificato nel loop degli eventi. Ciò consente ai programmati di verificare e rispondere agli eventi scrivendo subroutine o funzioni per elaborarli.

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is licensed under the GNU General Public License Version 2, or any later version, which can be found in the file COPYING. It may be redistributed and modified under the express written consent of the author.

Una guida per principianti a

Il modo in cui le informazioni sugli eventi vengono acquisite dal sistema non è molto importante per il programmatore. È possibile eseguire il polling dei tipi comuni di dati di input degli eventi nel loop degli eventi. Eventi complessi o proprietari possono richiedere l'uso di gestori di interrupt che reagiscono a eventi hardware, come i dati del segnale che attraversano una scheda di interfaccia di rete (NIC). La maggior parte dei sistemi basati sugli eventi utilizza entrambe le tecniche. Gli algoritmi scritti dal programmatore per rispondere agli eventi assicurano che quando un determinato evento viene attivato, viene fornito il codice per gestire l'evento in un modo accettabile per l'utente. Questo crea uno strato di astrazione software che emula un ambiente basato su interruzioni. I programmi basati su eventi sono generalmente costituiti da una serie di piccoli programmi chiamati gestori di eventi. I gestori di eventi vengono chiamati in risposta agli eventi. Un dispatcher viene utilizzato per chiamare il gestore eventi. Più spesso, il dispatcher viene implementato utilizzando una coda di eventi per contenere eventi non elaborati. A volte, i gestori di eventi possono attivare gli eventi stessi. Questo può eventualmente portare a una cascata di eventi (non va bene).

Gambas, come i più comuni programmi di interfaccia utente grafica, è programmato in uno stile basato sugli eventi. In Gambas, hai la possibilità di creare eventi da utilizzare nei tuoi programmi. Puoi dichiarare un evento come questo:

```
Identificatore EVENTO ([Parametro # 1 [, Parametro # 2 ...]]) [AS  
Boolean]
```

Questa dichiarazione dichiarerà un evento di classe. Un evento di classe viene generato da una chiamata di funzione. Inoltre, è possibile specificare se il gestore eventi restituirà o meno un valore (booleano). Se viene specificato un valore di ritorno, un risultato di ritorno TRUE indica che l'evento è stato annullato.

Una caratteristica unica di Gambas è che un oggetto può generare un evento. Per generare un evento, all'oggetto deve essere assegnato un nome (o handle) nel momento in cui viene creato con la parola chiave NEW. Senza l'assegnazione di un nome, gli eventi generati da un oggetto vengono ignorati. Gli eventi vengono rilevati dal genitore di un oggetto. Il genitore è in realtà l'osservatore di eventi unico di un oggetto. Il genitore funge da osservatore predefinito per l'oggetto quando viene generato un evento. Questo genitore può essere una classe, nel qual caso il gestore di eventi deve essere un metodo statico. In questo caso, l'handle e il genitore possono essere modificati entrambi in fase di esecuzione utilizzando i metodi Object.Attach () e Object.Detach () della classe Object. L'handle dell'oggetto non è memorizzato con un oggetto e pertanto non può essere recuperato.

Una guida per principianti a

In Gambas, la convenzione di chiamata standard per un gestore di eventi sarebbe
InstanceName_EventName per invocare il metodo che deve essere chiamato dal
listener di eventi

Una guida per principianti a

quando viene generato un tale evento. Aggiungiamo al nostro esempio precedente per dimostrare l'uso degli eventi. Aggiungi un altro pulsante al modulo per il nostro esempio precedente, denominato EventBtn. Fare doppio clic e aggiungere questo codice al programma:

```
EVENT MyEvent (Msg AS String) AS  
  
Boolean PUBLIC SUB Event1Btn_Click ()  
DIM bResult AS Boolean  
  
RAISE MyEvent ("Genera un evento") AS  
  
bResult IF bResult THEN  
    TextLabel1.Text = "Annullato l'evento"  
ELSE  
    TextLabel1.Text = "Ha generato  
l'evento" bResult = NON bResult  
    STOP  
EVENTO ENDIF  
FINE
```

Salva il tuo lavoro ed esegui il programma. Fai clic sul pulsante Evento e il tuo display dovrebbe apparire così:

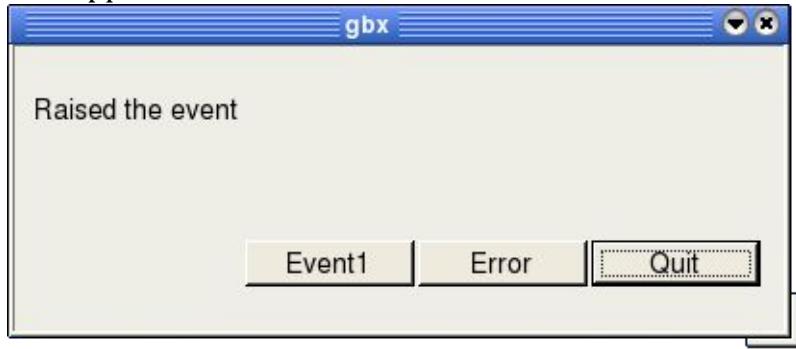


Figura 93 Il nostro evento è stato generato.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the public under the terms of the Open Content License (OCL) and may not be distributed or copied without the express written consent of the author.

Capitolo 14 - Mouse, tastiera e operazioni sui bit

Il mouse è stato progettato per sostituire la raccolta di input da tastiera da parte di un utente. Mentre la progettazione della GUI è progredita in modo significativo sin dai primi giorni delle interfacce a finestra, l'evoluzione del mouse non lo è stata. Probabilmente il cambiamento più grande per il mouse è stato il passaggio ottico e cordless. Tuttavia, la funzionalità di base rimane praticamente la stessa. Fornire una posizione X, Y, fare clic su un pulsante sinistro, destro o centrale e rilevare il suo stato come su o giù e controllare il movimento della rotellina del mouse o modificare il valore delta quando la rotellina si muove. Gambas, ovviamente, supporta tutte queste funzionalità di base con un approccio logico e facile da codificare. Nelle prossime pagine, ti mostreremo come padroneggiare l'input da mouse e tastiera. Infine, come ultimo esercizio, creeremo un programma per mostrarvi come utilizzare gli operatori di bit. Cominciamo con il mouse.

Operazioni con il mouse

In Gambas, possiamo usare la classe Mouse per ottenere informazioni su un evento del mouse. Questa classe definisce le costanti utilizzate con le proprietà del mouse. La classe è statica e le proprietà supportate sono:

| | | | | |
|----------------|-----------------|------------------|---------------------|-----------------|
| Alt | Pulsante | Controllo | Delta | Sinistra |
| Meta | Medio | Normale | Orientamento | Destra |
| ScreenX | ScreenY | Cambio | X | Centro |

La maggior parte di queste proprietà sono autoesplicate dal loro nome, ma alcune necessitano di ulteriori spiegazioni. Le proprietà Alt, Control, Meta e Shift restituiscono tutte un valore TRUE se vengono premuti i rispettivi tasti. Normale può essere controllato (testato) per vedere se è stato premuto un altro tasto speciale. Il pulsante può essere utilizzato per vedere se viene premuto un pulsante, ma è possibile utilizzare Sinistra, Destra e Centro per determinare se è stato premuto un pulsante del mouse specifico. Tutte queste proprietà restituiscono TRUE se viene premuto un pulsante. Delta restituisce il valore (offset) della rotellina del mouse mentre Orientation può essere utilizzato per controllare la direzione (avanti, indietro) della rotazione della rotellina. ScreenX e ScreenY restituiscono il valore assoluto del cursore in base al desktop mentre X e Y restituiscono la posizione relativa alla finestra.

This product is covered by the Open Content License (OCL) and may be distributed without the express written consent of the author. This document is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The author shall not be liable for any damages resulting from the use of this document.

Una guida per principianti a

Esiste un solo metodo per la classe Mouse, Move. Il metodo Move riposiziona semplicemente il cursore del mouse sulle coordinate X, Y specificate. Gli eventi del mouse che puoi monitorare nel tuo programma sono Mousemove, MouseDown, Mouseup,

Una guida per principianti a

e la rotellina del mouse. Qualsiasi modifica nello stato corrente di tali eventi consente di creare una risposta codificata per quell'evento. Quando si codificano le operazioni del mouse (o della tastiera), è molto importante utilizzare sempre le costanti predefinite. L'uso di qualsiasi altro metodo per ottenere o impostare le proprietà non è garantito per essere compatibile con le versioni successive da una versione di Gambas all'altra. La classe Mouse utilizza le seguenti costanti per specificare la forma del cursore del mouse:

| | | | | |
|---------------------|------------------|--------------------|---------------------|--------------------|
| Freccia | Vuoto | Attravers | Personalizza | Predefin |
| Orizzontal e | Indicando | are | to | ito |
| | | SizeAll | Taglia E. | Taglia H. |
| TagliaN | TagliaNE | TagliaNES W | TagliaNW | TagliaNW SE |
| TagliaS | TagliaSE | Taglia SW | TagliaV | TagliaW |
| SplitH | SplitV | Testo | Verticale | Aspettare |

Creiamo un breve programma Gambas per dimostrare come utilizzare il mouse ora. Dall'IDE Gambas, crea un nuovo progetto chiamato MouseOps e rendilo un progetto di interfaccia utente grafica, traducibile con controlli pubblici. Crea un nuovo modulo, Form1 e rendilo un modulo di classe di avvio. Successivamente, aggiungi un'area di disegno denominata da nell'angolo in alto a sinistra della finestra di Form1. Ridimensionalo per essere circa un pollice quadrato. Ora aggiungiamo un po 'di codice. Prima di tutto, quando il programma viene eseguito e la finestra si apre, vogliamo impostare una didascalia che dica all'utente come cancellare o uscire dalla finestra. Abbiamo anche bisogno di ridimensionare l'area di disegno per adattarla completamente al modulo e impostare la proprietà del buffer memorizzato nella cache su TRUE in modo che il nostro disegno possa essere visto dall'utente. Ecco come farlo:

```
"File di classe Gambas

'quando il modulo si apre in fase di esecuzione, imposta una didascalia e
ridimensiona l'area di disegno PUBLIC SUB Form_Open ()
    ME.Caption = "Premi la barra spaziatrice per cancellare,
    la rotellina del mouse esce ..." da.W = ME.W5
    da.H = ME.H5
    da.ForeColor = color.Black 'imposta il colore di primo piano su
    nero' imposta la proprietà memorizzata nella cache su true in
    modo da poter memorizzare il buffer di disegno da.Cached = TRUE
    draw.Begin (da) 'inizializza a disegnare
        'disegneremo una linea orizzontale e verticale nella sezione quattro
        quadranti draw.Line (1, da.h / 2, da.W1, da.h / 2)
        draw.Line (da.W / 2, 1, da.W / 2, da.H
    1) draw.End
FINE
```

Una guida per principianti a

Se il mouse si muove, vogliamo cambiare la forma del cursore, a seconda del quadrante dell'area di disegno in cui si trova il mouse. Useremo l'evento Mousemove e codificheremo questo:

Una guida per principianti a

```
'se il mouse si muove, aggiorneremo la forma del cursore a seconda del'  
quadrante in cui si trova il mouse  
PUBLIC SUB da_MouseMove ()  
    'controlla il quadrante in  
    alto a sinistra  
    IF mouse.X <= da.W / 2 AND mouse.Y <= da.H / 2  
        THEN da.Mouse = mouse.Pointing  
        altrimenti controlliamo il quadrante in  
        basso a sinistra ALTRIMENTI SE  
            mouse.X <= da.W / 2 AND mouse.Y >= da.H / 2  
            THEN da.Mouse = mouse.Cross  
            altrimenti controlliamo il quadrante  
            in alto a destra ALTRIMENTI SE  
            mouse.X >= da.W / 2 AND mouse.Y <= da.H / 2  
            THEN da.Mouse = mouse.SizeNWSE  
            'altrimenti, deve essere nel quadrante in basso a  
            destra ALTRIMENTI SE  
            mouse.X >= da.W / 2 AND mouse.Y >= da.H / 2  
            THEN da.Mouse = mouse.SizeNESW  
            'se è altrove, impostalo come predefinito  
ALTRO  
    da.Mouse = mouse.Default  
ENDIF
```

Ora che abbiamo impostato la forma del cursore, controlleremo se l'utente sta premendo un pulsante del mouse per disegnare. Se viene premuto il pulsante sinistro, disegneremo dei punti. Se è il pulsante giusto, disegneremo minuscole caselle.

```
'controllare il pulsante del mouse è premuto per vedere quale è  
premuto e' disegnare punti per btn sinistro, caselle per destra  
SE Mouse a  
    sinistra ALLORA  
        'inizializza  
        disegnare  
        Draw.Begin (da)  
            'piccoli puntini Draw.Point  
            (Mouse.X, Mouse.Y)  
        Disegna Fine  
ALTRO 'se è il pulsante  
    giusto Draw.Begin (da)  
        Draw.Rect (Mouse.X, Mouse.Y, 3,  
        3) Draw.End  
ENDIF  
END
```

Ogni volta che l'utente sposta la rotellina del mouse, termineremo l'applicazione.

```
'se l'utente muove la rotellina del mouse, termineremo l'applicazione
```

This product is (C) 2005 by John W. Rittinghouse. All rights are reserved. It is released to the
Gambas User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB da_MouseWheel ()  
    ME.Close  
END
```

Una guida per principianti a

Se l'utente ha premuto e rilasciato un pulsante del mouse, ripristineremo il cursore del mouse allo stato normale.

```
'ogni volta che si rilascia un pulsante del mouse, reimposta il  
cursore del mouse sulla normale PUBLIC SUB da_MouseUp ()  
da.Mouse = mouse.Default  
END
```

Infine, se l'utente preme un tasto qualsiasi, ripristineremo l'area di disegno svuotandola e ridisegneremo il quadrante richiamando la stessa routine usata per aprire il form:

```
'seleziona gli eventi kbd per cancellare l'area di disegno e  
ridisegnare i quadranti PUBLIC SUB Form_KeyPress ()  
da.Clear  
Form_Open  
()  
FINE
```

Questo è tutto. Quando esegui il programma, dovresti vedere qualcosa del genere:

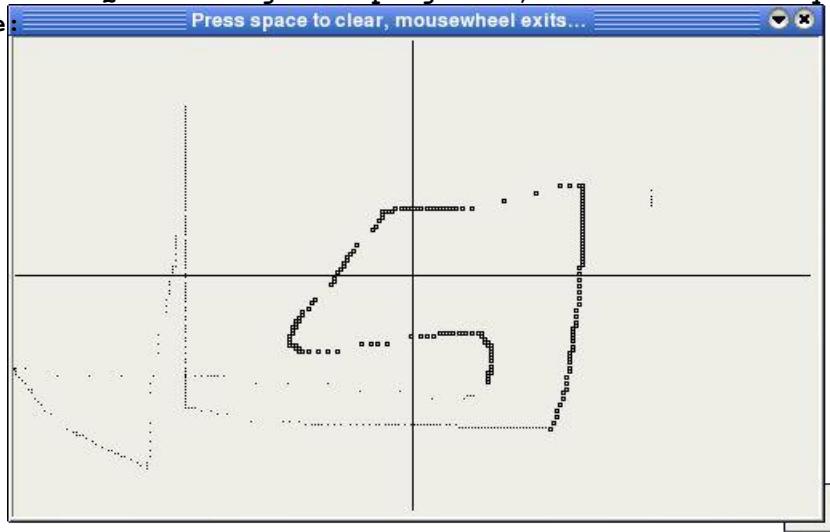


Figura 94 Programma MouseOps in esecuzione.

Mentre muovi il mouse da quadrante a quadrante, nota come cambia il cursore. Prova entrambi i pulsanti sinistro e destro del mouse e premi la barra spaziatrice per cancellare il disegno e ricomincia. Quando sei soddisfatto che tutto funzioni, muovi la rotellina del mouse e il programma terminerà. Ora impariamo le operazioni da tastiera.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Operazioni da tastiera

Questa classe viene utilizzata per ottenere informazioni su un evento della tastiera. Contiene tutte le costanti predefinite che Gambas utilizza per rappresentare i tasti presenti sulla tastiera. Come abbiamo sottolineato in precedenza, non dovresti mai usare direttamente i valori della chiave intera. Dovresti sempre usare le costanti predefinite di Gambas. Questa classe è statica e si comporta come un array di sola lettura. È possibile dichiarare una variabile intera e ottenere il valore di una costante chiave utilizzando questa convenzione di chiamata:

```
DIM MyInteger AS Integer  
MyInteger = Key [Key AS String]
```

Le proprietà supportate per la classe Key includono Alt, Code, Control, Meta, Normal, Shift, State e Text. Lo stato può essere utilizzato per verificare se viene premuto o meno un tasto speciale e Text restituirà la rappresentazione di un singolo carattere per un dato tasto. Tutte le altre proprietà funzionano come quelle usate con la classe Mouse che abbiamo descritto nella sezione precedente. La classe Key supporta le seguenti costanti predefinite:

Per il cursore chiavi: **Su, Giù, Sinistra e Destra**

Per la funzione chiavi: **F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12**

Per Fkeys spostati: **F13 F14 F15 F16 F17 F18 F19 F20 F21 F22 F23 F24**

Per tutti gli altri chiavi: **BackSpace BackTabCapsLockDeleteEnd
EnterEscEscapeHelpHome**

**InsertMenuNumLockPageDown
Pagina su PausePrintReturn
Blocco scorrimento Spazio SysReqTab**

Creiamo un altro breve programma Gambas per dimostrare come utilizzare la tastiera. Dall'IDE Gambas, crea un nuovo progetto chiamato KbdOps e rendilo un progetto di interfaccia utente grafica, traducibile con controlli pubblici. Crea un nuovo modulo, Form1 e rendilo un modulo di classe di avvio. Avremo solo bisogno di usare due etichette in questo programma, una TLabel e un'etichetta. Per prima cosa, crea un TLabel denominato TLabel1 e fallo alto circa 1/2 pollice e allungalo su tutta la larghezza del modulo. Impostare la proprietà TLabel1.Alignment su center. Ora crea un'etichetta denominata Label1 e

The project (c) 2005 by John W. R. Houghhouse, all rights are reserved. It is released under the GNU General Public License (GPL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
posizionala sotto il controllo TextLabel1. Fatene circa due

Una guida per principianti a

pollici di larghezza e 1/2 "di altezza. Impostare la proprietà Label.Text su "Press any key ..." e impostare la dimensione del carattere di controllo su 14 punti e renderla in grassetto. Fare doppio clic sul modulo e viene visualizzata la finestra del codice. Ecco il codice che aggiungeremo al nostro programma:

```
"File di classe Gambas

' controlleremo l'evento di rilascio della chiave per
vedere cosa fare PUBLIC SUB Form_KeyRelease ()
    'ogni volta che viene rilasciato un tasto,
    controllare il codice SELEZIONA
    codice.chiave

    Tasto CASE Tab 'se è un tasto di tabulazione
        textlabel1.Text = "Il codice chiave è:" & key.Code & TAB."
    Tasto CASE BackSpace "o backspace
        textlabel1.Text = "Il codice chiave è:" & key.Code &
Backspace."
    Tasto CASE CapsLock 'o blocco maiuscole
        textlabel1.Text = "Il codice chiave è:" & key.Code &
CapsLock."
    Tasto CASE F1 'o un Fkey
        textlabel1.Text = "Chi codi è: " & chiave & ", F1 chiav
ave ce
    Tasto CASE F2
        textlabel1.Text = "Chi codi è: " & chiave & ", F2 chiav
ave ce
    Tasto CASE F3
        textlabel1.Text = "Chi codi è: " & chiave & ", F3 chiav
ave ce
    Tasto CASE F4
        textlabel1.Text = "Chi codi è: " & chiave & ", F4 chiav
ave ce
    Tasto CASE F5
        textlabel1.Text = "Chi codi è: " & chiave & ", F5 chiav
ave ce
    Tasto CASE F6
        textlabel1.Text = "Chi codi è: " & chiave & ", F6 chiav
ave ce
    Tasto CASE F7
        textlabel1.Text = "Chi codi è: " & chiave & ", F7 chiav
ave ce
    Tasto CASE F8
        textlabel1.Text = "Chi codi è: " & chiave & ", F8 chiav
ave ce
'se nessuna delle precedenti, è stampabile? In tal caso, mostra
codice e valoreCASO ALTRO
    Chiave IF Codice> 32 Chiave AND Codice <128 ALLORA
        textlabel1.Text = "il codice chiave è:" & key.Code & "," & Chr
(key.Code)
    ALTRO 'altrimenti solo il codice
        textlabel1.Text = "il codice chiave è:" &
```

Una guida per principianti a

```
key.Code ENDIF
FINE SELEZIONA
FINE

'seleziona l'evento key down per filtrare shift, control, alt,
ecc. PUBLIC SUB Form_KeyPress ()

Tasto IF Shift THEN
    textlabel1.Text = "Il codice chiave è:" & key.Code & ","
SHIFT" ELSE IF key. Alt THEN
```

Una guida per principianti a

```
textlabel1.Text = "Il codice chiave è:" & key.Code & ", ALT"  
Tasto ALTRIMENTI SE Controllare ALLORA  
    textlabel1.Text = "Il codice chiave è:" & key.Code & ", CTRL"  
ELSE SE chiave Codice> 32 Chiave AND Codice <128 ALLORA  
    textlabel1.Text = "il codice chiave è:" & key.Code & "," & Chr  
        (key.Code)  
ELSE IF key.Esc THEN 'se viene premuto un tasto ESC  
    textlabel1.Text = "il codice chiave è:" & key.Code & ",  
        ESC"  
ALTRO  
    textlabel1.Text = "il codice chiave è:" &  
key.Code ENDIF  
FINE  
  
PUBLIC SUB Form_Open ()  
    ME.Caption = "Operazioni da  
tastiera" END
```

Come puoi vedere, il codice sopra è piuttosto diretto e semplice.
Quando esegui il programma, dovrà apparire qualcosa del genere:

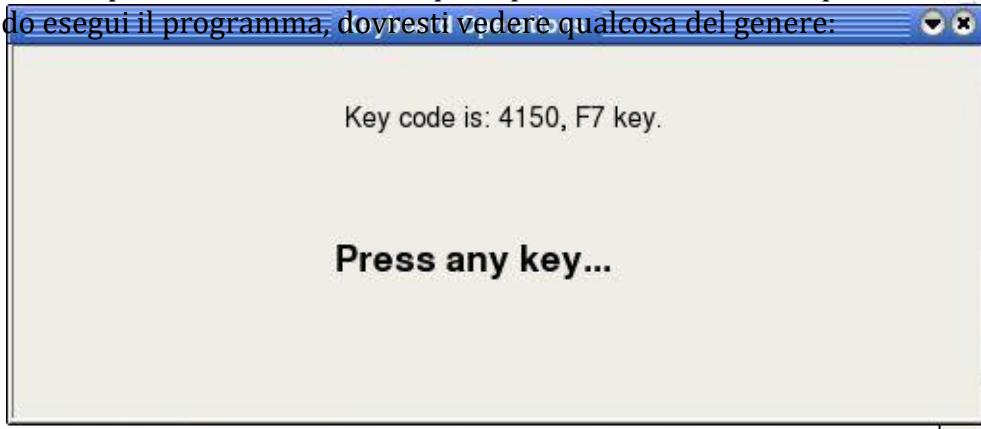


Figura 95 Programma KbdOps in esecuzione.

Operazioni sui bit

Bit sono gli elementi più piccoli utilizzati nelle operazioni del computer. Sono atomi indivisibili di memoria raggruppati in gruppi chiamati byte. Un byte è composto da n numero di bit, che dipende dalla dimensione della parola macchina. Ad esempio, in un sistema a otto bit, un byte è composto da otto bit. Per i moderni sistemi informatici utilizzati oggi, il sistema operativo è costruito utilizzando parole di 32 o 64 bit. La dimensione della parola aumenta per ospitare uno spazio di memoria indirizzabile più ampio. In un vecchio sistema a otto bit, ad esempio, il

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Microsoft Public License (MS-PL).
Gambas User Community under an Open Content License (OCL) and may not be distributed outside the community without the express written consent of the author.

Una guida per principianti a
sistema può indirizzare solo la memoria che può essere espressa in forma binaria
come 11111111. In

Una guida per principianti a

in altre parole, lo spazio di memoria è limitato a una dimensione che può essere espressa con quegli otto bit. Per un sistema a otto bit, gli intervalli di memoria si estendono dalla rappresentazione decimale a byte singolo di 65537 a +65538. In un moderno sistema a 32 bit, assume la forma binaria di 11111111111111111111111111111111 che può essere espresso in notazione decimale da 2.147.483.648 a +2.147.483.647. Come puoi vedere, la gamma di memoria indirizzabile con i sistemi a 32 bit è enorme!

Nella programmazione a livello di applicazioni, in genere non vi sono molte occasioni per operare a livello di bit quando ma occasionalmente se ne presenta la necessità. Gambas fornisce tutti gli strumenti necessari per testare un po', cancellarlo, modificarlo (invertirlo) o impostarlo. Inoltre, le operazioni bit per bit che puoi eseguire includono la rotazione leggermente a destra o sinistra e lo spostamento leggermente a destra o sinistra. La tabella seguente riassume le operazioni di bit che puoi eseguire in Gambas:

Gambas Bit Operations

| Operator e | Sintassi | Azione intrapresa |
|-------------------|------------------------------|--|
| BTst | Boolean = BTst (numero, bit) | Verificare se un bit è impostato in operando1 utilizzando operando2 |
| BClr | Valore = BClr (numero, bit) | Cancella un bit specifico in operando1 usando operando2 |
| BSet | Valore = BSet (numero, bit) | Restituisce TRUE se il bit specificato nell'operando2 dell'operando1 è impostato. |
| BChg | Valore = BChg (numero, bit) | Restituisce operando1 con il bit specificato in operando2 invertito. |
| Shl | Valore = Shl (numero, bit) | Restituisce operando1 con il bit specificato in operando2 spostato a sinistra di un bit posizione. |
| Shr | Valore = Shr (numero, bit) | Restituisce operando1 con il bit specificato in operando2 spostato a destra di un bit posizione. |
| Rol | Valore = Rol (numero, bit) | Restituisce le posizioni di bit dell'operando1 ruotato a sinistra dell'operando2. |
| Ror | Valore = Ror (Numero, Bit) | Restituisce le posizioni di bit dell'operando1 ruotato a destra dell'operando2. |

Ora creeremo un altro breve programma Gambas per dimostrare come utilizzare le funzioni bit. Dall'IDE Gambas, crea un nuovo progetto chiamato BitOps. Rendilo un progetto di interfaccia utente grafica, traducibile con controlli pubblici. Crea un nuovo modulo, Form1 e rendilo un modulo di classe di avvio. Il nostro programma prenderà un numero intero compreso tra 0 e 255 e visualizzerà i valori esadecimali e binari per quel numero. Ti consentirà di selezionare uno qualsiasi degli otto bit utilizzati per rappresentare il numero intero ed eseguire qualsiasi operazione rappresentata dal pulsante su cui fai clic. Il valore binario di quel numero intero verrà visualizzato in modo da poter confrontare quel valore binario con il valore dell'intero come originariamente specificato. Per questo programma, avremo bisogno di sette TLabel, denominate da TLabel1 a TLabel7. Avremo

Una guida per principianti a
bisogno di due TextBox, TextBox1 e TextBox2. Infine, aggiungeremo nove pulsanti
denominati RolBtn, RorBtn, ShlBtn, ShrBtn, ClearBtn, TestBtn, SetBtn, ChangeBtn e
QuitBtn. Disegna i controlli

Una guida per principianti a
sopra sul modulo, come mostrato nella figura seguente:

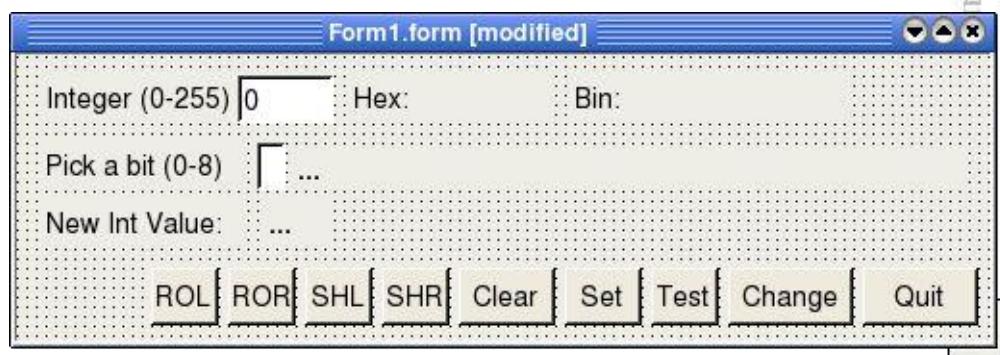


Figura 96 Il programma BitOps in modalità progettazione.

Una volta terminata la disposizione dei controlli come mostrato nella figura sopra, fare doppio clic da qualche parte sul modulo e verrà visualizzata la finestra del codice. Aggiungere il codice seguente per la subroutine Form_Open:

```
"File di classe Gambas
PUBLIC SUB Form_Open()
    ME.Caption = "bitOpns"
END
```

Ora, il programma visualizzerà la didascalia "bitOpns" quando viene eseguito. Codifichiamo una procedura di uscita in modo che l'utente possa terminare l'applicazione in seguito. Fai doppio clic sul pulsante Esci e aggiungi questo:

```
PUBLIC SUB QuitBtn_Click()
    ME.Close 'chiudi la finestra
FINE
```

Se l'utente immette un valore nel campo TextBox2 per modificare un po ', è necessario aggiornare il modulo e riflettere i valori modificati. Ci preoccuperemo di cosa fare con il nuovo valore un po 'più avanti nel codice. Per ora, è una semplice chiamata al metodo Refresh di cui abbiamo bisogno:

```
PUBLIC SUB TextBox2_Change ()
    form1.Refresh 'aggiorna la nostra visualizzazione ogni volta che
il valore del bit cambia FINE
```

Quando l'utente specifica un valore intero nel campo TextBox1, reagiremo solo alla voce quando viene premuto il tasto TAB. Ciò causa un evento LostFocus, che è ciò per cui codificheremo. Ecco cosa dobbiamo fare dopo:

This product is (C) 2005 by John Rittinghouse, all rights are reserved. It's released to the Gambas User Community under the Open Content License (OCL) and may be distributed under conditions without the express written consent of the author.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
PUBLIC SUB TextBox1_LostFocus ()  
    'abbiamo bisogno di un int var locale  
    con cui lavorare DIM myInt AS Integer  
    'converte il nostro valore di testo  
    della stringa in un int myInt = CInt  
    (textbox1.text)  
    'controlla per vedere se è nell'intervallo di 8 bit  
    IF Int (myInt) >= 0 AND (myInt) < 256 THEN  
        Label2.Text = "Hex:" & Hex (myInt) 'converti valore  
        esadecimale label3.Text = "Bin:" & Bin (myInt)  
        'converti valore bin label7.Text = Str $ (myInt)  
        'aggiorna i nostri campi di testo form1.Refresh  
        'aggiorna il modulo  
    ENDIF  
END
```

Quando l'utente preme il pulsante cancella, vogliamo cancellare il bit specificato per il numero intero rappresentato dal primo parametro.

```
PUBLIC SUB ClearBtn_Click  
    () DIM i AS Integer  
    Risultato DIM AS  
    Integer DIM myInt AS  
    Integer  
    'converte il nostro valore di testo  
    della stringa in un int myInt = CInt  
    (textbox1.text)  
    'se il numero intero è compreso tra 0 e 255 process, altrimenti  
    ignora IF Int (myInt) >= 0 AND (myInt) < 256 THEN  
        'converte il valore della  
        stringa in un int risultato =  
        CInt (TextBox2.Text) 'chiaro il  
        bit  
        i = BClr (myInt, risultato)  
        'aggiorna le visualizzazioni  
        dell'etichetta  
        label5.Text = "Bclr:" & Bin (i)  
        label7.Text = Str $ (i)  
    ENDIF  
END
```

Quando l'utente preme il pulsante Set, vogliamo impostare il bit specificato per l'intero rappresentato dal primo parametro. Questo è essenzialmente l'opposto di Clear discusso sopra.

```
PUBLIC SUB SetBtn_Click  
    () DIM i AS Integer  
    Risultato DIM AS  
    Integer DIM myInt AS
```

Una guida per principianti a

Integer

```
'converte il nostro valore di testo  
della stringa in un int myInt = CInt  
(textbox1.text)  
'se il numero intero è compreso tra 0 e 255 process, altrimenti  
ignora IF Int (myInt> = 0) AND (myInt <256) THEN  
'converte il valore della stringa in un int
```

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the Gamasas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
risultato = CInt  
(TextBox2.Text) 'imposta il  
bit  
i = BSet (myInt,  
risultato) 'aggiorna le  
etichette  
label5.Text = "Bset:" & Bin (i)  
label7.Text = Str $ (i)  
ENDIF  
END
```

La funzione Bchg è in realtà solo un inverso operazione. Quando utente fa clic sul pulsante Cambia, eseguiremo questo codice:

```
PUBLIC SUB ChangeBtn_Click  
() DIM i AS Integer  
Risultato DIM AS  
Integer DIM myInt AS  
Integer  
  
myInt = CInt (textbox1.text)  
IF Int (myInt) >= 0 AND (myInt < 256) THEN  
    risultato = CInt (TextBox2.Text)  
    i = BChg (myInt, risultato)  
    label5.Text = "Bchg:" & Bin (i)  
    label7.Text = Str $ (i)  
ENDIF  
END
```

La funzione Shl sposterà il valore di MyInt n bit a sinistra dove n è specificato dal secondo parametro. Quando il nostro utente fa clic sul pulsante Shl, eseguiremo questo codice:

```
PUBLIC SUB SHLBtn_Click  
() DIM i AS Integer  
Risultato DIM AS  
Integer DIM myInt AS  
Integer  
  
myInt = CInt (textbox1.text)  
IF Int (myInt) >= 0 AND (myInt < 256) THEN  
    risultato = CInt (TextBox2.Text)  
    i = Shl (myInt, result)  
    label5.Text = "SHL:" & Bin (i)  
    label7.Text = Str $ (i)  
ENDIF  
END
```

La funzione Shr sposterà il valore di MyInt n bit a destra dove n è specificato

Una guida per principianti a
dal secondo parametro. Quando il nostro utente fa clic sul pulsante Shr, verrà
eseguita questa subroutine:

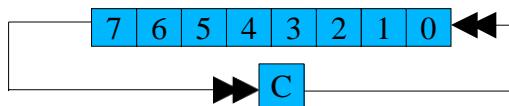
Una guida per principianti a

```
PUBLIC SUB SHRBtn_Click
() DIM i AS Integer
Risultato DIM AS
Integer DIM myInt AS
Integer

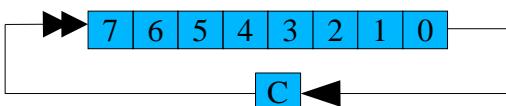
myInt = CInt (textbox1.text)
IF Int (myInt) >= 0 AND (myInt < 256) THEN
    risultato = CInt (TextBox2.Text)
    i = Shr (myInt, result)
    label5.Text = "SHR:" & Bin (i)
    label7.Text = Str $ (i)
ENDIF
END
```

Nei vecchi sistemi a 8 bit, era impossibile elaborare numeri maggiori di 255 a meno che non fossero suddivisi in blocchi da 8 bit. Le operazioni sono state quindi eseguite su ogni parte di un numero. Dopo che le operazioni sono state completate, è stato necessario riassemblare ogni numero per ricreare il numero intero. Tutto ciò è stato ottenuto utilizzando un piccolo bit, il carry bit. Le funzioni Gambas ROL (ruota a sinistra) e ROR (ruota a destra) vengono utilizzate per spostare i bit nei numeri binari. Queste funzioni funzionano concettualmente in questo modo:

L'istruzione ROL



L'istruzione ROR



ROL e ROR funzionano in modo tale che il bit di riporto venga spostato nel bit finale lasciato vuoto dal processo di rotazione invece di inserire un valore zero. ROL sposta il contenuto di un byte di una posizione a sinistra. ROL non pone uno zero nel bit 0. Invece, sposta il bit di riporto dalla posizione 7 del bit alla posizione 0 del numero che viene spostato. La posizione del bit 0 è stata liberata dalla rotazione di scorrimento e l'istruzione ROL pone il bit 7 nella posizione del bit di riporto. ROR funziona proprio come ROL, ma nella direzione opposta. Sposta ogni bit di un byte a destra di una posizione, posizionando il bit di riporto nel bit 7 e il bit 0 nel bit di

Una guida per principianti a
riporto.

Una guida per principianti a

```
'gira a sinistra
PUBLIC SUB ROLBtn_Click
    () DIM i AS Integer
    Risultato DIM AS
    Integer DIM myInt AS
    Integer

    myInt = CInt (textbox1.text)
    IF Int (myInt) >= 0 AND (myInt < 256) THEN
        risultato = CInt (TextBox2.Text)
        i = Rol (myInt, result)
        label5.Text = "ROL:" & Bin (i)
        label7.Text = Str $ (i)
    ENDIF
END

'ruota a destra
PUBLIC SUB RORBn_Click
    () DIM i AS Integer
    Risultato DIM AS
    Integer DIM myInt AS
    Integer

    myInt = CInt (textbox1.text)
    IF Int (myInt) >= 0 AND (myInt < 256) THEN
        risultato = CInt (TextBox2.Text)
        i = Ror (myInt, result)
        label5.Text = "ROR:" & Bin (i)
        label7.Text = Str $ (i)
    ENDIF
END
```

Se vuoi che il tuo programma controlli se è stato impostato un bit, puoi chiamare Btst. Questa funzione restituirà TRUE se il bit specificato nell'operando2 dell'operando1 è TRUE.

```
PUBLIC SUB TestBtn_Click
    () DIM i AS Boolean
    Risultato DIM AS
    Integer DIM myInt AS
    Integer

    myInt = CInt (textbox1.text)
    IF Int (myInt) >= 0 AND (myInt < 256) THEN
        risultato = CInt (TextBox2.Text)
        i = BTst (myInt, result)
        label5.Text = "Btst:" & i
    ENDIF
END
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Questo è tutto ciò che dobbiamo fare per i nostri piccoli scherzi programma il tuo lavoro e

Una guida per principianti a eseguire il programma. I tuoi risultati dovrebbero essere simili a questo:

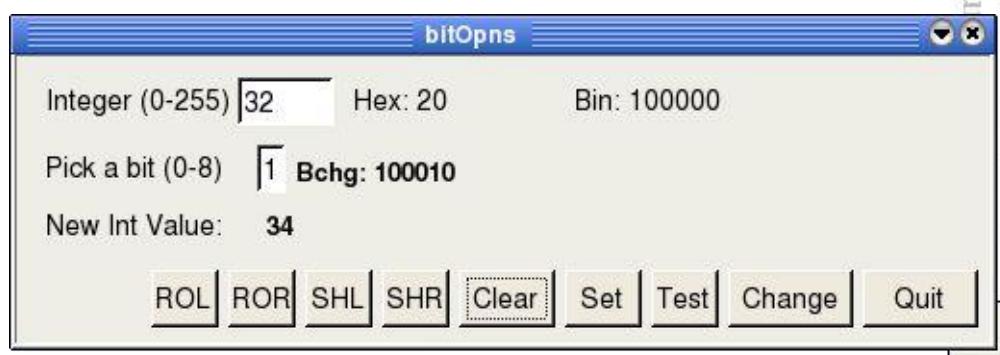


Figura 97 Il nostro programma bitOps in esecuzione.

Quando sei soddisfatto che il programma funzioni come previsto, salva il tuo lavoro. Sarebbe un buon esercizio in questo momento tornare indietro e modificare il codice per garantire che solo i valori interi vengano inseriti nei due campi della casella di testo. Inoltre, puoi consentire l'inserimento di numeri negativi per vedere quali risultati verrebbero prodotti. Lascio questo esercizio a coloro tra voi abbastanza desiderosi e ambiziosi da provare - non è davvero molto difficile e vi abbiamo già mostrato come eseguire le routine di convalida (ricordate, nel capitolo sulle raccolte?). Nel prossimo capitolo, tratteremo le operazioni di base del database con Gambas. La capacità di lavorare con i dati nei vostri programmi è essenziale e non possiamo trascurare questo aspetto vitale della programmazione nel nostro viaggio attraverso Gambas.

This product is (C) 2005 by John C. Chisholm. All rights are reserved. It is released to the Gambas User Community under the terms of the GNU General Public License (GPL) and may not be distributed under any other conditions without the express written consent of the author.

Capitolo 15 - Gambas e database

Questo capitolo descrive in dettaglio il processo di utilizzo di un database MySQL o PostgreSQL dall'interno dell'ambiente di programmazione Gambas. Per i nostri scopi, supporremo che MySQL sia già installato sulla tua macchina (o una macchina accessibile dalla tua macchina) e che sia disponibile un server MySQL a cui connetterti dall'interno del tuo programma Gambas. Se questo non è il caso, dovrà fare riferimento al manuale MySQL e capire come installare la distribuzione del database MySQL sul tuo particolare sistema. Dovresti anche essere consapevole del fatto che Gambas funzionerà anche con PostgreSQL e SQLite. SQLite non è una vera architettura client / server come MySQL e PostgreSQL. È un sistema di database flatfile e non utilizza il modello di database client / server. Per i nostri scopi in questo capitolo,

Il pacchetto database MySQL è costituito dal server MySQL e da molti programmi client. Il server MySQL è un programma che archivia e gestisce tutti i tuoi database all'interno dell'ambiente MySQL. I programmi client inclusi in MySQL sono troppo numerosi per essere elencati qui, ma è importante che tu comprenda che ogni applicazione client ha uno scopo specifico correlato alla gestione e / o amministrazione di dati o metadati memorizzati sul server MySQL. Il programma client di cui ci occuperemo in questo capitolo si chiama mysql (nota che è scritto in lettere minuscole). Questo programma client fornisce un'interfaccia a riga di comando (CLI) che è possibile utilizzare per inviare istruzioni SQL in modo interattivo al server MySQL e vedere i risultati visualizzati immediatamente.

A cosa serve avere sia un client che un server invece di un unico programma? I programmi server e client sono divisi in diverse entità in modo da poter utilizzare i programmi client sul sistema per accedere ai dati su un server MySQL che potrebbe essere in esecuzione su un altro computer. Separando il pacchetto in un server e un client, serve a separare i dati dall'interfaccia di recupero dei dati. Prima di iniziare a lavorare su questo capitolo, si consiglia vivamente di familiarizzare con MySQL scaricando l'ultimo manuale dal sito web ufficiale di MySQL¹⁸. Come minimo, completa il tutorial nel Capitolo 3 del Manuale utente MySQL prima di procedere con questo capitolo. Una volta che lo sei

18 Vedi URL <http://www.mysql.com/>.

This is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Una guida per principianti a

familiarità con il funzionamento delle operazioni SQL di base all'interno dell'ambiente MySQL, sei pronto per continuare con questo capitolo, che si concentra maggiormente su come realizzare gli stessi tipi di cose dimostrate dal tutorial MySQL, ma dall'interno di un programma Gambas.

Per accedere ai database all'interno dell'ambiente Gambas, è necessario assicurarsi che il progetto utilizzi il componente gb.db. Quando crei un nuovo progetto, devi andare alla finestra di dialogo delle proprietà del progetto e troverai una scheda per i componenti. Da questa finestra di dialogo a schede è possibile selezionare i componenti necessari per il programma. Nel caso dei database, sarà necessario controllare gb.db prima di poter utilizzare una qualsiasi delle classi definite da questo componente. Come abbiamo affermato in precedenza, il componente gb.db consente di accedere ai seguenti sistemi di gestione del database: PostgreSQL, MySQL e SQLite. Poiché PostgreSQL e MySQL sono database client / server, significa che il tuo programma dovrà stabilire una connessione a un server di database. SQLite è un database basato su file flat, quindi non esiste un processo server con cui il tuo programma può connettersi. Tuttavia,

Il componente gb.db è stato progettato per creare un livello intermedio (astratto) tra il server del database e il programma. Ciò ti consente di utilizzare lo stesso codice indipendentemente dal backend del database che decidi di utilizzare. Ci sono un paio di avvertimenti, ovviamente. Questo approccio astratto funziona solo se hai creato il tuo database utilizzando il database manager o utilizzando il componente gb.db. Inoltre, è necessario utilizzare i metodi Trova, Crea e Modifica forniti con il componente gb.db. Non è possibile inserire valori SQL direttamente in una richiesta: è necessario utilizzare la funzione di sostituzione dei metodi sopra menzionati per farlo. Infine, non è possibile utilizzare il metodo Exec per inviare richieste SQL direttamente al server per accedere a funzionalità specifiche del server. Anche se questo suona abbastanza restrittivo, in realtà non è un ostacolo al tuo programma. Alla luce di questi pochi avvertimenti, è possibile eseguire quasi tutte le operazioni di database necessarie dall'ambiente Gambas. Le classi supportate nel componente gb.db sono Connection, DB, Database, Field, Index, Result, ResultField, Table e User.

Classe di connessione

La classe Connection stabilisce la connessione al database. Per connettersi correttamente a un database, è necessario prima creare un oggetto di connessione e impostare le proprietà necessarie che verranno utilizzate dal database. Successivamente è necessario chiamare il metodo Open. Se ti stai connettendo a un database SQLite e il file Name

Una guida per principianti a

proprietà è null, un database viene aperto in memoria. Se il nome specificato è un percorso assoluto, viene utilizzato il percorso. Se il nome specificato è un percorso relativo e la proprietà Host è null, il database si trova nella directory temporanea dell'applicazione /tmp/gambas.\$UID/sqlite. In caso contrario, Host conterrà il nome della directory del database e il percorso del file del database è il risultato della concatenazione dei valori delle proprietà Host e Name. Questa classe è creabile e la convenzione di chiamata è:

```
DIM hConnection AS Connection  
hConnection = NEW Connection ()
```

Proprietà di connessione

Quando viene creata un'istanza dell'oggetto hConnection, viene creato un nuovo oggetto di connessione void. Sarà quindi necessario impostare le proprietà per questo oggetto. Le proprietà che è possibile utilizzare con un oggetto connessione includono set di caratteri, database, host, accesso, password, nome, porta, tabelle, tipo, utenti e versione. Spiegheremo ciascuno di questi nelle sezioni seguenti.

Charset

Charset è una proprietà di sola lettura che restituisce il set di caratteri utilizzato dal database. È possibile memorizzare le stringhe nel database direttamente in formato UTF8. Tuttavia, è anche possibile utilizzare questa proprietà in combinazione con la subroutine Conv \$ () per convertire dal formato UTF8 al formato del set di caratteri del database. Tuttavia, l'utilizzo del set di caratteri del database ha un costo in termini di prestazioni. Questo perché è necessario utilizzare la subroutine Conv \$ () ogni volta che si leggono o si scrivono dati stringa nei campi del database. Per ora, questo è l'unico modo per farlo se si desidera utilizzare le funzioni SQL basate su stringhe e le routine di ordinamento incorporate del sistema di gestione del database sottostante (MySQL, PostgreSQL o SQLite). Le versioni future di Gambas potrebbero eseguire automaticamente la conversione tra le stringhe di Gambas e il set di caratteri del database.

Banche dati

La proprietà Databases può essere controllata per ottenere un oggetto di raccolta che conterrà i nomi di tutti i database gestiti dal server database. La sintassi è:

```
PROPRIETÀ Database AS .ConnectionDatabases
```

Una guida per principianti a

La raccolta di dati risultante (che è enumerabile utilizzando FOR EACH

Una guida per principianti a

istruzione) sarebbe lo stesso come se avessi digitato SHOW DATABASES dalla CLI del client mysql.

Ospite

È possibile utilizzare la proprietà Host per impostare o ottenere il nome host della posizione in cui si trova il server database. Il nome host può essere un nome macchina o un indirizzo IP. L'host predefinito è denominato localhost. In SQLite, il nome host non è rilevante poiché SQLite non è un sistema di database client / server. Tuttavia, dalla versione 0.95 in avanti, è possibile impostare il percorso home per il database SQLite utilizzando questa proprietà. La sintassi per l'utilizzo della proprietà Host è:

PROPRIETÀ Host AS String

Login

La proprietà Login viene utilizzata per impostare o ottenere i dati di accesso dell'utente utilizzati per stabilire la connessione al database. Ancora una volta, questo vale per MySQL e PostgreSQL, non per SQLite. Poiché SQLite non ha il concetto di utenti, l'accesso a un database SQLite è controllato dalle impostazioni di autorizzazione del file del database stesso. Ciò significa che la proprietà Login sarà sempre impostata sull'id utente dell'utente che sta eseguendo il programma Gambas che utilizza il database SQLite. La sintassi è:

PROPRIETÀ Accesso come stringa

Nome

La proprietà Name imposta o ottiene il nome del database a cui si desidera connettersi. Se non si specifica un nome, viene utilizzato un database di sistema predefinito. Per PostgreSQL il database predefinito è denominato template1 e per MySQL è denominato mysql. Per SQLite, il nome predefinito è /tmp/sqlite.db. Per qualsiasi accesso al database, tuttavia, l'utente deve disporre almeno dei privilegi di accesso in lettura e connessione per utilizzare questi database. SQLite individuerà prima il database cercando di utilizzare il percorso completo se è stato fornito. In caso contrario, verificherà se la variabile Host è stata impostata. In tal caso, SQLite esaminerà il nome del percorso impostato nella variabile. Se è stata impostata la variabile di ambiente GAMBAS_SQLITE_DBHOME, SQLite la utilizzerà come directory di lavoro corrente. È anche possibile creare e utilizzare un database in memoria impostando la proprietà Name su ":memory:" anziché su "mysql" o "postgresql"

This product is covered by the terms of the GNU General Public License (GPL). All rights are reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of the author. This document is provided "as is" without warranty of any kind, either expressed or implied. The author makes no representations about the accuracy, reliability, or completeness of this document. The author shall not be liable for any damages arising from the use of this document. The author reserves the right to change the contents of this document at any time without notice.

Una guida per principianti a
o "sqlite". La sintassi è:

Una guida per principianti a

PROPRIETÀ Nome AS stringa

Parola d'ordine

La proprietà Password restituirà o imposterà la password utilizzata per stabilire una connessione al database. La convenzione di chiamata è:

PROPRIETÀ Password AS String

Porta

La proprietà Port viene utilizzata per ottenere o impostare la porta TCP / IP utilizzata per stabilire la connessione a un database. La porta predefinita dipende dal tipo di connessione. La sintassi è:

PROPRIETÀ Porta AS String

Tabelle

La proprietà Tables viene utilizzata per ottenere una raccolta virtuale che può essere utilizzata per la gestione di tutte le tabelle di un database. La sintassi è:

PROPRIETÀ Tabelle AS .ConnectionTables

genere

La proprietà Type rappresenta il tipo di server database a cui ci si desidera connettere. In Gambas, i tipi di database attualmente supportati sono: "postgresql", "mysql" e "sqlite". La proprietà type utilizza questa sintassi:

PROPRIETÀ Tipo AS String

Utenti

La proprietà Users restituisce una raccolta di tutti gli utenti registrati nel server di database. Come qualsiasi altro oggetto di raccolta, può essere enumerato con FOR EACH. La sintassi è:

PROPRIETÀ Users AS .ConnectionUsers

Versione

La proprietà Version è un valore di sola lettura che restituisce la versione di

Una guida per principianti a

database a cui il conducente si è connesso. La sintassi è:

```
PROPRIETÀ READ Versione AS Integer
```

Il concetto di transazione

Ci sono momenti in cui l'ordine in cui vengono eseguite le azioni (o le query) del database è importante. A volte, un programma deve garantire che tutte le query in un gruppo vengano eseguite correttamente e in ordine o elaborate nessuna di esse. Un classico esempio è tratto dall'ambiente bancario. Se una determinata somma di denaro (ad esempio, 100 dollari) viene prelevata da un conto e registrata su un altro conto, ci aspetteremmo che si verifichino le seguenti azioni:

```
AGGIORNAMENTO conto1 SET saldo = saldo - 100;  
AGGIORNA conto2 SET saldo = saldo + 100;
```

Entrambe le query devono essere eseguite correttamente o nessuna delle due deve essere eseguita. Il denaro non può essere trasferito da un account e non può essere registrato sull'altro account. Entrambe queste query formano un'unica transazione composta da due azioni separate (debitare 100 dal conto1 e accreditare 100 sul conto2). Una transazione è semplicemente una o più query di database individuali raggruppate tra le istruzioni BEGIN e COMMIT. Senza un'istruzione COMMIT, la transazione non è permanente e può essere annullata con l'istruzione ROLLBACK.

MySQL esegue automaticamente il commit delle dichiarazioni che non fanno parte di un processo di transazione. I risultati di qualsiasi istruzione SQL UPDATE o INSERT che non è preceduta da BEGIN saranno immediatamente visibili a tutte le connessioni al database perché il commit viene eseguito automaticamente. All'interno di MySQL, le transazioni sono note solo con tabelle "transazionali sicure" (cioè BDB e InnoDB). Gli altri tipi di tabella COMMIT immediatamente. Si dice che la maggior parte dei database in grado di eseguire questa operazione sia conforme ad ACID. Il modello ACID è uno dei concetti più antichi e importanti della teoria dei database. I quattro obiettivi che devono essere raggiunti affinché un database sia considerato affidabile sono Atomicity, Consistency, Isolation and Durability (ACID). Esaminiamo ciascuna di queste quattro caratteristiche in dettaglio.

Atomicità significa che le modifiche al database devono seguire una regola "tutto o niente". Ogni transazione è considerata un'unità "atomica". Se una qualsiasi parte della transazione fallisce, l'intera transazione fallisce. È fondamentale che il DBMS mantenga la natura atomica delle transazioni in qualsiasi circostanza.

Una guida per principianti a

Consistenza afferma che solo i dati validi verranno scritti in un database. Se viene eseguita una transazione che viola le regole di coerenza del database, verrà eseguito il rollback dell'intera transazione e il database verrà ripristinato a uno stato coerente con tali regole. Al contrario, se una transazione viene eseguita correttamente, manterrà uno stato di coerenza.

Solitudine richiede più transazioni che avvengono simultaneamente non si influenzano a vicenda. Ad esempio, se l'utente A emette una transazione su un database nello stesso momento in cui l'utente B emette una transazione, entrambe le transazioni dovrebbero operare sul database in una beatitudine isolata. Il database dovrebbe eseguire una transazione prima dell'altra. Ciò impedisce a una transazione di leggere i dati intermedi prodotti come effetto collaterale di un'altra transazione. L'isolamento non garantisce che nessuna delle due transazioni venga eseguita per prima, ma solo che non interferiranno tra loro.

Durevolezza garantisce che qualsiasi transazione salvata nel database non andrà persa. Ciò viene eseguito tramite backup del database e registri delle transazioni che facilitano il ripristino delle transazioni salvate nonostante eventuali errori successivi. Ora, diamo un'occhiata ai metodi di classe Gambas Connection disponibili nei tuoi programmi.

Metodi di classe di connessione

La classe Connection fornisce la maggior parte dei metodi necessari per stabilire una connessione a un database ed eseguire una o più transazioni per trovare, aggiungere, modificare o eliminare i dati.

Aperto chiuso

Il metodo Open viene utilizzato per aprire una connessione al database. Prima di effettuare la chiamata a Apri, è necessario impostare le proprietà di connessione con i dati necessari per utilizzare il database. In genere, è necessario specificare almeno il tipo di database, l'host, un ID di accesso e una password e il nome del database che si desidera utilizzare. La sintassi per open è:

```
FUNZIONE Open () AS Boolean
```

L'esempio di codice seguente mostra come utilizzare il metodo Open per connettere l'utente jittinghouse a un database MySQL denominato "GambasBugs" archiviato su Host localhost:

```
DIM $ hConn come NUOVA connessione
```

This product is (C) 2005 by John W. Rittinghouse, and
Gambas User Community under an Open Content license.
It is released under the terms of the GNU General Public License.
It is released without the express or implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. It is
not be distributed in modified form without the explicit
written consent of the author.

Una guida per principianti a

```
CON $ hConn
    .Type = "mysql"
    .Host = "localhost"
    .Login = "jrittinghouse"
    .Password = "ab32e44"
    .Name = "GambasBugs"
FINISCE CON

PROVA $ hConn.Open
IF Error THEN PRINT "Impossibile aprire il database. Error ="; Error.Text
```

Per chiudere la connessione, è sufficiente richiamare il metodo Close. Nel nostro esempio sopra, potresti usare questo codice:

```
$ hConn.Close
```

e la connessione verrebbe chiusa. Per poter utilizzare successivamente i dati memorizzati nel database, sarà necessario riaprire il database utilizzando il metodo Open.

Inizia / Conferma / Ripristina

Come affermato in precedenza, una transazione è una o più query di database individuali raggruppate tra le istruzioni BEGIN e COMMIT. Senza un'istruzione COMMIT, la transazione non è permanente e può essere annullata con un'istruzione ROLLBACK. Ricorda che MySQL esegue automaticamente il commit delle istruzioni che non fanno parte di un processo di transazione (definito da BEGIN e COMMIT). In Gambas, si consiglia di utilizzare i metodi Trova, Crea e Modifica per apportare modifiche al database. Ciò aiuta a mantenere l'indipendenza dal codice e consente di scrivere un singolo pezzo di codice che funzionerà con qualsiasi database supportato da Gambas. Gambas utilizza un oggetto risultato per restituire i risultati di una query SQL.

Trova

Il metodo Find restituisce un oggetto Result di sola lettura utilizzato per eseguire query sui record nella tabella specificata. La convenzione di chiamata per Trova è:

```
FUNZIONE Find (Table AS String [, Request AS String, Arguments AS, ...]) AS Risultato
```

Richiesta rappresenta un'istruzione SQL WHERE utilizzata per eseguire query sulla tabella. Gli argomenti vengono citati secondo necessità (detti dalla sintassi

Una guida per principianti a
SQL) e sostituiti all'interno della stringa di richiesta. Funziona come la funzione
Subst () in questo

Una guida per principianti a

Astuccio. L'utilizzo del metodo Find consente di scrivere richieste indipendenti dal database sottostante, il che significa che funzionerà con PostgreSQL o MySQL senza che sia richiesta alcuna modifica al codice.

Creare

Il metodo Create viene utilizzato per creare un oggetto Result di lettura / scrittura che può essere utilizzato per creare record in una tabella. La convenzione di chiamata standard è:

```
FUNZIONE Crea (tabella AS stringa) COME risultato
```

modificare

Il metodo Edit restituisce un oggetto Risultato di lettura / scrittura utilizzato per modificare i record nella tabella specificata. Request è una clausola SQL WHERE utilizzata per filtrare la tabella e gli argomenti sono citati secondo necessità dalla sintassi SQL e sostituiti nella stringa Request come spiegato in precedenza. La convenzione di chiamata standard è:

```
FUNZIONE Modifica (tabella AS stringa [, richiesta stringa AS, argomenti AS, ...]) COME risultato
```

Ecco un esempio di codice per mostrare come funziona Edit ():

```
DIM MyResult AS Result
DIM sQuery AS String
DIM iParm AS Integer

sQuery = "Seleziona * da tblDEFAULT dove id ="
'inseriremo un valore di parametro (12) alla fine della stringa di query
iParm = 12

$ hConn.Begin
MyResult = $ hConn.Edit ("tblDEFAULT", sQuery, iParm)
MyResult! Name = "Mr Rittinghouse" 'Imposta un valore
per il campo
$ hConn.Update 'Aggiorna con il nuovo valore
$ hConn.Commit 'rendere permanenti le modifiche
$ hConn.Close 'chiude la connessione al database
```

Exec

Il metodo Exec esegue una richiesta SQL e restituisce un oggetto Result di sola lettura contenente il risultato della richiesta. La convenzione di chiamata

Una guida per principianti a
standard è:

FUNZIONE Exec (Request AS String, Arguments AS, ...) AS Result

Una guida per principianti a

Richiesta è una clausola SQL WHERE e gli argomenti devono essere citati come richiesto dalla sintassi SQL.

Citazione

Il metodo Quote restituisce un identificatore tra virgolette che puoi utilizzare per inserire in una query SQL. Questo identificatore può essere una tabella o un nome di campo. La sintassi è:

```
FUNCTION Quote (Name AS String) AS String
```

Il meccanismo di quotazione dipende dal particolare driver del server di database che rende necessario questo metodo se è necessario scrivere codice indipendente dal database. Ecco un esempio di come utilizzare Quote:

```
'Restituisce il numero di record in una query
'sDBTable è il nome della tabella. Poiché il nome può includere
caratteri riservati, deve essere racchiuso tra virgolette

rResult = $ hConn.Exec ("SELECT COUNT (*) AS iNumRecs FROM" & DB.Quote
(sDBTable))
PRINT rResult! INumRecs
```

Oggetti risultato

L'oggetto risultato è una classe utilizzata per restituire il risultato di una richiesta SQL. Questa classe non è creabile e si comporta come un array di sola lettura. È enumerabile utilizzando l'istruzione FOR EACH. Dichiare e iterare un oggetto risultato come segue:

```
DIM hResult AS Result
FOR EACH hResult
  'fare qualcosa con i dati
  IL PROSSIMO
```

Lo snippet di codice seguente mostra come ottenere il valore di un campo nel record corrente di un oggetto Risultato:

```
DIM MyResult AS
Risultato DIM MyVariant
AS Variant

MyVariant = MyResult [Field AS String]
```

La classe ResultField rappresenta uno dei campi di un oggetto Result. È

Una guida per principianti a possibile utilizzare il metodo Trova per selezionare un campo particolare prima di utilizzarlo. Le proprietà supportate da questa classe sono Lunghezza, Nome, Risultato e Tipo. È anche possibile

Una guida per principianti a

enumerare i dati di ResultField con la parola chiave FOR EACH. Questa classe non è creabile. Il codice seguente mostra come modificare il valore di un campo nel record corrente di un oggetto Risultato:

```
DIM MyResult AS  
Risultato DIM MyVariant  
AS Variant  
  
MyResult [Field AS String] = MyVariant
```

Le proprietà che è possibile utilizzare con un oggetto risultato includono Disponibile, Connessione, Conteggio, Campi, Indice e Lunghezza. Available restituirà un valore TRUE se il risultato punta a un record di database esistente. Connection restituisce l'oggetto connessione padre. Count ti dice il numero di record (righe) restituiti con l'oggetto risultato. Fields restituisce una raccolta dei campi nella tabella del database restituita con l'oggetto risultato. Indice restituisce l'indice (puntatore o cursore della riga corrente) del record corrente, a partire da zero. La lunghezza è uguale a count e viene utilizzata in modo intercambiabile.

I metodi supportati dalla classe di oggetti risultato includono Delete, MoveFirst, MoveLast, MoveNext, MovePrevious, MoveTo e Update. La maggior parte di questi sono autoesplicativi, ma il metodo Update viene utilizzato per riscrivere il record corrente nel database con le modifiche ai dati specificate. Se usi MoveTo e specifichi un indice che è nullo o non valido, restituirà un valore TRUE che indica un errore. Ora che abbiamo una conoscenza di base di cosa sia un oggetto risultato e come usarlo, vediamo come Gambas ti permette di trovare dati in un database con il metodo Find.

Classe DB

Questa classe rappresenta la connessione al database corrente. Questa classe è statica e anche tutti i suoi membri sono statici. La maggior parte delle proprietà di questa classe sono le stesse della classe Connection, quindi tratteremo solo quelle diverse, ovvero Current, Databases e Debug. Current restituisce o imposta la connessione corrente. Database restituisce una raccolta di tutti i database gestiti dal server database. Debug imposta o restituisce TRUE se il componente del database è in modalità di debug. Quando il flag di debug è impostato, ogni query inviata a qualsiasi driver di database viene stampata sull'output di errore standard (generalmente, la console). I metodi utilizzati nella classe DB sono gli stessi di quelli utilizzati nella classe Connection (Begin, Close, Commit, Create, Edit, Exec, Find, Open, Quote e Rollback) quindi non ripeteremo queste informazioni qui.

Banca dati

La classe Database viene utilizzata per rappresentare le informazioni relative a un database. Questa classe non è creabile. Le proprietà che è possibile utilizzare con questa classe includono Connection, Name e System. La connessione può essere utilizzata per ottenere l'oggetto connessione padre. Name restituirà il nome del database e System restituirà un valore TRUE se il database è un database di sistema. La classe database ha un solo metodo, Delete, che eliminerà un database.

Campo

La classe Field viene utilizzata per rappresentare i dati per un campo della tabella. Questa classe non è creabile. Le proprietà supportate da questa classe sono Default, Length, Name, Table e Type. Default restituisce il valore predefinito di un campo. Lentgh restituirà la lunghezza massima di un campo di testo. Se il campo di testo da controllare non ha alcun limite specificato, viene restituito un valore zero. Name restituirà il nome del campo e Table restituirà l'oggetto tabella in cui è stato creato il campo. Il tipo restituisce il tipo di dati rappresentato dal campo. Gambas restituirà una delle seguenti costanti:

```
gb.Boolean gb.Integer gb.Float gb.Date ○ gb.String
```

Indice

La classe Index rappresenta un indice di tabella. Questa classe non è creabile. Le proprietà supportate da questa classe sono Fields, Name, Primary, Table e Unique. Fields restituisce un elenco di stringhe separate da virgole che rappresentano i campi che compongono l'indice. Name restituisce il nome dell'indice mentre Primary restituirà un valore TRUE se un indice è l'indice primario della tabella. Table restituirà l'oggetto tabella che possiede questo indice. Unique restituisce TRUE se l'indice è univoco.

tavolo

La classe Table rappresenta la definizione di una tabella di database. Questa classe non è creabile. Le proprietà supportate dalla classe Table includono Connection, Fields, Indexes, Name, PrimaryKey, System e Type. La connessione può essere utilizzata per ottenere l'oggetto connessione padre. Fields restituirà una raccolta dei campi della tabella mentre Indexes restituirà una raccolta degli indici della tabella. Name restituirà il nome della tabella. PrimaryKey restituirà o

Una guida per principianti a
imposterà la chiave primaria della tabella. La chiave primaria è un array di stringhe
contenente il

Una guida per principianti a

nome di ogni campo chiave primaria. Il sistema restituirà un valore TRUE se il database è un database di sistema. Type restituisce o imposta il tipo di una tabella. Questa proprietà è unica per MySQL e viene utilizzata solo con i database MySQL. Questa classe ha solo un metodo, Update, che viene chiamato per creare effettivamente la tabella nel database attualmente connesso.

Utente

La classe User viene utilizzata per rappresentare gli utenti di un database. Questa classe non è creabile. Le proprietà supportate dalla classe utente sono Amministratore, Connessione, Nome e Password. L'amministratore è una proprietà di sola lettura che restituisce TRUE se un utente è un amministratore del database. Connection restituisce l'oggetto di connessione padre dell'utente. Nome restituirà il nome dell'utente e Password otterrà o imposterà la password associata a un utente. Quando leggi la proprietà, dovrà ottenere la password in forma crittografata. La classe User ha un unico metodo, Delete, che viene utilizzato per eliminare un utente dal database.

Il programma di esempio del database

Le sezioni precedenti di questo capitolo hanno spiegato il componente del database Gambas e le caratteristiche in esso fornite. Tuttavia, niente sostituisce lavorare con codice reale in un'applicazione reale. Gambas viene fornito con un programma di database di esempio che esploreremo. Scritto dal fondatore di Gambas Benoit Minisini, questo programma è un eccellente esempio di come codificare il tuo programma per utilizzare il database all'interno dell'ambiente Gambas. Per iniziare, avvia Gambas e scegli il programma Database dal menu Esempi. Quando appare l'IDE, vedrai che il programma contiene due forme, FMain e FRequest. FMain dovrebbe assomigliare alla figura seguente.

Questo programma ti consentirà di scegliere quale piattaforma di database utilizzare (ad es. PostgreSQL, MySQL o SQLite) e di specificare il nome host, il nome del database, l'ID utente e la password. Esiste un'opzione per creare automaticamente un database e viene attivata ogni volta che la casella di controllo è selezionata. Dopo esserti connesso con successo al database, hai la possibilità di creare una tabella, eliminarla o riempirla con alcuni dati di prova. La casella di richiesta SQL nella parte inferiore del modulo consentirà di eseguire query SQL sul database. Tutto sommato, questo programma mostra quasi tutto ciò che devi sapere per connetterti a qualsiasi database utilizzando Gambas.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released under the terms of the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

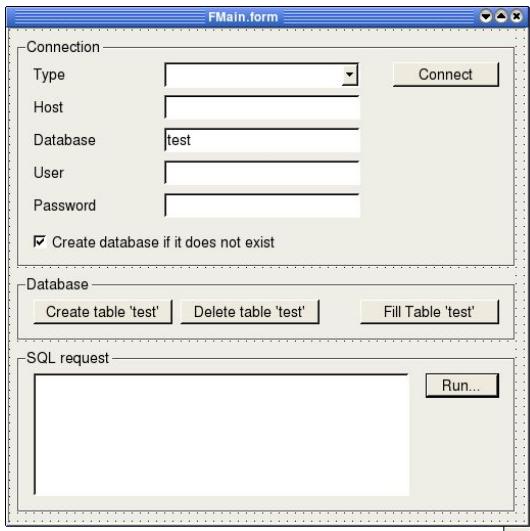


Figura 98 Il modulo FMain in modalità progettazione.

Il modulo FRequest è molto più semplice da costruire. È mostrato nella figura seguente:

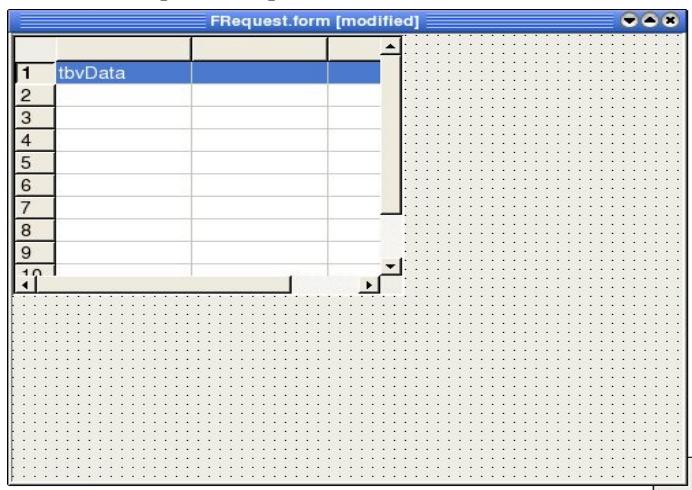


Figura 99 FRequest Form in modalità progettazione.

Il modulo FRequest contiene un singolo controllo, che non abbiamo introdotto in precedenza, il controllo TableView. Questo controllo si trova nel set di strumenti QT di cui abbiamo discusso in precedenza. Ti mostriremo come utilizzare questo controllo quando discuteremo il codice del modulo FRequest di seguito. Per ora, passiamo al codice per il modulo FMain e vediamo come funzionano le cose.

"File di classe Gambas

\$ hConn è dichiarata come variabile privata e verrà utilizzata da tutte le subroutine in

Una guida per principianti a
questo file di classe. Si noti che la variabile è preceduta da un simbolo \$ per indicare
che è un

This procedure (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the
Gambari Community under an Open Content License (OCL) and may not be
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

variabile di classe privata. Sebbene non richiesto, è una buona convenzione di programmazione da seguire.

Connessione \$ hConn AS PRIVATA

Ogni volta che l'utente ha compilato i dati nella parte superiore del modulo con il proprio nome utente, host, password, ecc. E fa clic sul pulsante di connessione, il programma arriverà a questa subroutine e proverà a utilizzare quei dati per stabilire una connessione a il database.

```
PUBLIC SUB btnConnect_Click ()  
    'abbiamo bisogno di una stringa con cui lavorare, quindi dichiara  
    sName come locale DIM sName AS String  
  
    'ora diventiamo un po' complicati. se un database è stato  
    precedentemente aperto 'ed è ancora aperto, la riga successiva lo  
    chiuderebbe. L'utilizzo di TRY 'consentirebbe di rilevare  
    qualsiasi errore (vedere CATCH di seguito).  
    "Se non è presente un database aperto da chiudere, non è  
    stato fatto alcun danno" e l'esecuzione del codice procede.
```

PROVA \$ hConn.Close

```
'assegniamo i dati nella casella di testo txtName alla nostra  
stringa var sName' e impostiamo le proprietà di connessione al  
database sui valori forniti 'dall'utente utilizzando il costrutto  
WITH / END WITH.
```

```
sName = txtName.Text  
CON $ hConn  
    .Type = cmbType.Text 'il tipo di piattaforma di database da  
    utilizzare  
    .Host = txtHost.Text 'nome host del database  
    .Login = txtUser.Text 'ID UTENTE  
    .Password = txtPassword.Text 'Password  
utente FINISCI CON
```

A questo punto, tutte le proprietà di connessione sono impostate ma non abbiamo ancora aperto la connessione. Il seguente segmento di codice vedrà se esiste un database con il nome specificato da sName e, in caso contrario, lo creerà se la casella di controllo per creare database è selezionata.

```
IF chkCreate.Value THEN  
    $ hConn.Open 'apri la connessione  
    'controlla se esiste un database sName e, in caso contrario,  
    aggiungilo al server SE NON $ hConn.Databases.Exist (sName) THEN  
        $ hConn.Databases.Add  
(sName) ENDIF
```

Una guida per principianti a
'ora chiudi la connessione
\$ hConn.Close

This product is (C) 2005 by John W. Rittinghouse, all rights reserved. It is released to the
Gnostice User Community under an Open Content License (OCL) and may not be distributed
under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

FINISCI SE

Ora siamo pronti per aprire il database. O esisteva già e apriamo quel database o apriremo quello nuovo che abbiamo creato se la casella di controllo è stata selezionata. Innanzitutto, imposta la proprietà Name per la connessione su sName e quindi chiama il metodo Open:

```
$ hConn.Name = sName
$ hConn.Open

'impostiamo le proprietà abilitate per entrambi i moduli del
programma su TRUE' e inseriamo del testo predefinito nella
casella della richiesta SQL in 'fondo del modulo FMain:

frmDatabase.Enabled = TRUE
frmRequest.Enabled = TRUE
txtRequest.Text = "MOSTRA TABELLE"

'se il nostro TRY non è riuscito o se si è verificato un errore
durante questa subroutine,' l'istruzione CATCH visualizzerà una
finestra di dialogo ERROR con il testo del 'messaggio di errore.
```

CATTURARE

```
Message.Error (Error.Text)
FINE 'della subroutine connectBtn_Click
```

Se, dopo aver stabilito la connessione al database, l'utente fa clic sul pulsante Crea tabella "test", viene eseguita questa subroutine. L'ho leggermente modificato per evitare di utilizzare stringhe codificate e per utilizzare lo stesso nome del database. Il codice originale è commentato in modo da poter utilizzare entrambi i metodi.

```
PUBLIC SUB btnCreate_Click ()
'dichiara una variabile di
tabella locale Tabella DIM
hTable AS

'aggiungi la tabella al database utilizzando il
metodo Add' hTable = $ hConn.Tables.Add ("test")

'questo è il codice che ho modificato per evitare di
codificare la stringa' come è stato fatto nella riga di
codice precedente
hTable = $ hConn.Tables.Add (txtName.text)

"ora aggiungi i campi della tabella alla tabella che
abbiamo appena aggiunto" nota che specifichiamo il nome del
campo e il tipo di dati per ciascuno dei campi che
inseriamo nella tabella. Per i dati stringa, 'specifichiamo
```

Una guida per principianti a
anche la lunghezza massima del campo
`stringa.hTable.Fields.Add ("id", gb.Integer)
hTable.Fields.Add ("firstname", gb.String, 16)`

This product is (C) 2005 by John W. Rittinghouse. It is released to the Gamasutra Community under an Open Content License and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
hTable.Fields.Add ("name", gb.String,
32) hTable.Fields.Add ("birth", gb.Date)
hTable.Fields.Add ("active", gb.Boolean)
hTable.Fields.Add ("stipendio",
gb.Float)

'possiamo specificare una chiave primaria per la tabella del
database. In questo caso, viene utilizzato il campo id poiché
possiamo creare dati di indice intero univoco

hTable.PrimaryKey = ["id"]

'la chiamata al metodo Update salva le modifiche al database' e le
rende permanenti.

hTable.Update

'ora mettiamo in piedi una semplice finestra di dialogo per
informare l'utente che abbiamo' completato con successo l'attività
di creazione.

Message.Info ("Table" & txtName.Text & "è stato creato.")
```

Questo segmento di codice seguente è stato aggiunto dall'autore per abilitare o disabilitare i pulsanti a seconda dello stato del database. Ad esempio, se sono già presenti dati nella tabella del database, non ha senso riempire nuovamente quella tabella, in modo che il pulsante venga disabilitato. Tuttavia, il pulsante Elimina sarebbe abilitato in modo che i dati possano essere rimossi. Se la tabella del database viene eliminata, verrà riattivata e se la tabella viene creata e non esistono dati, il pulsante di riempimento verrà nuovamente abilitato, ecc. Ciò impedisce agli errori di fare clic su un pulsante in cui un'azione non può essere eseguita correttamente.

```
btnFill.Enabled = TRUE
btnDelete.Enabled = TRUE
btnCreate.Enabled = FALSE

'successivamente, aggiungiamo del testo predefinito alla casella
delle query SQL txtRequest.Text = "Mostra TABELLE"

'l'istruzione CATCH visualizzerebbe una finestra di dialogo ERROR
con il testo del messaggio di errore se si verifica un errore in
questa subroutine.

CATTURARE
Message.Error (Error.Text)
FINE 'della subroutine del pulsante di creazione
```

Una guida per principianti a

Se, dopo aver stabilito la connessione al database, l'utente fa clic sul pulsante Elimina tabella "test", viene eseguita questa subroutine. ho anche

Una guida per principianti a

modificato leggermente per evitare di utilizzare stringhe hardcoded e per utilizzare lo stesso nome del database. Il codice originale è commentato in modo da poter utilizzare entrambi i metodi.

```
PUBLIC SUB btnDelete_Click ()  
  
    'rimuovi la tabella' $  
    hConn.Tables.Remove ("test")  
    $ hConn.Tables.Remove (txtName.Text)  
  
    'ha inserito un messaggio per informare l'utente che la tabella è  
    sparita Message.Info ("Tabella" & txtName.Text & "è stato  
    rimosso")  
  
    'abilita o disabilita i pulsanti per avere un senso. Se la  
    tabella 'è stata rimossa, non può essere cancellata di  
    nuovo e non può essere riempita poiché non esiste. Non  
    resta che 'ricrearlo e quel pulsante è abilitato.  
  
    btnDelete.Enabled = FALSE  
    btnFill.Enabled = FALSE  
    btnCreate.Enabled = TRUE  
  
    'non esiste alcuna tabella da interrogare, quindi cancella il  
    testo della query SQL. txtRequest.Text = ""  
  
    "Se si verifica un errore nella subroutine, catturalo qui e mostra un  
messaggio di dialogo" all'utente.  
  
CATTURARE  
    Message.Error (Error.Text)  
FINE 'la subroutine del pulsante Elimina'
```

Se la connessione al database è stata stabilita e l'utente ha creato la tabella, è necessario aggiungere alcuni dati per renderla utile. Questa subroutine viene eseguita e aggiungerà dati semiarbitrari alla tabella. Selezionerà casualmente un nome dalla matrice di cinque nomi forniti e concatenerà il numero del contatore alla stringa "Nome #" al server come cognome. La data di nascita viene creata casualmente scegliendo un valore che aggiunge un numero casuale da 110.000 alla data di base del 1 gennaio 1970. Il record ha un flag attivo, anch'esso impostato casualmente. I dati sui salari vengono selezionati in modo casuale nell'intervallo compreso tra 1.000 e 10.000. Infine, il metodo Update inserirà tutti i dati nella tabella e farà il COMMIT al database. Diamo un'occhiata a ciascuna riga di codice per vedere quanto è facile farlo in Gambas:

```
PUBLIC SUB btnFill_Click ()
```

Una guida per principianti a
'abbiamo bisogno di un contatore intero iInd per essere il nostro
indice e noi

Una guida per principianti a

```
'necessita di un oggetto Risultato per
memorizzare i nostri risultati DIM iInd AS
Integer
DIM rTest AS Risultato

'imposta il flag Occupato per evitare interruzioni al nostro
processo Applicazione INC

'inizieremo il processo di transazione del database con BEGIN
$ hConn.Begin
'crea prima la tabella del
database' rTest = $ hConn.Create
("test") rTest = $ hConn.Create
(txtName.Text)

'ora, imposta un ciclo per creare 100
record PER iInd = da 1 a 100
'rende l'ID del record il valore della variabile
del contatore rTest! id = iInd

'imposta in modo casuale il primo nome in modo che sia uno dei cinque
nell'array
rTest! firstname = ["Paul", "Pierre", "Jacques", "Antoine", "Mathieu"] [Int (Rnd (5))]

'rende il cognome un valore catenato con il valore di indice intero
rTest! name = "Name #" & iInd

"scegli a caso una data aggiungendo un valore compreso tra 1 e
10.000 alla" data di base del 1 gennaio 1970
rTest! birth = CDate ("01/01/1970") + Int (Rnd (10000))

'imposta il flag attivo su 0 o 1 (VERO o FALSO) rTest! active =
Int (Rnd (2))

'scegli a caso una cifra di stipendio da 1.000 a 10.000 rTest!
salary = Int (Rnd (1000, 10000))

'aggiorna questo record con i valori dei dati semiarbitrari
rTest.Update
IL PROSSIMO 'iterazione del ciclo

'salva tutti i record aggiunti nel database
$ hConn.Commit

'ultima cosa da eseguire prima di lasciare la
subroutine INFINE
'decrementa la bandiera
di occupato Applicazione
DEC

'compare un messaggio per informare l'utente di ciò che abbiamo
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a
fatto **Message.Info** (**txtName.Text & "è stato riempito."**)

This product is (C) 2005 by John W. Williams. All rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
'inserisce una query SQL predefinita nella casella di testo Query SQL txtRequest.Text = "seleziona * da" & txtName.Text

'aggiusta i nostri pulsanti per avere un senso
btnFill.Enabled = FALSE
btnDelete.Enabled = TRUE
btnCreate.Enabled = FALSE

'se qualcosa va storto, annulla tutte le modifiche con il rollback
e' mostra il messaggio di errore all'utente
CATTURARE
$ hConn.Rollback
Message.Error
(Error.Text)
FINE 'della subroutine dei dati di riempimento
```

Se l'utente immette una richiesta SQL arbitraria nella casella della query SQL nella parte inferiore del modulo, utilizzeremo il metodo Exec per eseguire la query e visualizzare i risultati della richiesta utilizzando il modulo FRequest. Ecco come si fa:

```
PUBLIC SUB btnRun_Click ()
'dichiara un oggetto risultato per contenere i risultati della query DIM rData AS Risultato

'dichiara una variabile del modulo in modo da poter mostrare il modulo FRequest DIM hForm AS FRequest

'esegue la query utilizzando il metodo Exec e assegna i risultati all'oggetto risultato rData
rData = $ hConn.Exec (txtRequest.Text)

'passa l'handle di connessione al database ei dati del risultato al modulo' FRequest quando viene istanziato
hForm = NEW FRequest ($ hConn, rData)

'ora mostra all'utente il modulo con i dati del risultato
hForm.Show

'vieni qui se c'è un problema nella subroutine e mostra un messaggio
CATTURARE
Message.Error (Error.Text)
END 'della subroutine query
sql
```

Ogni volta che il nostro programma viene eseguito e il form FMain si apre, dobbiamo istanziare la nostra variabile di connessione \$ hConn. Ogni volta che il

Una guida per principianti a
modulo viene chiuso, chiuderemo la connessione.

```
PUBLIC SUB Form_Open ()
```

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
$ hConn = NUOVA  
connessione FINE  
  
PUBLIC SUB Form_Close ()  
$  
hConn.Close  
END
```

Successivamente, dobbiamo dare un'occhiata al modulo FRequest e vedere come quel codice visualizza i dati dell'oggetto risultato quando viene richiamato. Ecco il codice per questo:

```
"File di classe Gambas  
'fRequest.class dichiara due variabili private da usare in questa  
classe' una per la connessione, una per i dati del risultato.  
PRIVATE $ hConn AS  
Connessione PRIVATE $ rData  
AS Risultato  
  
'verrà utilizzata una routine di costruzione per ricevere l'handle  
di connessione' ei risultati di una query come parametri quando  
FRequest viene chiamato dal 'modulo FMain'.  
  
PUBLIC SUB _new (hConn AS Connection, rData AS Risultato)  
'assegna il parametro hConn alla nostra variabile $ hConn  
privata  
$ hConn = hConn  
  
'assegna il parametro rData alla nostra variabile privata $  
rData' queste assegnazioni sono fatte in modo che le variabili  
con prefisso $ siano 'visibili all'intera classe, non solo alla  
routine del costruttore.  
$ rData = rData  
  
'chiama la nostra piccola subroutine per visualizzare il titolo nella  
finestra del form RefreshTitle  
  
'la subroutine ReadData viene utilizzata per popolare il nostro  
controllo TableView ReadData  
  
'ridimensiona la finestra per centrarla sul desktop ME.Move (Int  
(Rnd (Desktop.W ME.W)), Int (Rnd ((Desktop.H ME.H))))  
FINE 'del costruttore  
  
'questa subroutine aggiorna semplicemente la didascalia  
della finestra PRIVATE SUB RefreshTitle ()  
  
'abbiamo bisogno di una  
variabile stringa locale DIM
```

Una guida per principianti a

sTitle AS String

```
'concateneremo il nome della connessione al testo della didascalia  
sTitle = ("Risultati query SQL") & "" & $ hConn.Name
```

Una guida per principianti a

```
'e imposta la proprietà title come valore della variabile stringa  
ME.Title = sTitle  
FINE
```

La subroutine ReadData () viene utilizzata per definire la struttura dei dati passati dall'oggetto results al controllo TableView. Determina il numero di campi dall'oggetto risultati e imposta le colonne nella tabella, assegnando il nome della colonna (campo) e il tipo di dati appropriati. Il tipo di dati viene determinato chiamando un'altra subroutine, WidthFromType () per ottenere quelle informazioni.

Il controllo TableView basato sulla libreria QT è obsoleto nella libreria QT, essendo stato sostituito con il controllo QTable. Tuttavia, in Gambas, è ancora utilizzabile e ti mostreremo come si fa. Tieni presente, tuttavia, che non è raccomandato per l'uso poiché Trolltech, i proprietari della libreria QT, hanno dichiarato pubblicamente il controllo TableView obsoleto. Gambas TableView è basato su QTable e QTableWidgetItem, entrambi diventeranno obsoleti in Qt 4.0. TableView sarà molto probabilmente basato su QTableView o QTableWidget invece.

```
PRIVATE SUB ReadData ()  
    'questa variabile è dichiarata ma mai usata, puoi commentarla  
    Tabella DIM hTable AS  
  
    DIM hField AS ResultField  
  
    'questa variabile è dichiarata ma mai usata, puoi commentarla DIM  
    sField AS String  
    DIM iInd AS Integer  
  
    'questa variabile è dichiarata ma mai usata, puoi commentarla DIM  
    iLen AS Integer  
  
    'imposta il flag di occupato dell'applicazione per  
    evitare interruzioni Applicazione INC  
  
    'reimposta il conteggio delle righe del controllo TableView a zero  
    tbvData.Rows.Count = 0  
  
    'imposta il numero di colonne in modo che sia lo  
    stesso numero' del numero di campi nell'oggetto  
    risultato tbvData.Columns.Count = $  
    rData.Fields.Count  
  
    'ora itereremo attraverso ogni campo dell'oggetto risultato e'  
    otterremo il nome del campo e il tipo di dati e imposteremo le  
    intestazioni di colonna di TableView 'e il tipo / dimensione del  
    campo come appropriato
```

This program (c) 2005 by Jon W Rittinghouse, all rights are reserved. It is released to the public domain under the terms of the Open Content License (OCL) and may not be distributed without the express written consent of the author.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved.
Gambas User Community under an Open Content License (OCL).
under any other terms or conditions without the express written permission of the author.

Una guida per principianti a

```
PER OGNI hField IN $  
rData.Fields CON hField  
'questa è una riga di debug commentata' PRINT  
.Name; ":"; .Genere; ""; .Lunghezza  
  
'questa riga successiva imposta il nome della colonna come nome  
del campo tbvData.Columns [iInd] .Text = .Name  
  
'qui, la chiamata a WidthFromType viene utilizzata per determinare  
quale tipo di dati è il campo e quale dovrebbe essere la larghezza  
appropriata tbvData.Columns [iInd] .Width = WidthFromType (tbvData, .Type,  
.Length, .Name)  
FINISCI CON  
'incrementa il nostro  
contatore dell'indice INC  
iInd  
IL PROSSIMO 'iterazione dei dati dei risultati  
  
'imposta il numero di righe TableView in modo che sia uguale al  
numero' di righe di dati nell'oggetto risultato  
tbvData.Rows.Count = $ rData.Count  
  
L'ultima cosa che facciamo in questa subroutine è diminuire  
l'indicatore di occupato INFINE  
Applicazione DEC  
  
'se ci fossero degli errori in questa subroutine mostreremmo un  
messaggio CATTURARE  
Message.Error ("Impossibile eseguire la richiesta." & "\n \" & Error.Text) END 'della subroutine ReadData
```

Il bit di codice successivo viene attivato ogni volta che sono presenti dati. L'evento Data del controllo TableView viene utilizzato come driver per l'attivazione di questa subroutine. Inserisce una riga di dati dall'oggetto risultati nella riga e colonna correnti del controllo TableView. Tieni presente che non chiami mai direttamente l'evento dati. È disponibile pochissima documentazione per il controllo TableView, quindi nulla di quanto dichiarato nel presente documento deve essere considerato come la risposta definitiva per il suo utilizzo. Tuttavia, l'approccio generale all'utilizzo del controllo TableView consiste nel caricare prima tutti i dati che devono essere visualizzati in una matrice e preparare il controllo TableView con righe e colonne come è stato fatto nella subroutine ReadData sopra. Lo stesso processo di riempimento dell'array e definizione delle colonne in questo modo forza il widget Qt utilizzato nel controllo TableView a effettuare una chiamata interna all'evento Data che effettivamente riempie le celle con i dati dell'array. Spetta a te codificare le istruzioni di assegnazione che spostano i dati dall'oggetto dei risultati alla riga e alla colonna TableView in cui desideri visualizzare i dati. La cosa fondamentale

Una guida per principianti a
da ricordare è che devi avere un gestore di eventi, tbvControl_Data (...) da
qualche parte nel tuo file di classe per farlo. In un controllo TableView, i dati che
sono La cosa fondamentale da ricordare è che devi avere un gestore di eventi,
tbvControl_Data (...) da qualche parte nel tuo file di classe per farlo. In un
controllo TableView, i dati che sono La cosa fondamentale da ricordare è che
devi avere un gestore di eventi, tbvControl_Data (...) da qualche parte nel tuo file
di classe per farlo. In un controllo TableView, i dati che sono

Una guida per principianti a

aggiornato è solo ciò che è visibile nel controllo. Il controllo può visualizzare un'enorme quantità di dati senza conservare alcun valore dei dati direttamente nella memoria stessa. Questo è ciò per cui viene utilizzato il tuo array di programmi. Anche se sembra un po 'imbarazzante, è una soluzione pragmatica che consente a Gambas di visualizzare i dati da un database in modo abbastanza efficiente.

```
PUBLIC SUB tbvData_Data (Row AS Integer, Column AS Integer)
    'questo arriva alla riga corretta
    $ rData.MoveTo (riga)
    'questo assegna effettivamente i dati dall'oggetto risultati alla
    visualizzazione tabella tbvData.Data.Text = Str ($ rData
        [tbvData.Columns [Column] .Text])
FINE
```

Se il modulo viene ridimensionato dall'utente, è necessario ridimensionare il controllo. Questa subroutine farà il trucco.

```
PUBLIC SUB Form_Resize () tbvData.Resize  
    (ME.ClientW, ME.ClientH)  
FINE
```

Quando i dati vengono letti dall'oggetto risultati, questa subroutine viene chiamata per determinare quale sia il tipo di dati per ogni campo di una riga. Le proprietà della classe Field vengono passate come parametri a questa funzione e utilizzate nell'istruzione select. Ciò garantisce che i dati inseriti nelle singole celle del controllo TableView non vengano troncati e vengano visualizzati correttamente, indipendentemente dall'impostazione della proprietà del carattere stabilita dal controllo TableView stesso.

```
FUNZIONE PRIVATA WidthFromType (hCtrl AS control, iType AS Integer
iLength AS Integer, sTitle AS String) AS Integer

    DIM iWidth AS Integer

    SELECT CASE iType
        CASO gb.Boolean
            iWidth = hCtrl.Font.Width (Str (FALSE)) + 32

        CASE gb.Integer
            iWidth = hCtrl.Font.Width ("1234567890") + 16

        CASE gb.Float
            iWidth = hCtrl.Font.Width (CStr (Pi) & "E + 999") + 16

        CASE gb.Date
            iWidth = hCtrl.Font.Width (Str (ora)) +
```

Una guida per principianti a
16 CASE gb.String

This product is per C) 2005 by John W. Williams, all rights are reserved. It is released to the Gambas User Community under a Creative Commons License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

```
IF iLength = 0 THEN iLength = 255
iLength = Min (32, iLength)
iWidth = hCtrl.Font.Width ("X") * iLength +
16 END SELECT
iWidth = Max (iWidth, hCtrl.Font.Width (sTitle) + 8)
RETURN iWidth
FINE
```

Infine, se l'utente fa clic sulla "x" nell'angolo in alto a destra della finestra per chiudere il programma, arriviamo a questa routine e chiudiamo le cose.

```
PUBLIC SUB Form_Close
() ME.Close
FINE
```

Quando si esegue il programma Database di esempio, è necessario disporre di un sistema di database come PostgreSQL o MySQL installato sul sistema. È necessario disporre di un account utente e di una password per utilizzare i sistemi di database. Dati questi avvertimenti, eseguire il programma e accedere al database. Puoi giocare, creare e riempire la tabella di test e fare domande con il programma. A questo punto, hai appreso tutte le basi per la connessione e l'utilizzo di un database in Gambas. Nel prossimo capitolo tratteremo gli elementi essenziali per rendere il tuo programma disponibile agli utenti a livello globale, un processo di globalizzazione che coinvolge due processi, internazionalizzazione (I18N) e localizzazione (L10N).

Una guida per principianti a

Capitolo 16 - Global Gambas

Gambas è stato progettato fin dall'inizio per supportare una comunità internazionale. Essendo un prodotto open source, persone provenienti da tutto il mondo partecipano alla cura e allo sviluppo continuo del prodotto Gambas. La capacità che Gambas fornisce agli sviluppatori di creare una singola base di codice che soddisferà gli utenti di ogni paese è un passo significativo nel riconoscimento di una comunità globale. Gambas consente agli sviluppatori di scrivere programmi facilmente traducibili in quasi tutte le lingue, senza dover riscrivere il codice e senza dover assumere una costosa azienda di outsourcing per svolgere il lavoro. I processi di internazionalizzazione e localizzazione sono notevolmente migliorati dall'approccio Gambas. Le sezioni seguenti forniranno una panoramica di cui ogni sviluppatore Gambas dovrebbe essere a conoscenza per produrre codice globalmente accettabile. Noi siamo,

Internazionalizzazione

Internazionalizzazione è il processo di progettazione di un'applicazione in modo che possa essere adattata a varie lingue e regioni senza modifiche tecniche. A volte il termine internazionalizzazione è abbreviato in i18n, perché ci sono 18 lettere tra la prima "i" e l'ultima "n". Un programma internazionalizzato, con l'aggiunta di dati localizzati, può utilizzare lo stesso eseguibile in tutto il mondo. Gli elementi testuali, come i messaggi di stato e le etichette dei componenti della GUI, non sono codificati nel programma. Vengono invece archiviati al di fuori del codice sorgente e recuperati dinamicamente. Il supporto per le nuove lingue non richiede la ricompilazione. I dati culturalmente dipendenti, come date e valute, vengono visualizzati in formati conformi alla regione e alla lingua dell'utente finale. In questo modo, un'applicazione internazionalizzata può essere localizzata molto rapidamente.

Localizzazione

Localizzazione è il processo di adattamento del software per una regione o una lingua specifica aggiungendo componenti locali e traducendo il testo. Il termine localizzazione è spesso abbreviato in L10N, perché ci sono 10 lettere tra la "L" e la "N." Di solito, la parte che richiede più tempo del processo di localizzazione è la traduzione del testo. Altri tipi di dati, come suoni e immagini, possono richiedere la localizzazione se sono culturalmente sensibili. Gli ingegneri della localizzazione verificano che la formattazione di date, numeri e valute sia conforme ai requisiti locali.

Una guida per principianti a

Set di caratteri universale (UCS)

ISO / IEC 10646 definisce un set di caratteri molto ampio chiamato Universal Character Set (UCS), che comprende la maggior parte dei sistemi di scrittura del mondo. Lo stesso set di caratteri è definito anche dallo standard Unicode. Ad oggi, le modifiche in Unicode e le modifiche e le aggiunte a ISO / IEC 10646 sono state apportate all'unisono in modo che i repertori dei caratteri e le assegnazioni dei punti di codice siano rimasti sincronizzati. I comitati di normalizzazione competenti si sono impegnati a mantenere questo approccio.

Sia UCS che Unicode sono tabelle di codici che assegnano un valore intero a un carattere. Esistono diverse alternative per la rappresentazione di una stringa di tali caratteri (o dei loro valori interi) come sequenza di byte. Le due forme di codifica più ovvie memorizzano il testo Unicode come sequenze di 2 o 4 byte. Questi schemi di codifica sono designati come UCS2 e UCS4. In entrambi questi schemi, il byte più significativo viene per primo (formato Big Endian). I caratteri in Unicode sono identificati dal loro numero o "punto di codice" che di solito è dato in notazione esadecimale. Ad esempio, in ebraico, la lettera "he" è il punto di codice 5D4 che di solito viene scritto con la convenzione di anteporre un prefisso "U +" davanti al punto di codice e riempirlo con zeri iniziali secondo necessità, ad esempio U+05D4.

Unicode

Unicode attualmente definisce poco meno di 100.000 caratteri, ma ha spazio per 1.114.112 punti di codice¹⁹. I punti di codice sono organizzati in 17 "piani" di 216 (65.536) caratteri, numerati da 0 a 16. Il piano 0 è chiamato "Piano multilingue di base" o BMP e contiene simboli ritenuti utili. In generale, contiene tutti i caratteri disponibili per un programmatore prima che arrivasse Unicode. I caratteri nel BMP sono distribuiti in modo "da ovest a est", con i caratteri ASCII che hanno i loro valori ASCII familiari da 0 a 127, i caratteri ISOLatin1 che mantengono i loro valori da 128 a 255. Quindi, i set di caratteri si spostano verso est in tutta Europa (Greco, cirillico), in Medio Oriente (arabo, ebraico, ecc.) E attraverso l'India fino al sud-est asiatico, per finire con i set di caratteri da Cina, Giappone e Corea (CJK). Oltre questo BMP esistono i piani da 1 a 16.

¹⁹ The reader è incoraggiato a visitare <http://www.tbray.org/ongoing/When/200x/2003/04/26=UTF> for un full spiegazione.

Una guida per principianti a

I file ASCII possono essere convertiti in un file UCS2 inserendo 0x00 davanti a ogni byte ASCII. Per creare un file formattato UCS4, è necessario inserire tre byte 0x00 prima di ogni byte ASCII. L'utilizzo di UCS2 o UCS4 su piattaforme Unix può portare a problemi molto significativi. Le stringhe utilizzate con questi schemi di codifica possono contenere parti di molti byte di caratteri larghi come "\ 0" o "/" che hanno un significato speciale nei nomi dei file e in altre funzioni della libreria C. La stragrande maggioranza degli strumenti UNIX prevede di operare con file basati su ASCII e semplicemente non può leggere parole a 16 bit come caratteri. In poche parole, UCS2 / UCS4 non sono adatti per Unix se usati con nomi di file, file di testo, variabili d'ambiente, librerie di codice, ecc.

Esistono molte altre forme di codifica comuni a UCS e Unicode, ovvero UTF8, UTF16 e UTF32. In ciascuno di questi formati di codifica, ogni carattere è rappresentato come una o più unità di codifica. Tutti i formati di codifica UCS standard eccetto UTF8 hanno un'unità di codifica maggiore di un ottetto. Per molte applicazioni e protocolli, il presupposto di un set di caratteri a 8 o 7 bit rende quasi impossibile l'uso di qualcosa di diverso da UTF8. Poiché UTF8 ha un'unità di codifica oneoctet, utilizza tutti i bit di un ottetto ma può comunque contenere l'intera gamma di caratteri USASCII, che sono tutti codificati in un singolo ottetto. Qualsiasi ottetto con tale valore può rappresentare solo un carattere USASCII.

UTF-8

UTF8 è stato inventato da Ken Thompson²⁰. Secondo la storia di Rob Pike, che all'epoca era con Ken, fu sviluppato durante le ore serali del 2 settembre 1992 in un ristorante del New Jersey. Ken lo ha disegnato in presenza di Rob Pike su una tovaglietta (vedi la storia dell'UTF8 di Rob Pike²¹). Il nuovo progetto ha sostituito un precedente tentativo di progettare un File System Safe UCS Transformation Format (FSS / UTF) che è stato diffuso in un documento di lavoro X / Open nell'agosto dello stesso anno da Gary Miller (IBM), Greger Leijonhufvud e John Entenmann (SMI) in sostituzione della pesante codifica UTF1 presentata in ISO 10646²². Entro una settimana da quel periodo, Pike e Thompson hanno reso Plan 9 di AT&T Bell Lab il primo sistema operativo al mondo a utilizzare la codifica UTF8. Lo hanno riferito alla conferenza tecnica USENIX Winter 1993²³.

20 <http://www.cs.bell-labs.com/who/ken/>

21 <http://www.cl.cam.ac.uk/~mgk25/ucs/utf-8-history.txt>

22 <http://64.233.167.104/search?q=cache:U4zV7dcaq0EJ:www.faqs.org/ftp/rfc/pdf/rfc2279.txt.pdf+Greger+Leijonhufvud&hl=en>

23 USENIX Technical Conference tenutasi a San Diego, CA, 25-29 gennaio 1993, pubblicata negli Atti della conferenza, pp. 43-50.

Una guida per principianti a

UTF8 codifica i caratteri UCS come un numero variabile di ottetti. Il numero di ottetti e il valore di ciascuno dipendono dal valore intero assegnato al carattere in ISO / IEC 10646 (il numero del carattere, noto anche come posizione del codice, punto di codice o valore scalare Unicode). Questa forma di codifica ha le seguenti caratteristiche²⁴ (valori in esadecimale):

- I numeri di caratteri da U + 0000 a U + 007F (set USASCII) corrispondono agli ottetti da 00 a 7F (valori USASCII a 7 bit). Una conseguenza diretta è che anche una semplice stringa ASCII è una stringa UTF8 valida.
- I valori dell'ottetto USASCII non vengono visualizzati altrimenti in un flusso di caratteri con codifica UTF8. Ciò fornisce compatibilità con i file system o altri software (ad esempio, la funzione printf () nelle librerie C) che analizzano in base ai valori USASCII ma sono trasparenti ad altri valori.
- La conversione di andata e ritorno è facile tra UTF8 e altre forme di codifica.
- Il primo ottetto di una sequenza multiocet indica il numero di ottetti nella sequenza.
- I valori dell'ottetto da C0, C1, da F5 a FF non vengono mai visualizzati.
- I confini dei caratteri sono facilmente individuabili da qualsiasi punto di un flusso di ottetti.
- L'ordinamento lessicografico di bytevalue delle stringhe UTF8 è lo stesso se ordinato per numeri di caratteri. Ovviamente questo è di interesse limitato poiché un ordinamento basato sui numeri dei caratteri non è quasi mai culturalmente valido.
- L'algoritmo di ricerca rapida BoyerMoore può essere utilizzato con i dati UTF8.
- Le stringhe UTF8 possono essere riconosciute in modo abbastanza affidabile come tali da un semplice algoritmo, ovvero la probabilità che una stringa di caratteri in qualsiasi altra codifica appaia come UTF8 valido è bassa e diminuisce con l'aumentare della lunghezza della stringa.

La codifica UTF8 è formalmente definita nell'ISO 10646: 2000 Allegato D. È anche descritta nella RFC 3629 e nella sezione 3.9 dello standard Unicode 4.0.

Una guida per principianti a

24 See URL <http://www.faqs.org/rfcs/rfc2279.html>.

Una guida per principianti a

Per poter utilizzare Unicode con sistemi operativi in stile Unix, UTF8 è il miglior schema di codifica da utilizzare. Questo perché i caratteri UCS da U + 0000 a U + 007F (ASCII) sono codificati come byte da 0x00 a 0x7F (per compatibilità ASCII). Ciò significa che i file e le stringhe che contengono solo caratteri ASCII a 7 bit avranno la stessa codifica sia in ASCII che in UTF8. Tutti i caratteri UCS che rientrano nell'intervallo maggiore di U + 007F sono codificati come una sequenza multibyte, ciascuno dei quali ha impostato il bit più significativo. Pertanto, nessun byte ASCII (0x00-0x7F) può apparire come parte di qualsiasi altro carattere.

Il primo byte di una sequenza multibyte che rappresenta un carattere non ASCII ricadrà sempre nell'intervallo da 0xC0 a 0xFD e indica quanti byte seguiranno per questo carattere. Eventuali ulteriori byte in una sequenza multibyte saranno nell'intervallo da 0x80 a 0xBF. Ciò consente una facile risincronizzazione e rende la codifica senza stato e robusta contro i byte mancanti. Tutti i possibili codici 231 UCS possono essere codificati. I caratteri codificati UTF8 possono teoricamente essere lunghi fino a sei byte. L'ordine di ordinamento delle stringhe di byte Big Endian UCS4 viene mantenuto. I byte 0xFE e 0xFF non vengono mai utilizzati nella codifica UTF8. Le seguenti sequenze di byte vengono utilizzate per rappresentare un carattere. La sequenza da utilizzare dipende dal numero Unicode del carattere:

U-00000000 - U-0000007F: 0xxxxxx
U-00000080 - U-000007FF: 110xxxxx 10xxxxxx
U-00000800 - U-0000FFFF: 1110xxxx 10xxxxxx 10xxxxxx
U-00010000 - U-001FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
U-00200000 - U-03FFFFFF: 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
U-04000000 - U-7FFFFFFF: 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

Le posizioni dei bit xxx vengono riempite con i bit del numero di codice del carattere nella rappresentazione binaria. Il bit x più a destra è il bit meno significativo. È possibile utilizzare solo la sequenza multibyte più breve possibile che può rappresentare il numero di codice del carattere. Notare che nelle sequenze multibyte, il numero di bit iniziali nel primo byte è identico al numero di byte nell'intera sequenza.

Come tradurre in Gambas

Per tradurre Gambas nella tua lingua, apri il progetto Gambas nell'IDE e fai clic su Traduci ... nel menu Progetto, come mostrato nella figura seguente:

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambas User Community under a Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a



Figura 100 Scelta dell'opzione Traduci in Gambas.

Successivamente, è necessario selezionare la lingua di traduzione di destinazione nella casella combinata della finestra di dialogo di traduzione, come mostrato di seguito:

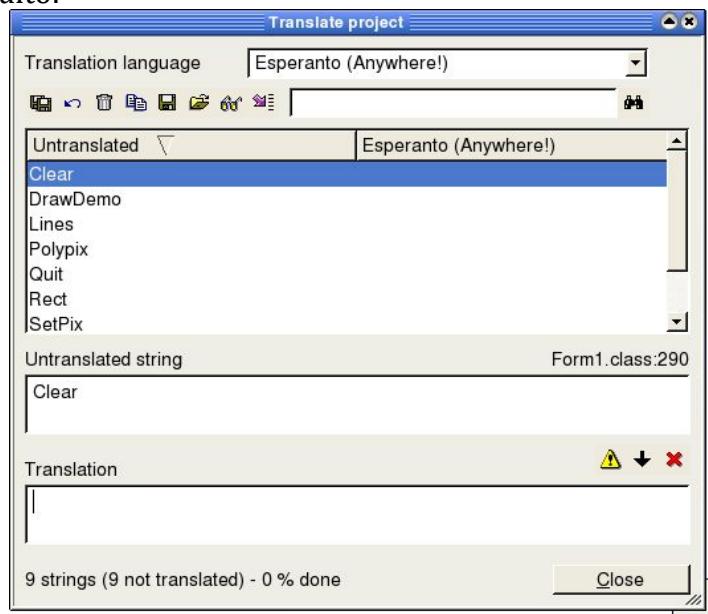


Figura 101 La finestra di dialogo Traduci in Gambas.

Seleziona una stringa dall'elenco delle stringhe non tradotte e inserisci la sua traduzione nel campo di testo mostrato nella parte inferiore della finestra di dialogo. È necessario ripetere questo processo fino a quando tutte le stringhe nell'applicazione non sono state tradotte. Al termine, fare clic sul pulsante Chiudi. Non è necessario completare tutte le corde in una volta sola. Puoi tradurre il progetto in diversi tentativi. Le icone della barra degli strumenti che vedi nella parte

Una guida per principianti a
superiore della finestra di dialogo Traduci ti consentono di:

Una guida per principianti a

- Salva la traduzione corrente.
- Ricarica la traduzione corrente (Nota: le modifiche vengono perse quando ricarichi).
- Elimina la traduzione corrente.
- Duplica una traduzione esistente.
- Esporta la traduzione corrente in un file .po.
- Unisci un file di traduzione.
- Verificare una traduzione controllando se ogni carattere simbolo è conservato.

Ci sono alcune occasioni in cui potresti non voler mai tradurre una stringa. In Gambas, è possibile indicare che una stringa non deve essere tradotta inserendo un segno meno come stringa tradotta. La tua traduzione è archiviata in un file *.po file nella directory .lang del progetto. Il nome del file .po dipende dalla lingua tradotta. Ad esempio, il file di traduzione francese si chiama fr.po.

Ogni volta che l'ultima versione di Gambas cambia, qualsiasi stringa non tradotta nel programma potrebbe cambiare. Tuttavia, tutte le stringhe tradotte verranno conservative. Se una stringa non tradotta scompare completamente, anche la traduzione scomparirà e se viene aggiunta una nuova stringa non tradotta, le viene assegnata una stringa di traduzione vuota o nulla. Tutte le altre stringhe non tradotte manterranno le corrispondenti stringhe di traduzione. Sebbene la funzione di traduzione delle stringhe di Gambas ti consenta di mantenere facilmente le conversioni di stringhe, altre cose specifiche della locale come date, formati di numeri, ecc. Sono lasciate a te da gestire. Si spera che Gambas2 affronti queste funzionalità quando verrà rilasciato. Fino a quel momento, ecco una breve lista di controllo generalizzata per aiutarti a localizzare la tua applicazione:

- Assicurati che tutti i messaggi, le icone e il contenuto leggibile dall'uomo siano archiviati in file di risorse esterne e possano essere facilmente tradotti.
- Il messaggio o i file di risorse tradotti devono essere caricati dinamicamente dall'applicazione, a seconda della lingua corrente e delle impostazioni locali per la sessione.

Una guida per principianti a

- Data / ora, ordinamento, formattazione numerica e monetaria ecc. Sono nella lingua di destinazione. L'ordinamento dovrebbe essere configurabile, a seconda di

Una guida per principianti a la lingua dell'utente.

- I caratteri della lingua di destinazione possono essere inseriti e visualizzati correttamente dall'utente e possono essere letti e scritti nel file system nativo della piattaforma di destinazione.

Secondo Huang et. al., 25 dovresti prendere in considerazione la creazione di una checklist di test I18N per la tua applicazione. Alcune delle domande che ti consigliano di considerare includono le seguenti:

1. La progettazione del programma tiene conto di considerazioni internazionali?
2. Un utente può inserire testo, acceleratori e combinazioni di tasti di scelta rapida su una tastiera internazionale?
3. Un utente può inserire date, orari, valute e numeri utilizzando formati internazionali?
4. Un utente può tagliare / incollare con successo il testo con caratteri accentati (o doppi byte)?
5. L'applicazione può funzionare correttamente su diversi tipi di hardware venduti nel mercato di destinazione?
6. L'applicazione funziona correttamente su sistemi operativi localizzati?
7. Un utente può digitare caratteri europei accentati (o doppi byte asiatici) nei documenti e nelle finestre di dialogo?
8. I documenti creati in una lingua di destinazione possono essere aperti in altre lingue e viceversa?
9. Gli utenti possono salvare e stampare file con caratteri accentati (o doppi byte)?

Sebbene non esista un unico elenco di controllo che possa spiegare l'ampio grado di variazione da applicazione a applicazione, la considerazione di questi problemi di base probabilmente contribuirà notevolmente a raggiungere un pubblico globale. Internet ha reso il mondo un posto molto più piccolo e il pubblico per le applicazioni è diventato molto

This product is © 2005 by John W. Rittinghouse, all rights are reserved. It is released to the
Gnattus User Community under an Open Content Licence (OCL) and may not be distributed
unmodified or otherwise without the express written consent of the author.

Una guida per principianti a

- 25 Huang, E., Haft, R. e Hsu, J. "Developing a Roadmap for Software Internationalization", White paper datato ottobre 2000, recuperato daURL www.symbio-group.com/doc/Developing una tabella di marcia per l'internazionalizzazione del software.pdf su 5 ottobre 2005.

Una guida per principianti a

più grandi. Non puoi semplicemente presumere che chiunque utilizzi la tua applicazione parli inglese e utilizzi la stessa tastiera o capirà le tue icone. Pittogrammi diversi significano cose diverse a seconda della località in cui risiede l'utente. Soprattutto, fai lo sforzo di soddisfare il pubblico più ampio possibile e molto probabilmente i tuoi utenti consentiranno alcuni errori qua e là. Non fare uno sforzo del genere generalmente lascerà loro l'impressione che potrebbe importarti di meno e probabilmente si sentiranno allo stesso modo riguardo all'uso del tuo programma.

Copyright (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the JWS User Community under an Open Content License (OCL) and may not be distributed further terms or conditions without the express written consent of the author.

Una guida per principianti a

Questa pagina è intenzionalmente
vuota.

This product is (C) 2005 by John W. Rittinghouse, all rights are reserved. It is released to the Gambar User Community under an Open Content License (OCL) and may not be distributed under any other terms or conditions without the express written consent of the author.

Una guida per principianti a

Indice alfabetico

| | |
|--|-------------------------------|
| Funzione di accesso | (188) |
| Conforme ad ACID..... | (331) |
| Attiva evento | (155, 162, 169p.) |
| Aggiungi metodo | (89, 152, 154, 159, 167, 176) |
| Operatore AND | (41) |
| Parola chiave APPEND | (192) |
| Percorso dell'applicazione | (327) |
| Operatori aritmetici..... | (36) |
| Frase di assegnazione | (35) |
| Atomicità | (331) |
| Proprietà BackColor | (62, 67, 268) |
| Proprietà dello sfondo | (62) |
| Benoît Minisini | (1, 19, 149, 157) |
| BigEndian | (192) |
| Funzioni bit | (319) |
| Bit..... | (318) |
| Tipi di dati booleani..... | (31) |
| Proprietà di confine | (63) |
| Costanti incorporate | (34) |
| Proprietà occupata | (151p.) |
| Byte..... | (318) |
| Tipo di dati byte | (31p.) |
| Proprietà memorizzata nella cache | (267) |
| Pulsante Annulla | (61) |
| Annulla proprietà | (63) |
| Proprietà didascalia..... | (64, 81) |
| Porta un po ' | (323) |
| CASE ELSE blocco | (49) |
| Dichiarazione CASE..... | (48, 103, 114) |
| Dichiarazione CASE..... | (48) |
| Dichiarazione CATCH..... | (305) |
| Classe CheckBox..... | (93) |
| Controllo CheckBox | (92 pagine) |
| Proprietà controllata | (100p., 157) |
| Chr (10) | (166) |
| Chr (13) | (199) |
| Chr (9) | (136, 194) |
| Evento di classe | (310) |
| Gerarchia di classe | (28) |
| Invariante di classe..... | (241) |
| Appuccio OOP basato sulla classe | (241) |
| Fare clic su evento | (76) |
| Fare clic su NRun® | (22) |
| Classe di clip | (268) |
| Area di ritaglio | (268) |
| Coesione | (241) |
| Collezioni | (55) |
| Classe di colore | (268) |
| Controllo ColumnView | (170, 174, 177) |
| Matrici ComboBox | (89) |
| Controllo ComboBox | (85pp., 186) |
| Elementi ComboBox | (89) |
| Interfaccia a riga di comando | (326) |
| Dichiarazione COMMIT | (331) |
| Operatori di confronto | (36) |
| Condizionali | (46) |
| Classe di connessione | (327, 332) |
| Oggetto connessione | (327) |
| Consistenza..... | (332) |
| Costante | (34) |
| Routine del costruttore | (150, 210, 213, 248p.) |
| Contentore classe | (62) |
| Classe di controllo | (61) |
| Gruppi di controllo | (100) |
| Controlli | (58) |
| Funzione COPIA | (195) |
| Avviso sul copyright | (3) |
| Avviso sul copyright | (2p.) |
| Proprietà cursore | (65) |
| Astrazione dei dati | (239) |
| Proprietà database | (328) |
| Tipo di dati data | (32) |
| Evento doppio clic | (78) |
| Operatore DEC | (43) |
| Pulsante predefinito | (61) |
| Proprietà predefinita | (66) |
| Elimina metodo | (70) |
| Proprietà di design | (66) |
| Desktop.Charset | (142, 152) |
| Classe di dialogo | (103) |
| Proprietà del titolo della finestra di dialogo | (103) |
| Funzione dir | (189) |
| Parola chiave DIRETTA | (192) |
| Divisione per zero | (39p.) |
| Divisione per zero | (40) |
| FARE [MENTRE] LOOP | (51) |
| Notazione del punto | (36) |
| Evento DoubleClick | (77) |
| Metodo di trascinamento | (70) |
| Disegna classe | (267) |
| Draw.Ellipse metodo | (276) |
| Draw.FillStyle | (279) |
| Metodo Draw.Line | (276) |
| Draw.LineWidth proprietà | (276) |
| Metodo Draw.Point | (274) |
| Metodo Draw.Polygon | (280p.) |
| Metodo Draw.Polyline | (280, 282) |
| Metodo Draw.Rect | (274, 279) |
| Controllo DrawingArea | (271) |
| Elimina proprietà | (66) |
| Durevolezza | (332) |
| Proprietà abilità | (66) |
| Incapsulamento | (240) |
| Parola chiave END | (37) |
| FINE CON istruzione | (35) |
| Inserisci evento | (74) |
| Funzione Eof | (190) |
| Classe di errore | (304) |
| Gestione degli errori | (298) |
| Proprietà Error.Class | (304) |
| Error.Clear | (305) |
| Proprietà Error.Code | (304) |
| Error.Raise | (305) |
| Proprietà Error.Text | (304) |
| Errore, dove proprietà | (305) |

Una guida per principianti a

| | | | |
|--|---------------|---------------------------------------|---------------------------------------|
| Errori di logica | (301) | HTML | (70) |
| Linguaggio guidato dagli eventi | (46) | HTML..... | (26, 81p., 84p.) |
| Gestore di eventi | (77) | Notazione ungherese | (31) |
| Programmazione basata su eventi | (309) | Editor di icone | (28) |
| Sistema basato sugli eventi | (309) | Controllo IconView | (149, 154 pagine, 158) |
| Metodo Exec | (327) | Dichiarazione IF | (47, 101, 161, 168) |
| Esiste la funzione | (190) | Istruzione IF | (47) |
| Metodo esistente..... | (167) | SE ALLORA..... | (110) |
| Espandi proprietà | (67) | Classe dell'immagine..... | (284p.) |
| File System Safe UCS Transformation Format (FSS / UTF) | (353) | Operatore INC..... | (43) |
| Riempì la classe..... | (269) | Ciclo infinito | (51) |
| Proprietà FillColor | (268) | Eredità | (240) |
| Costanti FillStyle | (269) | Inkscape..... | (292) |
| Proprietà FillStyle..... | (269) | Tipo di dati intero | (32) |
| Proprietà FillX..... | (269) | Internazionalizzazione | (22, 351) |
| Proprietà FillY..... | (269) | Funzioni intrinseche e derivate | (215) |
| Infine..... | (306) | Inverti proprietà | (269, 275) |
| Tipo di dati Float..... | (32) | Funzione IsDir | (191) |
| Funzione FLUSH..... | (195) | ISO / IEC 10646 | (352) |
| Controllo della messa a fuoco | (72) | Solitudine..... | (332) |
| Classe di caratteri | (108pp., 269) | Giava | (19, 53, 237) |
| Proprietà del carattere | (67) | Javalike array..... | (53) |
| PER CIASCUNO | (50) | Classe chiave..... | (316) |
| PER OGNI costrutto..... | (55) | Funzione KILL | (196) |
| PER dichiarazione | (49p., 55) | ETICHETTA..... | (49) |
| Proprietà ForeColor..... | (67) | Controlli del layout | (177p.) |
| Primo piano proprietà | (67) | Lascia l'evento | (74) |
| Classe di frame | (92) | Proprietà di sinistra | (68) |
| Editor di codice Gambas | (28) | PIACE | (43, 45, 189) |
| Ambiente di codifica Gambas | (29) | Proprietà LineStyle | (270, 276) |
| Libreria dei componenti Gambas..... | (91) | Linspire®..... | (22) |
| Costanti di Gambas | (34) | Linux | (19, 142, 178, 196p., 238, 266) |
| Costanti di Gambas | (34) | Editor di elenchi | (186) |
| Tipi di dati Gambas | (30, 33) | Controllo ListBox..... | (89p.) |
| Ambiente di sviluppo di Gambas | (66) | Controllo ListView..... | (158 pagine) |
| Editor di icone di Gambas..... | (162, 198) | LittleEndian | (192) |
| Gambas IDE | (27, 58) | Condizioni di carico | (302) |
| Interprete Gambas | (19, 21, 266) | Localizzazione | (138 pagine, 142 pagine, 145, 351) |
| Interprete GAMBAS | (141) | Funzione Lof () | (195) |
| Parole chiave di Gambas | (46) | Proprietà di accesso | (329) |
| Gambas mascotte | (38) | Strutture ad anello | (49) |
| Gambas ToolBox | (57) | Metodo inferiore | (71) |
| Gambas Wiki | (20, 165) | Editor di menu..... | (98p., 197) |
| Componente Gb.db..... | (327) | Voci del menu | (99) |
| Licenza pubblica GNU | (19) | Classe messaggio | (112) |
| Istruzione GOTO..... | (49) | Controllo MessageBox | (112) |
| Metodo di afferrare | (71) | Microsoft Visual Basic® | (19) |
| Classi dell'interfaccia utente grafica | (58) | TIPO MIME..... | (70) |
| GridCell..... | (171) | Funzione MKDIR | (196) |
| Matrice GridView | (171) | Funzione MOD | (40) |
| Controllo GridView | (171p.) | Operatore MOD | (40) |
| GTK + | (21) | Classe del mouse | (312) |
| Proprietà H | (68) | Proprietà del mouse | (68) |
| Gestire la proprietà | (68) | Metodo di spostamento | (71) |
| Maniglie | (61) | Metodo MoveCurrent..... | (161, 165p., 168) |
| Proprietà altezza..... | (68, 272) | MySQL..... | (326) |
| Nascondere metodo | (71) | Manuale MySQL | (326) |
| Proprietà dell'ospite | (329) | Server MySQL | (326) |
| Funzione HSV | (63) | Proprietà nome | (329) |

Una guida per principianti a

Matrice nativa(53)

Una guida per principianti a

| | | | |
|--|---------------------------|---|---------------------------------|
| NUOVA parola chiave..... | (30) | Proprietà ScreenY | (69) |
| Nuovo progetto guidato | (37, 72) | Controllo ScrollView | (293p.) |
| Proprietà successiva | (69) | Istruzione SELECT | (48, 105, 114) |
| Dichiarazione NEXT | (50) | Finestra di dialogo Seleziona colore | (103, 105) |
| Dichiarazione NEXT | (49) | Finestra di dialogo Seleziona directory | (104, 119) |
| NON operatore | (44) | Finestra di dialogo Seleziona carattere..... | (108 pagine) |
| Tipo di dati oggetto..... | (33) | Strumento di selezione | (28) |
| Programmazione orientata agli oggetti..... | (237) | Metodo SetFocus | (72) |
| Dati del tipo di oggetto | (238) | Metodo SetFocus () | (69) |
| Concetti OO..... | (238) | Tipo di dati breve | (32) |
| Paradigma OO | (237) | Mostra metodo | (71) |
| OOP | (237) | Istruzione SQL WHERE | (333) |
| OOPL..... | (238) | SQLite..... | (326) |
| Licenza OpenContent..... | (2p.) | Esegibile autonomo | (20) |
| Finestra di dialogo OpenFile | (104, 117p.) | Funzione stat | (150, 153, 191) |
| Operatore OR | (41, 188) | Parola chiave STATICa | (30) |
| Classe del pannello | (95) | Barra di stato..... | (26) |
| Pannello di controllo | (95, 177, 185) | Parola chiave STEP..... | (50) |
| Errori di passaggio dei parametri | (301) | Costanti stringa | (34p.) |
| Proprietà genitore | (69) | Tipo di dati stringa..... | (32, 34, 55, 126) |
| Parola d'ordine proprietà | (330) | Funzioni di stringa | (125) |
| Immagine classe | (69, 287) | Operatori di stringa | (36) |
| Controllo dell'immagine | (78) | Sottomenu | (99) |
| Oggetto immagine | (65, 79) | SVG | (292) |
| Proprietà dell'immagine | (69) | Specifiche SVG | (292) |
| File PNG | (79) | Classe di sistema | (91, 150) |
| Polimorfismo..... | (240) | Proprietà Tables | (330) |
| Proprietà portuale | (330) | Controllo TableView | (348) |
| PostgreSQL..... | (326) | Controllo TabStrip | (182p.) |
| Operatore di potenza | (41) | Proprietà tag | (69) |
| Costanti predefinite | (62, 67, 103, 148) | Funzione temp \$ | (192) |
| Proprietà precedente | (69) | Applicazione terminale | (37, 125, 232) |
| Dichiarazione PRINT | (38, 46, 51) | Proprietà di testo | (64, 69) |
| Istruzione PRINT | (46) | Controllo della casella di testo | (83, 166) |
| Parola chiave PRIVATA..... | (30) | Controllo TextBox | (82p., 252) |
| Controllo ProgressBar | (73) | Proprietà TextHeight | (272) |
| Project Explorer | (25 pagine, 37 pagine) | Proprietà TextWidth | (272) |
| Menu progetto | (27) | Evento timer | (187) |
| Parola chiave PUBLIC | (30, 34) | Controllo ToggleButton | (92p.) |
| Qt..... | (21, 58, 142) | ToolBar | (28) |
| QT..... | (58, 60) | ToolBox | (27) |
| Libreria QT | (347) | Finestra ToolBox | (27p., 60) |
| Comando QUIT | (37) | ToolTip | (28) |
| Metodo di citazione | (335) | Proprietà ToolTip | (69) |
| Condizione di gara | (301) | Proprietà superiore | (70) |
| RadioButton classe..... | (95) | Transazione | (331) |
| Controllo RadioButton..... | (95) | Proprietà trasparente | (270) |
| Metodo di sollevamento | (71) | Visualizzazione ad albero | (105) |
| Errore recuperabile | (298) | Visualizzazione ad albero | (25, 28, 37, 57 pagg., 64, 163) |
| Metodo di aggiornamento | (72) | Visualizzazione ad albero controllo | (163 pagine, 167 pagine) |
| Rimuovi metodo | (168) | Trolltech | (347) |
| REPEAT e FINO A | (53) | PROVARE | (304) |
| Metodo di ridimensionamento | (72) | Dichiarazione di PROVA | (305) |
| Oggetto risultato | (333, 335) | Tipo di proprietà | (330) |
| Comando RETURN | (37) | UCS2 | (353) |
| Funzione RGB | (63) | UCS4 | (353) |
| Dichiarazione ROLLBACK..... | (331) | Standard Unicode | (352) |
| Finestra di dialogo Salva file | (104, 118) | Proprietà degli utenti | (330) |
| Grafica vettoriale scalabile | (292) | UTF8 | (353) |
| Proprietà ScreenX | (69) | Set di caratteri UTF8 | (135, 141) |

Una guida per principianti a

| | | | |
|--------------------------------|------------------|---|----------------|
| Errore di convalida | (298) | MENTRE ... ANDARE | (52) |
| Proprietà di valore | (70) | Proprietà di larghezza | (68, 272, 276) |
| Tipo di dati variante | (33, 55) | Wikipedia..... | (215) |
| VB..... | (19, 32, 37, 59) | Proprietà finestra..... | (70) |
| Proprietà della versione | (330) | CON | (35) |
| Classe virtuale .DrawClip..... | (268) | CON la parola chiave | (35) |
| Spedizione virtuale | (242) | X proprietà..... | (68) |
| Proprietà visibile | (70) | XWindows..... | (78) |
| Proprietà W | (68) | X / Open Joint Internationalization Group | (353) |
| ASPETTARE..... | (71) | Operatore XOR | (42) |
| Messaggi di avviso | (115) | Y proprietà..... | (68) |
| Parola chiave WATCH..... | (192) | | |

© 2005 by John W. Rittinghouse, all rights are reserved. It is released to the public under any other terms or conditions without the express written consent of the author.