**Dokumentation des Erdkundelernprogramms**

Treffen mit dem Arbeitgeber (15.08.13):

In einer großen Pause haben wir uns das erste Mal mit dem Mathematik- und Erdkundelehrer Herr Müller, welcher ein interessierter Abnehmer unserer Projektidee war, getroffen. Wir berichteten von unserer, später im Lastenheft angeführten, Idee, des Erdkundelernprogramms und schilderten ihm unsere Vorstellungen. Er schien sofort begeistert zu sein und uns war klar, dass mit seiner Hilfe die Umsetzung gut funktionieren wird. Jedoch verlangte er, dass man sich nur auf die Orientierungsstufe (5./6.) fokussieren sollte, da hier die wesentlichen geographischen Anforderungen unseres geplanten Programms im Unterricht Thema wären. Viel Wert legte er auf die Spiele, wo man Städte, Flüsse, Gebirge…etc. in Karten einzeichnen muss, da diese im Praktischen schwer zu lernen sind. Insgesamt kann man also sagen, dass wir bei diesem Treffen grobe inhaltliche Anforderungen von Herrn Müller bekommen haben, die kreative Umsetzung jedoch noch uns überlassen wurde. Am Ende dieser Übereinkunft gab er uns noch den gesamten Lehrplan der 5. und 6. Klasse des Otto-Hahn-Gymnasiums und viele Bücher und Zettel mit den von uns benötigten Informationen.

Treffen der Gruppe, um eine genaue Planung zu erstellen, wann was fertig sein muss und wann, wer was macht(30.8.13)

Wir haben von unserem Nebenauftraggeber Herrn Wienberg die Aufgabe bekommen, vor dem richtigen Beginn des Projektes eine genaue Planung zu machen, wann wir die einzelnen Bestandteile fertig haben müssen und wann wir mit Anderen beginnen. Dazu haben wir eine Grafik erstellt und diese dann am 05.9.13 abgeschickt.

Treffen der Gruppe, um sich in GitHup einzuarbeiten ( 17.09.13):

Nebeninfo: GitHup ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte. Er verwendet namensgebenderweise das Versionsverwaltungs-System Git. (wikipedia.de)

An dem Nachmittag hat sich unsere Gruppe getroffen, um zu besprechen wie GitHup funktioniert. Dabei haben Olga und Arne uns Anderen einen Einführungskurs gegeben, da sie sich bereits in das Programm eingearbeitet hatten. Es wurden Dinge geklärt, die die Veränderung von Daten sowie das richtige Speichern von Daten mit diesem Programm betreffen. Dabei haben wir festgestellt, dass es gewisse Risiken bezüglich der Sicherheit gibt. Denn wenn zwei Leute gleichzeitig etwas ändern und speichern, dann entsteht im Programm ein Fehler, der die veränderten Dateien unzugänglich macht. Deshalb haben Niklas und Dario entschieden, dass sie das Programm für die Erstellung der Dokumentation nicht verwenden werden, um unnötige Risiken zu vermeiden. Anstatt dessen werden sie die Texte gemeinsam erstellen oder die Texte einzeln erstellen und später vergleichen bzw. zusammenfügen. Arne, Olga, Steffen, Artur haben festgestellt, dass sie sich aufgrund dieses Fehlers absprechen müssen, wenn sie Dinge ändern und speichern wollen.

Arbeit während der Unterrichtseinheit am (30.09.13)

Arne und Steffen arbeiten an der groben Menüstruktur, Farben, Positionen und vieles mehr wird ausgewählt. Olga arbeitet an unserem Maskottchen, sodass es weitere Mimiken erhält.

Artur macht Notizen für seine Teilprogramme. Das Minispiel, in dem Kontinente etc. benannt werden müssen.

Dario und Niklas arbeiten an der Dokumentation und geben Olga Vorschläge für neue Mimiken, die in das Programm mit eingeführt werden können.

Besprechung der Funktionen der Lehrerkonsole.

Programm:

Struktur.pas:

|  |  |  |
| --- | --- | --- |
|  |  | + |
|  |  | +interface |
|  |  | + |
|  |  | +uses |
|  |  | + Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, |
|  |  | + StdCtrls; |
|  |  | + |
|  |  | +type |
|  |  | + TMenue = class(TForm) |
|  |  | + procedure FormPaint(Sender: TObject); |
|  |  | + procedure FormCreate(Sender: TObject); |
|  |  | + procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  |  | + Y: Integer); |
|  |  | + procedure FormMouseDown(Sender: TObject; Button: TMouseButton; |
|  |  | + Shift: TShiftState; X, Y: Integer); |
|  |  | + procedure FormMouseUp(Sender: TObject; Button: TMouseButton; |
|  |  | + Shift: TShiftState; X, Y: Integer); |
|  |  | + private |
|  |  | + { Private-Deklarationen } |
|  |  | + public |
|  |  | + { Public-Deklarationen } |
|  |  | + end; |
|  |  | + |
|  |  | +var |
|  |  | + Menue: TMenue; |
|  |  | + Themenfarbe1: TColor; |
|  |  | + Themenfarbe2: TColor; |
|  |  | + |
|  |  | +implementation |
|  |  | + |
|  |  | +{$R \*.DFM} |
|  |  | + |
|  |  | +procedure TMenue.FormPaint(Sender: TObject); |
|  |  | +begin |
|  |  | + Menue.Color := Themenfarbe1; |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TMenue.FormCreate(Sender: TObject); |
|  |  | +begin |
|  |  | + Themenfarbe1 := RGB(233,169,58); //Themenfarben können sich durchs ganze |
|  |  | + Themenfarbe2 := RGB(107,142,35); //Programm ziehen... (sind noch nicht beschlossen) |
|  |  | + self.DoubleBuffered := true; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure TMenue.FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  |  | + Y: Integer); |
|  |  | +var dif:integer; |
|  |  | +begin |
|  |  | + if Y < 50 then Cursor := crHandpoint // Wenn sich die Maus im oberen Bildschirmbereich |
|  |  | + else Cursor := crDefault; // befindet, bekommt sie ein Handsymbol |
|  |  | + |
|  |  | + if Menue.Align = alNone then // Wenn das Menü-Fenster im verschiebbaren Modus ist, |
|  |  | + begin |
|  |  | + dif := Mouse.CursorPos.y -Y; |
|  |  | + Menue.Top := Y+dif; // verschiebt sich das Fenster mit der Maus in y-Richtung. |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure TMenue.FormMouseDown(Sender: TObject; Button: TMouseButton; |
|  |  | + Shift: TShiftState; X, Y: Integer); |
|  |  | +begin |
|  |  | + if Button = mbleft then // Wenn die linke Maustaste gedrückt wird |
|  |  | + begin |
|  |  | + if Cursor = crHandpoint then // und sich die Maus im oberen Bildschirmberech befindet (s.o.), |
|  |  | + begin // wird aus dem Vollbild "Normalbild" (ist aber nicht sichtbar, |
|  |  | + end; // da dich Höhe und Breite des Fenstern NICHT ändert! |
|  |  | + Menue.Align := alNone; // Dies muss aber geschehen, damit man das Fenster im |
|  |  | + end; // nächsten Schritt verschieben kann. |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TMenue.FormMouseUp(Sender: TObject; Button: TMouseButton; |
|  |  | + Shift: TShiftState; X, Y: Integer); |
|  |  | +var i: byte; |
|  |  | +begin |
|  |  | + if Button = mbleft then // linke Taste wird gelöst: |
|  |  | + begin |
|  |  | + if Menue.Top > Screen.Height div 2 then // Wenn das Fenster bis unter die Hälfte |
|  |  | + begin // der Bildschirmhöhe gezogen wurde, |
|  |  | + for i := Y to Screen.Height - 100 do // wird das Menü bis zum Verschwinden |
|  |  | + begin |
|  |  | + Menue.Top := Menue.Top + i; // weiter nach unten verschoben |
|  |  | + sleep(10); // (nicht sofort, sondern "langsam") |
|  |  | + end; |
|  |  | + close; // und schließlich geschlossen. |
|  |  | + end else // Wird das Fenster nicht genügend weit |
|  |  | + begin // herunter gezogen, |
|  |  | + for i := Mouse.CursorPos.y downto 0 do |
|  |  | + begin |
|  |  | + Menue.Top := i; // gelangt das Fenster wieder "langsam" in |
|  |  | + sleep(1); // die Ausgangsposition und zum Schluss wieder |
|  |  | + end; |
|  |  | + Menue.Align := alClient; // in den unverschiebbaren Vollbildmodus |
|  |  | + end; |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | +end. |

Arbeit an der farblichen Hervorhebung des oberen Aktivierungsbereiches (30.9.13)

Arne hat hier an der farblichen Hervorhebung des Aktivierungsbereiches gearbeitet.

|  |  |  |
| --- | --- | --- |
| procedure TMenue.FormCreate(Sender: TObject); |  |  |
|  |  | begin |
|  |  | - Themenfarbe1 := RGB(233,169,58); //Themenfarben können sich durchs ganze |
|  |  | - Themenfarbe2 := RGB(107,142,35); //Programm ziehen... (sind noch nicht beschlossen) |
|  |  | + Themenfarbe1 := RGB(244,164,96); //Themenfarben können sich durchs ganze |
|  |  | + Themenfarbe2 := RGB(205,133,63); //Programm ziehen... (sind noch nicht beschlossen) |
|  |  | self.DoubleBuffered := true; |
|  |  | end; |
|  |  |  |
|  | | @@ -48,8 +48,20 @@ procedure TMenue.FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  |  | Y: Integer); |
|  |  | var dif:integer; |
|  |  | begin |
|  |  | - if Y < 50 then Cursor := crHandpoint // Wenn sich die Maus im oberen Bildschirmbereich |
|  |  | - else Cursor := crDefault; // befindet, bekommt sie ein Handsymbol |
|  |  | + if Y < 50 then // Wenn sich die Maus im oberen Bildschirmbereich |
|  |  | + begin |
|  |  | + Cursor := crHandpoint; // befindet, bekommt sie ein Handsymbol und |
|  |  | + Canvas.Brush.Color := Themenfarbe2; |
|  |  | + Canvas.Pen.Color := Themenfarbe2; // mit Canvas wird der obere Bildschirmbereich |
|  |  | + Canvas.Rectangle(0,0,ClientWidth,50); // in der 2. Themenfarbe gefärbt. |
|  |  | + end else |
|  |  | + begin // andererseits wird, |
|  |  | + if Cursor = crHandpoint then // nur wenn es nicht schon der Fall ist, |
|  |  | + begin |
|  |  | + Cursor := crDefault; // der Maus der Normale Zeiger zugeordnet |
|  |  | + refresh; // und der andersfarbige Bereich wieder gelöscht. |
|  |  | + end; |
|  |  | + end; |
|  |  |  |
|  |  | if Menue.Align = alNone then // Wenn das Menü-Fenster im verschiebbaren Modus ist, |
|  |  | begin |
|  |  | dif := Mouse.CursorPos.y -Y; |
|  |  | Menue.Top := Y+dif; // verschiebt sich das Fenster mit der Maus in y-Richtung. |
|  |  | end; |
|  |  | end; |
|  |  |  |
|  |  |  |
|  |  | procedure TMenue.FormMouseDown(Sender: TObject; Button: TMouseButton; |
|  |  | Shift: TShiftState; X, Y: Integer); |
|  |  | begin |
|  |  | if Button = mbleft then // Wenn die linke Maustaste gedrückt wird |
|  |  | begin |
|  |  | if Cursor = crHandpoint then // und sich die Maus im oberen Bildschirmbereich befindet (s.o.), |
|  |  | begin // wird aus dem Vollbild "Normalbild" (ist aber nicht sichtbar, |
|  |  | end; // da sich Höhe und Breite des Fenstern NICHT �ändert! |
|  |  | Menue.Align := alNone; // Dies muss aber geschehen, damit man das Fenster im |
|  |  | end; // nächsten Schritt verschieben kann. |
|  |  | end; |
|  |  |  |
|  |  | procedure TMenue.FormMouseUp(Sender: TObject; Button: TMouseButton; |
|  |  | Shift: TShiftState; X, Y: Integer); |

Arne arbeitet an der Menüaufteilung (03.10.13)

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | procedure TMenue.FormPaint(Sender: TObject); |
|  |  | +var i : integer; |
|  |  | + x,y : integer; |
|  |  | + ScreenMitte: TPoint; |
|  |  | + Anzahl:integer; |
|  |  | + Radius\_x,Radius\_y : integer; |
|  |  | + Buttonbreite : integer; |
|  |  | begin |
|  |  | Menue.Color := Themenfarbe1; |
|  |  | + |
|  |  | + ButtonBreite := Screen.Width div 6; // Buttonbreite wird in Abhängigkeit der |
|  |  | + Image1.Width := ButtonBreite; // Bildschrimgröße bestimmt |
|  |  | + Image1.Height := ButtonBreite; |
|  |  | + Image2.Width := ButtonBreite; // provisorisch habe ich Images als Platzhalter |
|  |  | + Image2.Height := ButtonBreite; // für die späteren Butten genommen... |
|  |  | + Image3.Width := ButtonBreite; // sie könnten Bilder enthälten, Zeichen, nur Text |
|  |  | + Image3.Height := ButtonBreite; // Form steht auch nicht fest... |
|  |  | + Image4.Width := ButtonBreite; // Das Alles wird sich alles im Laufe der |
|  |  | + Image4.Height := ButtonBreite; // Programmentwicklung noch ergeben. |
|  |  | + Image5.Width := ButtonBreite; |
|  |  | + Image5.Height := ButtonBreite; |
|  |  | + // im Folgenden werden die Menüpunkte im Kreis(Ellipse) angeordnet: |
|  |  | + ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | + Anzahl := 5; // Anzahl der Menüobjekte |
|  |  | + Radius\_x := Screen.Width div 3; // Radius in x-Richtung |
|  |  | + Radius\_y := Screen.Height div 3; // Radius in y-Richtung |
|  |  | + for i := 1 to Anzahl do |
|  |  | + begin |
|  |  | + x := Kreisposition\_x(i,Anzahl,ScreenMitte,Radius\_x); // x- und y-Koordinate für das i-te Objekt wird ermittelt |
|  |  | + y := Kreisposition\_y(i,Anzahl,ScreenMitte,Radius\_y); // dabei werden oben bestimmte Parameter übergeben |
|  |  | + case i of |
|  |  | + 1: begin |
|  |  | + Image1.Left := x - ButtonBreite div 2; // den Menüobjekten werden ihre Koordinaten zugeordnet |
|  |  | + Image1.Top := y - ButtonBreite div 2; |
|  |  | + end; |
|  |  | + 2: begin |
|  |  | + Image2.Left := x - ButtonBreite div 2; |
|  |  | + Image2.Top := y - ButtonBreite div 2; |
|  |  | + end; |
|  |  | + 3: begin |
|  |  | + Image3.Left := x - ButtonBreite div 2; |
|  |  | + Image3.Top := y - ButtonBreite div 2; |
|  |  | + end; |
|  |  | + 4: begin |
|  |  | + Image4.Left := x - ButtonBreite div 2; |
|  |  | + Image4.Top := y - ButtonBreite div 2; |
|  |  | + end; |
|  |  | + 5: begin |
|  |  | + Image5.Left := x - ButtonBreite div 2; |
|  |  | + Image5.Top := y - ButtonBreite div 2; |
|  |  | + end; |
|  |  | + end; |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | +function TMenue.Kreisposition\_x(Objektnummer:integer;Objektanzahl:integer;Zentrum:TPoint; |
|  |  | + Radius:integer) : integer; // Parameterübergabe |
|  |  | +var |
|  |  | + RadWinkel : real; |
|  |  | + x : integer; |
|  |  | +begin |
|  |  | + RadWinkel := Objektnummer\*((2\*pi)/Objektanzahl); // Winkel des aktuellen Objektes wird errechnet |
|  |  | + x := round(Zentrum.x + cos(RadWinkel+(pi/2))\*Radius); // daraus wird die x-Koordinate des Objektes ermittelt |
|  |  | + result := x; |
|  |  | +end; |
|  |  | + |
|  |  | +function TMenue.Kreisposition\_y(Objektnummer:integer;Objektanzahl:integer;Zentrum:TPoint; |
|  |  | + Radius:integer) : integer; // Parameterübergabe |
|  |  | +var |
|  |  | + RadWinkel : real; |
|  |  | + y : integer; |
|  |  | +begin |
|  |  | + RadWinkel := Objektnummer\*((2\*pi)/Objektanzahl); // Winkel des aktuellen Objektes wird errechnet |
|  |  | + y := round(Zentrum.y - sin(RadWinkel+(pi/2))\*Radius); // daraus wird die y-Koordinate des Objektes ermittelt |
|  |  | + result := y; |
|  |  | end; |
|  |  |  |

Entwicklung der Verschlüsselung der Accountdaten (04.10.13)

Steffen hat die Verschlüsselung der Accountdaten entwickelt, damit es keine Datenschutzverletzung geben kann. Dadurch werden die Daten zwar in Textdateien gespeichert, sind jedoch für niemanden einsehbar, sodass Datenschutz gewährleistet ist.

Arne arbeitet weiter am Menü (07.10.13)

Diesmal arbeitet Arne daran, dass das Menü aus dem Bildschirmmittelpunkt erscheint.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | + MenuePos:integer = 1; |
|  |  | + ScreenMitte:TPoint; |
|  |  |  |
|  |  | implementation |
|  |  |  |
|  |  | {$R \*.DFM} |
|  |  |  |
|  |  | procedure TMenue.FormPaint(Sender: TObject); |
|  |  | +begin |
|  |  | + Menue.Color := Themenfarbe1; |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TMenue.FormCreate(Sender: TObject); |
|  |  | +begin |
|  |  | + Themenfarbe1 := RGB(244,164,96); //Themenfarben können sich durchs ganze |
|  |  | + Themenfarbe2 := RGB(205,133,63); //Programm ziehen... (sind noch nicht beschlossen) |
|  |  | + self.DoubleBuffered := true; |
|  |  | + |
|  |  | + ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | + |
|  |  | +procedure TMenue.MenueEffektTimer(Sender: TObject); |
|  |  | +begin |
|  |  | + inc(MenuePos,3); // Timer lässt das Menue aus der Bildschirmmitte erscheinen |
|  |  | + MenuePosition(MenuePos); |
|  |  | + if MenuePos >= Screen.Height div 3 then MenueEffekt.Enabled := false; |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TMenue.MenuePosition(Radius:integer); |
|  |  | var i : integer; |
|  |  | x,y : integer; |
|  |  | - ScreenMitte: TPoint; |
|  |  | - Anzahl:integer; |
|  |  | - Radius\_x,Radius\_y : integer; |
|  |  | + Radius\_x,Radius\_y : real; |
|  |  | Buttonbreite : integer; |
|  |  | + Anzahl:integer; |
|  |  | begin |
|  |  |  |
|  |  | - |
|  |  | - ButtonBreite := Screen.Width div 6; // Buttonbreite wird in Abhängigkeit der |
|  |  | + ButtonBreite := (3\*Radius)div 4; // Buttonbreite wird in Abhängigkeit der |
|  |  | Image1.Width := ButtonBreite; // Bildschrimgröße bestimmt |
|  |  | Image1.Height := ButtonBreite; |
|  |  | Image2.Width := ButtonBreite; // provisorisch habe ich Images als Platzhalter |
|  | | @@ -62,10 +87,9 @@ procedure TMenue.FormPaint(Sender: TObject); |
|  |  | Image5.Width := ButtonBreite; |
|  |  | Image5.Height := ButtonBreite; |
|  |  | // im Folgenden werden die Menüpunkte im Kreis(Ellipse) angeordnet: |
|  |  | - ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | - Anzahl := 5; // Anzahl der Menüobjekte |
|  |  | - Radius\_x := Screen.Width div 3; // Radius in x-Richtung |
|  |  | - Radius\_y := Screen.Height div 3; // Radius in y-Richtung |
|  |  | + Radius\_x := Radius\*(Screen.Width / Screen.Height); // Radius in x-Richtung |
|  |  | + Radius\_y := Radius; // Radius in y-Richtung |
|  |  | + Anzahl := 5; |
|  |  | for i := 1 to Anzahl do |
|  |  | begin |
|  |  | x := Kreisposition\_x(i,Anzahl,ScreenMitte,Radius\_x); // x- und y-Koordinate für das i-te Objekt wird ermittelt |
|  | | @@ -96,7 +120,7 @@ procedure TMenue.FormPaint(Sender: TObject); |
|  |  | end; |
|  |  |  |

Testbilder für das Menü (08.10.13)

Olga hat die Bilder für das Menü erstellt und hochgeladen, damit sie in das Menü eingefügt werden können.







Fertige Bilder für Das Menü (09.10.13)

Olga hat die Bilder zu ihrer Zufriedenheit überarbeitet und neu eingefügt.

Zusätzlich hat sie ein Bild für das Lexikon eingefügt.









Kleine Änderungen am Programm (10.10.13)

Durchgeführt von Olga. Sie hat die Überschrift erstellt und in das Programm eingebettet.



Programm/Struktur.pas:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | procedure TMenue.MenuePosition(Radius:integer); |
|  |  | var i : integer; |
|  |  | x,y : integer; |
|  |  | Radius\_x,Radius\_y : real; |
|  |  | Buttonbreite : integer; |
|  |  | Anzahl:integer; |
|  |  | begin |
|  |  | ButtonBreite := (3\*Radius)div 4; // Buttonbreite wird in Abh�ngigkeit der |
|  |  | Image1.Width := ButtonBreite; // Bildschrimgr��e bestimmt |
|  |  | Image1.Height := ButtonBreite; |
|  |  | Image2.Width := ButtonBreite; // provisorisch habe ich Images als Platzhalter |
|  |  | Image2.Height := ButtonBreite; // f�r die sp�teren Butten genommen... |
|  |  | Image3.Width := ButtonBreite; // sie k�nnten Bilder enth�lten, Zeichen, nur Text |
|  |  | Image3.Height := ButtonBreite; // Form steht auch nicht fest... |
|  |  | Image4.Width := ButtonBreite; // Das Alles wird sich alles im Laufe der |
|  |  | Image4.Height := ButtonBreite; // Programmentwicklung noch ergeben. |
|  |  | - Image5.Width := ButtonBreite; |
|  |  | - Image5.Height := ButtonBreite; |
|  |  | - Image6.Width := ButtonBreite; |
|  |  | + Image5.Width := ButtonBreite + 100; |
|  |  | + Image5.Height := ButtonBreite + 100; |
|  |  | + {Image6.Width := ButtonBreite; |
|  |  | Image6.Height := ButtonBreite; |
|  |  | Image7.Width := ButtonBreite; |
|  |  | - Image7.Height := ButtonBreite; |
|  |  | + Image7.Height := ButtonBreite; } |
|  |  | // im Folgenden werden die Menüpunkte im Kreis(Ellipse) angeordnet: |
|  |  | Radius\_x := Radius\*(Screen.Width / Screen.Height); // Radius in x-Richtung |
|  |  | Radius\_y := Radius; // Radius in y-Richtung |
|  |  | - Anzahl := 7; |
|  |  | + Anzahl := 5; |
|  |  | for i := 1 to Anzahl do |
|  |  | begin |
|  |  | x := Kreisposition\_x(i,Anzahl,ScreenMitte,Radius\_x); // x- und y-Koordinate für das i-te Objekt wird ermittelt |
|  |  | y := Kreisposition\_y(i,Anzahl,ScreenMitte,Radius\_y); // dabei werden oben bestimmte Parameter �bergeben |
|  |  | case i of |
|  |  | 1: begin |
|  |  | Image1.Left := x - ButtonBreite div 2; // den Men�objekten werden ihre Koordinaten zugeordnet |
|  |  | Image1.Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | 2: begin |
|  |  | Image2.Left := x - ButtonBreite div 2; |
|  |  | Image2.Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | 3: begin |
|  |  | Image3.Left := x - ButtonBreite div 2; |
|  |  | Image3.Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | 4: begin |
|  |  | Image4.Left := x - ButtonBreite div 2; |
|  |  | Image4.Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | 5: begin |
|  |  | - Image5.Left := x - ButtonBreite div 2; |
|  |  | - Image5.Top := y - ButtonBreite div 2; |
|  |  | + Image5.Left := x - ButtonBreite div 2 - 50; |
|  |  | + Image5.Top := y - ButtonBreite div 2 ; |
|  |  | end; |
|  |  | - 6: begin |
|  |  | + { 6: begin |
|  |  | Image6.Left := x - ButtonBreite div 2; |
|  |  | Image6.Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | 7: begin |
|  |  | Image7.Left := x - ButtonBreite div 2; |
|  |  | Image7.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | + end;} |
|  |  | end; |
|  |  | end; |
|  |  | end; |
|  |  |  |
|  |  |  |

Eine Veränderung ( ebenfalls von Olga vorgenommen ) am Programm noch am gleichen Tag:

Programm/Struktur.pas:

|  |  |  |
| --- | --- | --- |
| end; |  |  |
|  |  |  |
|  |  | if Menue.Top > (Screen.Height div 6) \* 5 then close; // Sicherheitsschließen |
|  |  | + |
|  |  | + |
|  |  | + // Nur eine Idee. Wollte ausprobieren ob das auch ohne Komponenten geht. Erfolglos! |
|  |  | + |
|  |  | + {if (Y > Image1.Top) // Wenn der Mauszeiger sich in dem Feld des Bildes befindet |
|  |  | + and (X > Image1.Left) |
|  |  | + and (Y < Image1.Top + Image1.Height) |
|  |  | + and (X < Image1.Left + Image1.Width) then |
|  |  | + begin |
|  |  | + Cursor := crHelp; // dann soll sich der Zeiger ändern |
|  |  | + end else |
|  |  | + begin // In diesem Fall crHelp weil das Programm mit dem Handpointer |
|  |  | + if Cursor = crHelp then // und mit dem damit verbundenen Befehl vorher nicht klar kam. |
|  |  | + begin |
|  |  | + Cursor := crDefault; |
|  |  | + refresh; |
|  |  | + end; |
|  |  | + end;} |
|  |  | end; |
|  |  |  |
|  |  |  |
|  | |  |

Arne hat ebenfalls an diesem Tag die von Olga erstellten Menüobjekte in Komponenten verwandelt.

Programm/Imagebutton.pas:

|  |  |  |
| --- | --- | --- |
| +unit ImageButton; |  |  |
|  |  | + |
|  |  | +interface |
|  |  | + |
|  |  | +uses |
|  |  | + Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, |
|  |  | + ExtCtrls, axctrls; |
|  |  | + |
|  |  | +type |
|  |  | + TImageButton = class(TImage) |
|  |  | + private |
|  |  | + { Private-Deklarationen } |
|  |  | + FThemenfarbe1:TColor; |
|  |  | + FThemenfarbe2:TColor; |
|  |  | + FZoom:boolean; |
|  |  | + protected |
|  |  | + { Protected-Deklarationen } |
|  |  | + procedure MouseMove(Shift: TShiftState; X, |
|  |  | + Y: Integer); override; |
|  |  | + function Pixelfarbe(const x,y: integer): TColor; |
|  |  | + public |
|  |  | + { Public-Deklarationen } |
|  |  | + constructor Create(AOwner:TComponent);override; |
|  |  | + procedure BildLaden(Datei: string); |
|  |  | + procedure Vergroessern; |
|  |  | + Procedure Verkleinern; |
|  |  | + published |
|  |  | + { Published-Deklarationen } |
|  |  | + property OnMouseMove; |
|  |  | + property Stretch default true; |
|  |  | + property Themenfarbe1:Tcolor read FThemenfarbe1 write FThemenfarbe1 default clWhite; |
|  |  | + property Themenfarbe2:Tcolor read FThemenfarbe2 write FThemenfarbe2 default clWhite; |
|  |  | + property Zoom : boolean read FZoom write FZoom default false; |
|  |  | + end; |
|  |  | + |
|  |  | + |
|  |  | +procedure Register; |
|  |  | + |
|  |  | +implementation |
|  |  | + |
|  |  | +var ScreenMitte:TPoint; |
|  |  | + |
|  |  | + |
|  |  | +constructor TImageButton.Create(AOwner:TComponent); |
|  |  | +begin |
|  |  | + inherited Create(AOwner); |
|  |  | + Stretch := true; |
|  |  | + ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TImageButton.BildLaden(Datei: string); // im Internet gefundene Prozedur zum laden von |
|  |  | +var // unanimierten gifs mit transparentem Hintergrund |
|  |  | + FStream: TFileStream; |
|  |  | + OLEBild: TOleGraphic; |
|  |  | +begin |
|  |  | + OLEBild := TOleGraphic.Create; |
|  |  | + FStream := TFileStream.Create(Datei, fmOpenRead or fmShareDenyNone); |
|  |  | + try |
|  |  | + OLEBild.LoadFromStream(FStream); |
|  |  | + Picture.Assign(OLEBild); |
|  |  | + finally |
|  |  | + FStream.Free; |
|  |  | + OLEBild.free; |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure TImageButton.MouseMove(Shift:TShiftState;X,Y:integer); |
|  |  | +var k:integer; |
|  |  | +begin |
|  |  | + inherited MouseMove(Shift,X,Y); |
|  |  | + |
|  |  | + |
|  |  | + if (Pixelfarbe(X+Left,Y+Top) = Themenfarbe1) and // wenn die Farbe des Pixels unter der Maus |
|  |  | + (Height > k) then Zoom := false; // der Themenfarbe entspricht -> Verkleinerungs-Modus |
|  |  | + |
|  |  | + if (Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1) // wenn die Farbe des Pixels unter der Maus NICHT |
|  |  | + then Zoom := true; // der Themenfarbe entspricht -> Zoom-Modus |
|  |  | + |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TImageButton.Vergroessern; |
|  |  | +begin |
|  |  | + if Left + (Width div 2) > ScreenMitte.x then // Befindet sich das Objekt in der rechten Bildschirmhälfte |
|  |  | + begin |
|  |  | + Left := Left - 4; // vergrößert sich das Bild in die linke Richtung. |
|  |  | + Width := Width + 4; |
|  |  | + end else Width := Width + 4; // Wenn nicht, dann nach rechts. |
|  |  | + |
|  |  | + if Top + (Height div 2) > ScreenMitte.y then // Befindet sich das Objekt in der unteren Bildschrimhälfte |
|  |  | + begin |
|  |  | + Top := Top - 4; // vergrößert sich das Bild nach oben. |
|  |  | + Height := Height + 4; |
|  |  | + end else Height := Height + 4; // Wenn nicht dann nach unten. |
|  |  | +end; |
|  |  | + // -> Das Bild wird in die Mitte gezogen, wo eventuell das Masskottchen steht... |
|  |  | +procedure TImageButton.Verkleinern; |
|  |  | +begin |
|  |  | + if Left + (Width div 2) > ScreenMitte.x then // Gegenstück zum Vergrößern |
|  |  | + begin |
|  |  | + Left := Left + 4; |
|  |  | + Width := Width - 4; |
|  |  | + end else Width := Width - 4; |
|  |  | + |
|  |  | + if Top + (Height div 2) > ScreenMitte.y then |
|  |  | + begin |
|  |  | + Top := Top + 4; |
|  |  | + Height := Height - 4; |
|  |  | + end else Height := Height - 4; |
|  |  | +end; |
|  |  | + |
|  |  | +function TImageButton.Pixelfarbe(const x,y: integer): TColor; |
|  |  | +var |
|  |  | + c:TCanvas; |
|  |  | +begin |
|  |  | + c:=TCanvas.create; // Diese Funktion gibt die Pixelfarbe |
|  |  | + c.handle:= GetWindowDC(GetDesktopWindow); // an der Stelle x,y zurück |
|  |  | + result:=getpixel(c.handle,x,y); |
|  |  | + c.free; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure Register; |
|  |  | +begin |
|  |  | + RegisterComponents('Übung', [TImageButton]); |
|  |  | +end; |
|  |  | + |
|  |  | +end. |

Programm/Imagebutton.~pas:

|  |  |  |
| --- | --- | --- |
| +unit ImageButton; |  |  |
|  |  | + |
|  |  | +interface |
|  |  | + |
|  |  | +uses |
|  |  | + Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, |
|  |  | + ExtCtrls, axctrls; |
|  |  | + |
|  |  | +type |
|  |  | + TImageButton = class(TImage) |
|  |  | + private |
|  |  | + { Private-Deklarationen } |
|  |  | + FThemenfarbe1:TColor; |
|  |  | + FThemenfarbe2:TColor; |
|  |  | + protected |
|  |  | + { Protected-Deklarationen } |
|  |  | + procedure MouseMove(Shift: TShiftState; X, |
|  |  | + Y: Integer); override; |
|  |  | + function Pixelfarbe(const x,y: integer): TColor; |
|  |  | + public |
|  |  | + { Public-Deklarationen } |
|  |  | + constructor Create(AOwner:TComponent);override; |
|  |  | + procedure BildLaden(Datei: string); |
|  |  | + procedure pause(zeit:longint); |
|  |  | + published |
|  |  | + { Published-Deklarationen } |
|  |  | + property OnMouseMove; |
|  |  | + property Stretch default true; |
|  |  | + property Themenfarbe1:Tcolor read FThemenfarbe1 write FThemenfarbe1 default clWhite; |
|  |  | + property Themenfarbe2:Tcolor read FThemenfarbe2 write FThemenfarbe2 default clWhite; |
|  |  | + end; |
|  |  | + |
|  |  | + |
|  |  | +procedure Register; |
|  |  | + |
|  |  | +implementation |
|  |  | + |
|  |  | +procedure TImageButton.pause(zeit:longint); |
|  |  | +var zeit1 : longint; |
|  |  | +begin |
|  |  | + zeit1 := GetTickCount; |
|  |  | + repeat |
|  |  | + Application.ProcessMessages; |
|  |  | + until (GetTickCount - zeit1 > zeit); |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +constructor TImageButton.Create(AOwner:TComponent); |
|  |  | +begin |
|  |  | + inherited Create(AOwner); |
|  |  | + Stretch := true; |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TImageButton.BildLaden(Datei: string); |
|  |  | +var |
|  |  | + FStream: TFileStream; |
|  |  | + OLEBild: TOleGraphic; |
|  |  | +begin |
|  |  | + OLEBild := TOleGraphic.Create; |
|  |  | + FStream := TFileStream.Create(Datei, fmOpenRead or fmShareDenyNone); |
|  |  | + try |
|  |  | + OLEBild.LoadFromStream(FStream); |
|  |  | + Picture.Assign(OLEBild); |
|  |  | + finally |
|  |  | + FStream.Free; |
|  |  | + OLEBild.free; |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure TImageButton.MouseMove(Shift:TShiftState;X,Y:integer); |
|  |  | +var i,k,l: integer; |
|  |  | +begin |
|  |  | + inherited MouseMove(Shift,X,Y); |
|  |  | + |
|  |  | + k := round(3\*((Screen.Height / 3)+1)/ 4); |
|  |  | + l := round(4\*((Screen.Height / 3))/ 5); |
|  |  | + if Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1 then |
|  |  | + begin |
|  |  | + if Height < l then |
|  |  | + begin |
|  |  | + repeat; |
|  |  | + Left := Left - 1; |
|  |  | + Top := Top - 1; |
|  |  | + Width := Width +2; |
|  |  | + Height:= Height+2; |
|  |  | + pause(1); |
|  |  | + until Height = l; |
|  |  | + end; |
|  |  | + end; |
|  |  | + if Pixelfarbe(X+Left,Y+Top) = Themenfarbe1 then |
|  |  | + begin |
|  |  | + if Height > k then |
|  |  | + begin |
|  |  | + repeat; |
|  |  | + Left := Left + 1; |
|  |  | + Top := Top + 1; |
|  |  | + Width := Width -2; |
|  |  | + Height:= Height-2; |
|  |  | + pause(1); |
|  |  | + until Height = k; |
|  |  | + end; |
|  |  | + end; |
|  |  | +end; |
|  |  | + |
|  |  | +function TImageButton.Pixelfarbe(const x,y: integer): TColor; |
|  |  | +var |
|  |  | + c:TCanvas; |
|  |  | +begin |
|  |  | + c:=TCanvas.create; |
|  |  | + c.handle:= GetWindowDC(GetDesktopWindow); |
|  |  | + result:=getpixel(c.handle,x,y); |
|  |  | + c.free; |
|  |  | +end; |
|  |  | + |
|  |  | + |
|  |  | +procedure Register; |
|  |  | +begin |
|  |  | + RegisterComponents('Übung', [TImageButton]); |
|  |  | +end; |
|  |  | + |
|  |  | +end. |

Programm/Struktur.pas:

|  |  |  |
| --- | --- | --- |
| @@ -4,17 +4,13 @@ interface |  |  |
|  |  |  |
|  |  | uses |
|  |  | Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, |
|  |  | - StdCtrls, ExtCtrls, Math, jpeg; |
|  |  | + StdCtrls, ExtCtrls, Math, jpeg, ImageButton; |
|  |  |  |
|  |  | type |
|  |  | TMenue = class(TForm) |
|  |  | - Image1: TImage; |
|  |  | - Image2: TImage; |
|  |  | - Image3: TImage; |
|  |  | - Image4: TImage; |
|  |  | MenueEffekt: TTimer; |
|  |  | ImageScreen: TImage; |
|  |  | - Image5: TImage; |
|  |  | + Zoomen: TTimer; |
|  |  | procedure FormPaint(Sender: TObject); |
|  |  | procedure FormCreate(Sender: TObject); |
|  |  | procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  | | @@ -30,6 +26,7 @@ TMenue = class(TForm) |
|  |  | Zentrum:TPoint;Radius:real): integer; |
|  |  | procedure MenueEffektTimer(Sender: TObject); |
|  |  | procedure Startansicht(); |
|  |  | + procedure ZoomenTimer(Sender: TObject); |
|  |  | private |
|  |  | { Private-Deklarationen } |
|  |  | public |
|  | | @@ -43,6 +40,8 @@ TMenue = class(TForm) |
|  |  | MenuePos:integer = 1; |
|  |  | ScreenMitte:TPoint; |
|  |  |  |
|  |  | + MenueObjekt : array[1..10] of TImageBUtton; |
|  |  | + |
|  |  | implementation |
|  |  |  |
|  |  | {$R \*.DFM} |
|  | | @@ -54,11 +53,25 @@ procedure TMenue.FormPaint(Sender: TObject); |
|  |  | end; |
|  |  |  |
|  |  | procedure TMenue.FormCreate(Sender: TObject); |
|  |  | +var i : integer; |
|  |  | begin |
|  |  | Themenfarbe1 := RGB(244,164,96); //Themenfarben können sich durchs ganze |
|  |  | Themenfarbe2 := RGB(205,133,63); //Programm ziehen... (sind noch nicht beschlossen) |
|  |  | Self.DoubleBuffered := true; |
|  |  | ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | + |
|  |  | + for i := 1 to 5 do |
|  |  | + begin // Die Menüobjekte werden vom Typ |
|  |  | + MenueObjekt[i]:= TImageButton.Create(self); // ImageButton erstellt |
|  |  | + MenueObjekt[i].Parent := self; |
|  |  | + MenueObjekt[i].Themenfarbe1 := Themenfarbe1; // und es wird ihnen die aktuellen Themenfarbe |
|  |  | + MenueObjekt[i].Themenfarbe2 := Themenfarbe2; // übermittelt. |
|  |  | + end; |
|  |  | + MenueObjekt[1].BildLaden('Bilder/Karten-Menüpunkt.gif'); // Jedes Menüobjekt lädt sein bestimmtes Bild |
|  |  | + MenueObjekt[2].BildLaden('Bilder/Lexikon-Menüpunkt.gif'); // im gif-Format |
|  |  | + MenueObjekt[3].BildLaden('Bilder/Profil-Menüpunkt.gif'); |
|  |  | + MenueObjekt[4].BildLaden('Bilder/Spiel-Menüpunkt.gif'); |
|  |  | + MenueObjekt[5].BildLaden('Bilder/Titel.gif'); |
|  |  | end; |
|  |  |  |
|  |  | procedure TMenue.Startansicht(); |
|  | | @@ -103,21 +116,12 @@ procedure TMenue.MenuePosition(Radius:integer); |
|  |  | Buttonbreite : integer; |
|  |  | Anzahl:integer; |
|  |  | begin |
|  |  | - ButtonBreite := (3\*Radius)div 4; // Buttonbreite wird in Abhängigkeit der |
|  |  | - Image1.Width := ButtonBreite; // Bildschrimgröße bestimmt |
|  |  | - Image1.Height := ButtonBreite; |
|  |  | - Image2.Width := ButtonBreite; // provisorisch habe ich Images als Platzhalter |
|  |  | - Image2.Height := ButtonBreite; // für die späteren Butten genommen... |
|  |  | - Image3.Width := ButtonBreite; // sie könnten Bilder enthälten, Zeichen, nur Text |
|  |  | - Image3.Height := ButtonBreite; // Form steht auch nicht fest... |
|  |  | - Image4.Width := ButtonBreite; // Das Alles wird sich alles im Laufe der |
|  |  | - Image4.Height := ButtonBreite; // Programmentwicklung noch ergeben. |
|  |  | - Image5.Width := ButtonBreite + 100; |
|  |  | - Image5.Height := ButtonBreite + 100; |
|  |  | - {Image6.Width := ButtonBreite; |
|  |  | - Image6.Height := ButtonBreite; |
|  |  | - Image7.Width := ButtonBreite; |
|  |  | - Image7.Height := ButtonBreite; } |
|  |  | + ButtonBreite := (3\*Radius)div 4; // Buttonbreite wird in Abhängigkeit der Bildschrimgröße bestimmt |
|  |  | + for i := 1 to 5 do |
|  |  | + begin |
|  |  | + MenueObjekt[i].Width := ButtonBreite; |
|  |  | + MenueObjekt[i].Height:= ButtonBreite; |
|  |  | + end; |
|  |  | // im Folgenden werden die Menüpunkte im Kreis(Ellipse) angeordnet: |
|  |  | Radius\_x := Radius\*(Screen.Width / Screen.Height); // Radius in x-Richtung |
|  |  | Radius\_y := Radius; // Radius in y-Richtung |
|  | | @@ -126,36 +130,9 @@ procedure TMenue.MenuePosition(Radius:integer); |
|  |  | begin |
|  |  | x := Kreisposition\_x(i,Anzahl,ScreenMitte,Radius\_x); // x- und y-Koordinate für das i-te Objekt wird ermittelt |
|  |  | y := Kreisposition\_y(i,Anzahl,ScreenMitte,Radius\_y); // dabei werden oben bestimmte Parameter übergeben |
|  |  | - case i of |
|  |  | - 1: begin |
|  |  | - Image1.Left := x - ButtonBreite div 2; // den Menüobjekten werden ihre Koordinaten zugeordnet |
|  |  | - Image1.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | - 2: begin |
|  |  | - Image2.Left := x - ButtonBreite div 2; |
|  |  | - Image2.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | - 3: begin |
|  |  | - Image3.Left := x - ButtonBreite div 2; |
|  |  | - Image3.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | - 4: begin |
|  |  | - Image4.Left := x - ButtonBreite div 2; |
|  |  | - Image4.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | - 5: begin |
|  |  | - Image5.Left := x - ButtonBreite div 2 - 50; |
|  |  | - Image5.Top := y - ButtonBreite div 2 ; |
|  |  | - end; |
|  |  | - { 6: begin |
|  |  | - Image6.Left := x - ButtonBreite div 2; |
|  |  | - Image6.Top := y - ButtonBreite div 2; |
|  |  | - end; |
|  |  | - 7: begin |
|  |  | - Image7.Left := x - ButtonBreite div 2; |
|  |  | - Image7.Top := y - ButtonBreite div 2; |
|  |  | - end;} |
|  |  | - end; |
|  |  | + |
|  |  | + MenueObjekt[i].Left := x - ButtonBreite div 2; // jeder Komponente wird ihre Position übergeben. |
|  |  | + MenueObjekt[i].Top := y - ButtonBreite div 2; |
|  |  | end; |
|  |  | end; |
|  |  |  |
|  | | @@ -181,13 +158,31 @@ function TMenue.Kreisposition\_y(Objektnummer:integer;Objektanzahl:integer;Zentru |
|  |  | result := y; |
|  |  | end; |
|  |  |  |
|  |  | - |
|  |  | +procedure TMenue.ZoomenTimer(Sender: TObject); // Ein permanenter Timer... |
|  |  | +var i,k,l :integer; |
|  |  | +begin |
|  |  | + k := round(3\*((Screen.Height / 3)) / 4); // (normale Größe) |
|  |  | + l := round(15\*((Screen.Height / 3)) / 16); // (zoom Größe) |
|  |  | + for i := 1 to 4 do // 1-4: Titel wird nicht gezoomt!!! |
|  |  | + begin |
|  |  | + if (MenueObjekt[i].Zoom = true) and // prüft ob, ein Menüobjekt im Zoom-Modus ist |
|  |  | + (MenueObjekt[i].Height < l) then // und kleiner als die Zoom-End-Größe ist |
|  |  | + begin |
|  |  | + MenueObjekt[i].Vergroessern; // und lässt sie sich vergrößern, |
|  |  | + end; |
|  |  | + if (MenueObjekt[i].Zoom = false) and // oder wenn es nicht der Fall ist und das Menüobjekt |
|  |  | + (MenueObjekt[i].Height > k) then // auch nicht seine normale Größe besitzt |
|  |  | + begin |
|  |  | + MenueObjekt[i].Verkleinern; // schrumpft das Objekt wieder. |
|  |  | + end; |
|  |  | + end; |
|  |  | +end; |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | procedure TMenue.FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  |  | Y: Integer); |
|  |  | -var dif:integer; |
|  |  | +var dif,i:integer; |
|  |  | begin |
|  |  | if Y < 20 then // Wenn sich die Maus im oberen Bildschirmbereich |
|  |  | begin |
|  | | @@ -206,29 +201,17 @@ procedure TMenue.FormMouseMove(Sender: TObject; Shift: TShiftState; X, |
|  |  |  |
|  |  | if Menue.Align = alNone then // Wenn das Menü-Fenster im verschiebbaren Modus ist, |
|  |  | begin |
|  |  | - dif := Mouse.CursorPos.y -Y; |
|  |  | + dif := Mouse.CursorPos.y -Y; |
|  |  | Menue.Top := Y+dif; // verschiebt sich das Fenster mit der Maus in y-Richtung. |
|  |  | end; |
|  |  |  |
|  |  | if Menue.Top > (Screen.Height div 6) \* 5 then close; // Sicherheitsschließen |
|  |  |  |
|  |  | - |
|  |  | - // Nur eine Idee. Wollte ausprobieren ob das auch ohne Komponenten geht. Erfolglos! |
|  |  | - |
|  |  | - {if (Y > Image1.Top) // Wenn der Mauszeiger sich in dem Feld des Bildes befindet |
|  |  | - and (X > Image1.Left) |
|  |  | - and (Y < Image1.Top + Image1.Height) |
|  |  | - and (X < Image1.Left + Image1.Width) then |
|  |  | + for i := 1 to 4 do |
|  |  | begin |
|  |  | - Cursor := crHelp; // dann soll sich der Zeiger ändern |
|  |  | - end else |
|  |  | - begin // In diesem Fall crHelp weil das Programm mit dem Handpointer |
|  |  | - if Cursor = crHelp then // und mit dem damit verbundenen Befehl vorher nicht klar kam. |
|  |  | - begin |
|  |  | - Cursor := crDefault; |
|  |  | - refresh; |
|  |  | - end; |
|  |  | - end;} |
|  |  | + If MenueObjekt[i].Zoom = true // Sicherheitsverkleinern: Wenn die Maus wieder auf der Form ist |
|  |  | + then MenueObjekt[i].Zoom := false; // und das Menüobjekt noch nicht am Verkleinrn ist. |
|  |  | + end; |
|  |  | end; |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Ebenfalls am selben Tag hat Steffen Veränderungen am Programm vorgenommen.

Programm/Imagebutton.~pas:

|  |  |  |
| --- | --- | --- |
| @@ -12,6 +12,7 @@ type |  |  |
|  |  | { Private-Deklarationen } |
|  |  | FThemenfarbe1:TColor; |
|  |  | FThemenfarbe2:TColor; |
|  |  | + FZoom:boolean; |
|  |  | protected |
|  |  | { Protected-Deklarationen } |
|  |  | procedure MouseMove(Shift: TShiftState; X, |
|  | | @@ -21,13 +22,15 @@ type |
|  |  | { Public-Deklarationen } |
|  |  | constructor Create(AOwner:TComponent);override; |
|  |  | procedure BildLaden(Datei: string); |
|  |  | - procedure pause(zeit:longint); |
|  |  | + procedure Vergroessern; |
|  |  | + Procedure Verkleinern; |
|  |  | published |
|  |  | { Published-Deklarationen } |
|  |  | property OnMouseMove; |
|  |  | property Stretch default true; |
|  |  | property Themenfarbe1:Tcolor read FThemenfarbe1 write FThemenfarbe1 default clWhite; |
|  |  | property Themenfarbe2:Tcolor read FThemenfarbe2 write FThemenfarbe2 default clWhite; |
|  |  | + property Zoom : boolean read FZoom write FZoom default false; |
|  |  | end; |
|  |  |  |
|  |  |  |
|  | | @@ -35,24 +38,18 @@ procedure Register; |
|  |  |  |
|  |  | implementation |
|  |  |  |
|  |  | -procedure TImageButton.pause(zeit:longint); |
|  |  | -var zeit1 : longint; |
|  |  | -begin |
|  |  | - zeit1 := GetTickCount; |
|  |  | - repeat |
|  |  | - Application.ProcessMessages; |
|  |  | - until (GetTickCount - zeit1 > zeit); |
|  |  | -end; |
|  |  | +var ScreenMitte:TPoint; |
|  |  |  |
|  |  |  |
|  |  | constructor TImageButton.Create(AOwner:TComponent); |
|  |  | begin |
|  |  | inherited Create(AOwner); |
|  |  | Stretch := true; |
|  |  | + ScreenMitte := Point(Screen.Width div 2,Screen.Height div 2); // Mitte des Screen wird ermittelt |
|  |  | end; |
|  |  |  |
|  |  | -procedure TImageButton.BildLaden(Datei: string); |
|  |  | -var |
|  |  | +procedure TImageButton.BildLaden(Datei: string); // im Internet gefundene Prozedur zum laden von |
|  |  | +var // unanimierten gifs mit transparentem Hintergrund |
|  |  | FStream: TFileStream; |
|  |  | OLEBild: TOleGraphic; |
|  |  | begin |
|  | | @@ -69,46 +66,54 @@ end; |
|  |  |  |
|  |  |  |
|  |  | procedure TImageButton.MouseMove(Shift:TShiftState;X,Y:integer); |
|  |  | -var i,k,l: integer; |
|  |  | +var k:integer; |
|  |  | begin |
|  |  | inherited MouseMove(Shift,X,Y); |
|  |  | + |
|  |  | + if (Pixelfarbe(X+Left,Y+Top) = Themenfarbe1) and // wenn die Farbe des Pixels unter der Maus |
|  |  | + (Height > k) then Zoom := false; // der Themenfarbe entspricht -> Verkleinerungs-Modus |
|  |  | + |
|  |  | + if (Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1) // wenn die Farbe des Pixels unter der Maus NICHT |
|  |  | + then Zoom := true; // der Themenfarbe entspricht -> Zoom-Modus |
|  |  | + |
|  |  | +end; |
|  |  | + |
|  |  | +procedure TImageButton.Vergroessern; |
|  |  | +begin |
|  |  | + if Left + (Width div 2) > ScreenMitte.x then // Befindet sich das Objekt in der rechten Bildschirmhälfte |
|  |  | + begin |
|  |  | + Left := Left - 4; // vergrößert sich das Bild in die linke Richtung. |
|  |  | + Width := Width + 4; |
|  |  | + end else Width := Width + 4; // Wenn nicht, dann nach rechts. |
|  |  |  |
|  |  | - k := round(3\*((Screen.Height / 3)+1)/ 4); |
|  |  | - l := round(4\*((Screen.Height / 3))/ 5); |
|  |  | - if Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1 then |
|  |  | + if Top + (Height div 2) > ScreenMitte.y then // Befindet sich das Objekt in der unteren Bildschrimhälfte |
|  |  | + begin |
|  |  | + Top := Top - 4; // vergrößert sich das Bild nach oben. |
|  |  | + Height := Height + 4; |
|  |  | + end else Height := Height + 4; // Wenn nicht dann nach unten. |
|  |  | +end; |
|  |  | + // -> Das Bild wird in die Mitte gezogen, wo eventuell das Masskottchen steht... |
|  |  | +procedure TImageButton.Verkleinern; |
|  |  | +begin |
|  |  | + if Left + (Width div 2) > ScreenMitte.x then // Gegenstück zum Vergrößern |
|  |  | begin |
|  |  | - if Height < l then |
|  |  | - begin |
|  |  | - repeat; |
|  |  | - Left := Left - 1; |
|  |  | - Top := Top - 1; |
|  |  | - Width := Width +2; |
|  |  | - Height:= Height+2; |
|  |  | - pause(1); |
|  |  | - until Height = l; |
|  |  | - end; |
|  |  | - end; |
|  |  | - if Pixelfarbe(X+Left,Y+Top) = Themenfarbe1 then |
|  |  | + Left := Left + 4; |
|  |  | + Width := Width - 4; |
|  |  | + end else Width := Width - 4; |
|  |  | + |
|  |  | + if Top + (Height div 2) > ScreenMitte.y then |
|  |  | begin |
|  |  | - if Height > k then |
|  |  | - begin |
|  |  | - repeat; |
|  |  | - Left := Left + 1; |
|  |  | - Top := Top + 1; |
|  |  | - Width := Width -2; |
|  |  | - Height:= Height-2; |
|  |  | - pause(1); |
|  |  | - until Height = k; |
|  |  | - end; |
|  |  | - end; |
|  |  | + Top := Top + 4; |
|  |  | + Height := Height - 4; |
|  |  | + end else Height := Height - 4; |
|  |  | end; |
|  |  |  |
|  |  | function TImageButton.Pixelfarbe(const x,y: integer): TColor; |
|  |  | var |
|  |  | c:TCanvas; |
|  |  | begin |
|  |  | - c:=TCanvas.create; |
|  |  | - c.handle:= GetWindowDC(GetDesktopWindow); |
|  |  | + c:=TCanvas.create; // Diese Funktion gibt die Pixelfarbe |
|  |  | + c.handle:= GetWindowDC(GetDesktopWindow); // an der Stelle x,y zurück |
|  |  | result:=getpixel(c.handle,x,y); |
|  |  | c.free; |
|  |  | end; |
|  | |  |

(11.10.13)

An diesem Tag haben wir das Lastenheft abgegeben, an dem die gesamte Gruppe mitgearbeitet hat.

Zusätzlich hat Arne die Größe der Überschrift verändert.

Programm/Imagebutton.pas:

|  |  |  |
| --- | --- | --- |
|  |  | inherited MouseMove(Shift,X,Y); |
|  |  |  |
|  |  | if (Pixelfarbe(X+Left,Y+Top) = Themenfarbe1) and // wenn die Farbe des Pixels unter der Maus |
|  |  | - (Height > k) then Zoom := false; // der Themenfarbe entspricht -> Verkleinerungs-Modus |
|  |  | - |
|  |  | - if (Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1) // wenn die Farbe des Pixels unter der Maus NICHT |
|  |  | - then Zoom := true; // der Themenfarbe entspricht -> Zoom-Modus |
|  |  | - |
|  |  | + (Height > k) then |
|  |  | + begin |
|  |  | + Zoom := false; // der Themenfarbe entspricht -> Verkleinerungs-Modus |
|  |  | + Cursor := crDefault; |
|  |  | + end; |
|  |  | + if (Pixelfarbe(X+Left,Y+Top) <> Themenfarbe1) then // wenn die Farbe des Pixels unter der Maus NICHT |
|  |  | + begin |
|  |  | + Zoom := true; // der Themenfarbe entspricht -> Zoom-Modus |
|  |  | + Cursor := crHandpoint; |
|  |  | + end; |
|  |  | end; |
|  |  |  |
|  |  | procedure TImageButton.Vergroessern; |
|  |  | begin |
|  |  | if Left + (Width div 2) > ScreenMitte.x then // Befindet sich das Objekt in der rechten Bildschirmh�lfte |
|  |  | begin |
|  |  | Left := Left - 4; // vergr��ert sich das Bild in die linke Richtung. |
|  |  | Width := Width + 4; |
|  |  | end else Width := Width + 4; // Wenn nicht, dann nach rechts. |
|  |  |  |
|  |  | if Top + (Height div 2) > ScreenMitte.y then // Befindet sich das Objekt in der unteren Bildschrimh�lfte |
|  |  | begin |
|  |  | Top := Top - 4; // vergr��ert sich das Bild nach oben. |
|  |  | Height := Height + 4; |
|  |  | end else Height := Height + 4; // Wenn nicht dann nach unten. |
|  |  | end; |
|  |  | // -> Das Bild wird in die Mitte gezogen, wo eventuell das Masskottchen steht... |
|  |  | procedure TImageButton.Verkleinern; |
|  |  | begin |
|  |  | if Left + (Width div 2) > ScreenMitte.x then // Gegenst�ck zum Vergr��ern |
|  |  | begin |
|  |  | Left := Left + 4; |
|  |  | Width := Width - 4; |
|  |  | end else Width := Width - 4; |
|  |  |  |
|  |  | if Top + (Height div 2) > ScreenMitte.y then |
|  |  | begin |
|  |  | Top := Top + 4; |
|  |  | Height := Height - 4; |
|  |  | end else Height := Height - 4; |
|  |  | end; |
|  |  |  |
|  |  | function TImageButton.Pixelfarbe(const x,y: integer): TColor; |
|  |  | var |
|  |  | c:TCanvas; |
|  |  | begin |
|  |  | c:=TCanvas.create; // Diese Funktion gibt die Pixelfarbe |
|  |  | c.handle:= GetWindowDC(GetDesktopWindow); // an der Stelle x,y zur�ck |
|  |  | result:=getpixel(c.handle,x,y); |
|  |  | c.free; |
|  |  | end; |
|  |  |  |
|  |  |  |
|  | |  |

Programm/Struktur.pas:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | Anzahl:integer; |
|  |  | | begin |
|  |  | | ButtonBreite := (3\*Radius)div 4; // Buttonbreite wird in Abhängigkeit der Bildschrimgröße bestimmt |
|  |  | | - for i := 1 to 5 do |
|  |  | | + for i := 1 to 4 do |
|  |  | | begin |
|  |  | | MenueObjekt[i].Width := ButtonBreite; |
|  |  | | MenueObjekt[i].Height:= ButtonBreite; |
|  |  | | end; |
|  |  | | + |
|  |  | | + MenueObjekt[5].Width := (ButtonBreite\*5) div 2; |
|  |  | | + MenueObjekt[5].Height:= ButtonBreite; |
|  |  | | // im Folgenden werden die Menüpunkte im Kreis(Ellipse) angeordnet: |
|  |  | | Radius\_x := Radius\*(Screen.Width / Screen.Height); // Radius in x-Richtung |
|  |  | | Radius\_y := Radius; // Radius in y-Richtung |
|  |  | | Anzahl := 5; |
|  |  | | - for i := 1 to Anzahl do |
|  |  | | + for i := 1 to 5 do |
|  |  | | begin |
|  |  | | x := Kreisposition\_x(i,Anzahl,ScreenMitte,Radius\_x); // x- und y-Koordinate für das i-te Objekt wird ermittelt |
|  |  | | y := Kreisposition\_y(i,Anzahl,ScreenMitte,Radius\_y); // dabei werden oben bestimmte Parameter übergeben |
|  |  | |  |
|  |  | | - MenueObjekt[i].Left := x - ButtonBreite div 2; // jeder Komponente wird ihre Position übergeben. |
|  |  | | - MenueObjekt[i].Top := y - ButtonBreite div 2; |
|  |  | | + if i <> 5 then |
|  |  | | + begin |
|  |  | | + MenueObjekt[i].Left := x - ButtonBreite div 2; // jeder Komponente wird ihre Position übergeben. |
|  |  | | + MenueObjekt[i].Top := y - ButtonBreite div 2; |
|  |  | | + end else |
|  |  | | + begin |
|  |  | | + MenueObjekt[i].Left := x - ((ButtonBreite\*5) div 2) div 2; // jeder Komponente wird ihre Position übergeben. |
|  |  | | + MenueObjekt[i].Top := y - ButtonBreite div 2; |
|  |  | | + end; |
|  |  | | end; |
|  |  | | end; |
|  |  | |  |
|  | | @@ -141,7 +151,7 @@ procedure TMenue.ZoomenTimer(Sender: TObject); // Ein permanenter Timer... | |
|  |  | | var i,k,l :integer; |
|  |  | | begin |
|  |  | | k := round(3\*((Screen.Height / 3)) / 4); // (normale Größe) |
|  |  | | - l := round(15\*((Screen.Height / 3)) / 16); // (zoom Größe) |
|  |  | | + l := round(15\*((Screen.Height / 3)) / 18); // (zoom Größe) |
|  |  | | for i := 1 to 4 do // 1-4: Titel wird nicht gezoomt!!! |
|  |  | | begin |
|  |  | | if (MenueObjekt[i].Zoom = true) and // prüft ob, ein Menüobjekt im Zoom-Modus ist |
|  | | @@ -171,11 +181,8 @@ procedure TMenue.FormMouseMove(Sender: TObject; Shift: TShiftState; X, | |
|  |  | | Canvas.Rectangle(0,0,ClientWidth,20); // in der 2. Themenfarbe gefärbt. |
|  |  | | end else |
|  |  | | begin // andererseits wird, |
|  |  | | - if Cursor = crHandpoint then // nur wenn es nicht schon der Fall ist, |
|  |  | | - begin |
|  |  | | - Cursor := crDefault; // der Maus der Normale Zeiger zugeordnet |
|  |  | | - refresh; // und der andersfarbige Bereich wieder gelöscht. |
|  |  | | - end; |
|  |  | | + Cursor := crDefault; // der Maus der Normale Zeiger zugeordnet |
|  |  | | + refresh; // und der andersfarbige Bereich wieder gelöscht. |
|  |  | | end; |
|  |  | |  |
|  |  | | if Menue.Align = alNone then // Wenn das Menü-Fenster im verschiebbaren Modus ist, |
|  |  | | begin |
|  |  | | dif := Mouse.CursorPos.y -Y; |
|  |  | | Menue.Top := Y+dif; // verschiebt sich das Fenster mit der Maus in y-Richtung. |
|  |  | | end; |
|  |  | |  |
|  |  | | if Menue.Top > (Screen.Height div 6) \* 5 then close; // Sicherheitsschlie�en |
|  |  | |  |
|  |  | | for i := 1 to 4 do |
|  |  | | begin |
|  |  | | If MenueObjekt[i].Zoom = true // Sicherheitsverkleinern: Wenn die Maus wieder auf der Form ist |
|  |  | | then MenueObjekt[i].Zoom := false; // und das Men�objekt noch nicht am Verkleinrn ist. |
|  |  | | end; |
|  |  | | end; |

Erstellung einer Ortsliste und Zufällige Auswahl auf diesen 76 Städten (17.10.13)

An diesem Tage hat Arne eine Liste der wichtigsten bzw. größten Städte Deutschlands aufgestellt, die in dem Minispiel, in dem man die Lage dieser Städte bestimmen muss, gebraucht wird. Dann hat er eine zufällige Auswahl aus diesen 76 Städten programmiert, sodass nie die gleiche Abfragereihenfolge stattfinden wird.

Programm, welches in der Endverwendung des Spiels nicht mehr verwendet wird:

unit Orte;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls;

type

TOrte = record

Index : integer;

Ortsname : string[20];

Schwierigkeit : string[10];

KoSy\_x, KoSy\_y : integer;

end;

TForm1 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

EdtIndex: TEdit;

EdtOrt: TEdit;

EdtKoSy\_x: TEdit;

Label4: TLabel;

EdtKoSy\_y: TEdit;

BtnSpeichern: TButton;

BtnWeiter: TButton;

BtnZurueck: TButton;

BtnNeu: TButton;

Label5: TLabel;

EdtSchwierigkeit: TEdit;

Label6: TLabel;

procedure FormCreate(Sender: TObject);

procedure SatzLadenAnzeigen;

procedure SatzSpeichern;

procedure Neu;

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure BtnSpeichernClick(Sender: TObject);

procedure BtnWeiterClick(Sender: TObject);

procedure BtnZurueckClick(Sender: TObject);

procedure BtnNeuClick(Sender: TObject);

private

{ Private-Deklarationen }

aktueller\_record : integer;

ROrte : TOrte;

Orte\_Datei : file of TOrte;

public

{ Public-Deklarationen }

end;

var

Form1: TForm1;

implementation

{$R \*.DFM}

procedure TForm1.FormCreate(Sender: TObject);

begin

AssignFile(Orte\_Datei,'Orte\_KoSy.dat'); // Datei wird geöffnet

aktueller\_record := 1;

if FileExists('Orte\_KoSy.dat') then

begin

Reset(Orte\_Datei);

SatzLadenAnzeigen; // erster Datensatz wird geladen

end

else Rewrite(Orte\_Datei);

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.SatzLadenAnzeigen;

begin

Seek(Orte\_Datei,aktueller\_record-1);

if FileSize(Orte\_Datei) > 0 then

begin

Read(Orte\_Datei,ROrte); // Der Datensatz wird in den Record geladen

with ROrte do

begin

EdtIndex.Text := IntToStr(Index); // und in den Edits ausgegeben

EdtOrt.Text := Ortsname;

EdtSchwierigkeit.Text := Schwierigkeit;

EdtKoSy\_x.Text := IntToStr(KoSy\_x);

EdtKoSy\_y.Text := IntToStr(KoSy\_y);

end;

end;

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.BtnSpeichernClick(Sender: TObject);

begin

SatzSpeichern;

end;

procedure TForm1.SatzSpeichern; // Speichern...

begin

with ROrte do

begin

Index := StrToInt(EdtIndex.Text);

Ortsname := EdtOrt.Text;

Schwierigkeit := EdtSchwierigkeit.Text;

KoSy\_x := StrToInt(EdtKoSy\_x.Text);

KoSy\_y := StrToInt(EdtKoSy\_y.Text);

end;

Seek(Orte\_Datei,aktueller\_record-1);

Write(Orte\_Datei,ROrte);

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.Neu;

begin

SatzSpeichern;

aktueller\_record := FileSize(Orte\_Datei) + 1;

EdtIndex.Text := IntToStr(aktueller\_record);

EdtOrt.Text := '';

EdtSchwierigkeit.Text := '';

EdtKoSy\_x.Text := '';

EdtKoSy\_y.Text := '';

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.BtnWeiterClick(Sender: TObject);

begin

SatzSpeichern;

if aktueller\_record < FileSize(Orte\_Datei) then

begin

inc(aktueller\_record);

SatzLadenAnzeigen;

end;

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.BtnZurueckClick(Sender: TObject);

begin

SatzSpeichern;

if aktueller\_record > 1 then

begin

dec(aktueller\_record);

SatzLadenAnzeigen;

end;

Label5.Caption := IntToStr(aktueller\_record);

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

begin

CloseFile(Orte\_Datei);

end;

procedure TForm1.BtnNeuClick(Sender: TObject);

begin

SatzSpeichern;

neu;

end;

end.

Prgramm/Programm.~dpr

|  |  |  |
| --- | --- | --- |
| ... | ... | @@ -1,16 +0,0 @@ |
| 1 |  | -program Programm; |
| 2 |  | - |
| 3 |  | -uses |
| 4 |  | - Forms, |
| 5 |  | - Struktur in 'Struktur.pas' {Menue}, |
| 6 |  | - Karte in 'Karte.pas' {Orte\_Finden}, |
| 7 |  | - ImageButton in 'ImageButton.pas'; |
| 8 |  | - |
| 9 |  | -{$R \*.RES} |
| 10 |  | - |
| 11 |  | -begin |
| 12 |  | - Application.Initialize; |
| 13 |  | - Application.CreateForm(TMenue, Menue); |
| 14 |  | - Application.CreateForm(TOrte\_Finden, Orte\_Finden); |
| 15 |  | - Application.Run; |
| 16 |  | -end. |

Prgramm/Struktur.pas

|  |  |  |
| --- | --- | --- |
|  | | @@ -244,9 +244,11 @@ procedure TMenue.FensterOeffnen(Button:integer); |
| 244 | 244 | case Button of |
| 245 | 245 | 4: begin |
| 246 | 246 | Application.CreateForm(TOrte\_Finden, Orte\_Finden); |
| 247 |  | - Orte\_Finden.ShowModal; |
|  | 247 | + Orte\_Finden.BringToFront; |
|  | 248 | + Orte\_Finden.ShowModal |
| 248 | 249 | end; |
| 249 | 250 | end; |
|  | 251 | + //self. |
| 250 | 252 | end; |
| 251 | 253 |  |

Erstellung einer neuen Form auf der das erste Spiel platzfinden wird:

unit Karte;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

ImageOrten, ShapeSchliessen, OleCtrls, SHDocVw, StdCtrls, ExtCtrls;

type

TOrte\_Finden = class(TForm)

BtnNeu: TButton;

ShpHintergrund1: TShape;

LblUeberschrift: TLabel;

Maskottchen: TLabel;

LblStadt: TLabel;

LblSchwierigkeit: TLabel;

pruefenTimer: TTimer;

LblPunkte: TLabel;

LblEntfernung: TLabel;

procedure FormCreate(Sender: TObject);

procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure BtnNeuClick(Sender: TObject);

procedure ShpHintergrund1MouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure pruefenTimerTimer(Sender: TObject);

private

{ Private-Deklarationen }

public

{ Public-Deklarationen }

end;

var

Orte\_Finden: TOrte\_Finden;

Themenfarbe1: TColor;

Themenfarbe2: TColor;

SchliessenShape:TShapeSchliessen;

SuchKarte:TImageOrten;

Rand:integer;

index:integer = 1;

implementation

uses Struktur;

{$R \*.DFM}

procedure TOrte\_Finden.FormCreate(Sender: TObject);

var Stadt:integer;

begin

self.DoubleBuffered := true;

Themenfarbe1 := Menue.Themenfarbe1; // Themenfarbe wird aus dem Menü-Formular gelesen

Themenfarbe2 := Menue.Themenfarbe2;

Orte\_Finden.Color := Themenfarbe1;

SuchKarte := TImageOrten.Create(self); // Die Suchkarte wird erzeugt

SuchKarte.Parent := self;

SuchKarte.Picture.Bitmap.LoadFromFile(ExtractFilePath(ParamStr(0)) + 'Bilder/DKarte Ohne Städte.bmp'); // Die Deutschlandkarte wird geladen

LblStadt.Caption := SuchKarte.SatzLadenAnzeigen(1); // ein zufälliger Datensatz wird geladen

SchliessenShape := TShapeSchliessen.Create(self); // Erstellen der Schließen-Komponente

SchliessenShape.Parent := self;

SchliessenShape.Themenfarbe1 := Themenfarbe1; // die Themenfarben werden übergeben

SchliessenShape.Themenfarbe2 := Themenfarbe2;

SchliessenShape.Fenster := Orte\_Finden; // Wichtig! Das Fenster wird übergeben, damit die Komponente weiß

// welches Fenster geschlossen werden soll.

Rand := Screen.Height div 30; // Rand ist eine bestimme Länge in Abhängigkeit der Fensterhöhe

ShpHintergrund1.Left := Rand; // Objekte werden platziert

ShpHintergrund1.Top := Rand;

ShpHintergrund1.Width := Screen.Width - 3\*Rand - ((Screen.Height\*133)div 195);

ShpHintergrund1.Height:= Screen.Height - 2\*Rand;

ShpHintergrund1.Brush.Color := Themenfarbe2;

LblUeberschrift.Font.Size := Screen.Height div 30;

LblUeberschrift.Top := 2\*Rand;

LblUeberschrift.Left := ((ShpHintergrund1.Width + 2\*Rand) div 2)

- LblUeberschrift.Width div 2;

Maskottchen.Top := 17\*Rand;

Maskottchen.Left := 2\*Rand;

Maskottchen.Width := Screen.Width - 5\*Rand - ((Screen.Height\*133)div 195);

Maskottchen.Height := 11\*Rand;

LblStadt.Font.Size := Screen.Height div 30;

LblStadt.Top := 5\*Rand;

LblStadt.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblStadt.Width div 2);

LblSchwierigkeit.Font.Size := Screen.Height div 40;

LblSchwierigkeit.Top := 7\*Rand;

LblSchwierigkeit.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblSchwierigkeit.Width div 2);

LblPunkte.Font.Size := Screen.Height div 30;

LblPunkte.Top := 9\*Rand;

LblPunkte.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblPunkte.Width div 2);

LblEntfernung.Font.Size := Screen.Height div 40;

LblEntfernung.Top := 11\*Rand;

LblEntfernung.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblEntfernung.Width div 2);

end;

procedure TOrte\_Finden.FormMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

if SchliessenShape.inaktiv = false then // Wenn das Shape zum Schließen noch

begin // aktiv ist,

SchliessenShape.inaktiv := true; // wird sein Status auf inaktiv gesetzt

SchliessenShape.Repaint; // und es zeichnet sich neu.

end;

end;

procedure TOrte\_Finden.BtnNeuClick(Sender: TObject);

begin

SuchKarte.Picture := nil; // Die Suchkarte wird geleert und

SuchKarte.Picture.Bitmap.LoadFromFile(ExtractFilePath(ParamStr(0)) + 'Bilder/DKarte Ohne Städte.bmp'); // neu geladen

SuchKarte.geklickt := false;

if index = 77 then index := 1 else inc(index); // ein neuer zufälliger Datensatz wird geladen

LblStadt.Caption := SuchKarte.SatzLadenAnzeigen(index);

LblStadt.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblStadt.Width div 2);

LblSchwierigkeit.Caption := SuchKarte.ROrte.Schwierigkeit;

LblSchwierigkeit.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblSchwierigkeit.Width div 2);

pruefenTimer.Enabled := true;

end;

procedure TOrte\_Finden.ShpHintergrund1MouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

if SchliessenShape.inaktiv = false then // Wenn das Shape zum Schließen noch

begin // aktiv ist,

SchliessenShape.inaktiv := true; // wird sein Status auf inaktiv gesetzt

SchliessenShape.Repaint; // und es zeichnet sich neu.

end;

end;

procedure TOrte\_Finden.pruefenTimerTimer(Sender: TObject);

begin

if SuchKarte.geklickt = true then // Wenn geklickt wurde,

begin

LblPunkte.Caption := IntToStr(SuchKarte.Punkte); // werden die berechneten Punkte

LblEntfernung.Caption := IntToStr(SuchKarte.Entfernung) + ' km'; // und die Entfernung in km aus der Suchkarte gelesen

LblPunkte.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblPunkte.Width div 2);

LblEntfernung.Left := (ShpHintergrund1.Width div 2 + Rand)-(LblEntfernung.Width div 2);

pruefenTimer.Enabled := false;

end;

end;

end.

Eine neue Komponente, auf der die Karte und Orte geladen werden:

unit ImageOrten;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

ExtCtrls, Math;

type

TOrte = record

Index : integer;

Ortsname : string[20];

Schwierigkeit : string[10];

KoSy\_x, KoSy\_y : integer;

end;

TImageOrten = class(TImage)

private

{ Private-Deklarationen }

aktueller\_record : integer;

Orte\_Datei : file of TOrte;

protected

{ Protected-Deklarationen }

procedure MouseUp(Button: TMouseButton;

Shift: TShiftState; X, Y: Integer); override;

public

{ Public-Deklarationen }

ROrte : TOrte;

geklickt:boolean;

Punkte : integer;

Entfernung : integer;

constructor Create(AOwner:TComponent); override;

function SatzLadenAnzeigen(index:integer) : string;

published

{ Published-Deklarationen }

end;

procedure Register;

implementation

uses karte;

var

KoSy: array of array of boolean; //Dynamisches zweidimensionales Array welches jedes Pixel auf der Komponente/auf dem Bild darstellt

Ort:TPoint;

constructor TImageOrten.Create(AOwner:TComponent);

begin

inherited Create(AOwner);

Height := (Screen.Height\*14) div 15; // Die Komponente platziert sich auf dem Formular.

Width := (Height\*19) div 26;

Top := Screen.Height div 30;

Left := Screen.Width - Width - Top;

Cursor := crCross; // Ein Zielkreuz wird als Cursor eingestellt.

Stretch := true;

geklickt := false;

end;

function TImageOrten.SatzLadenAnzeigen(index:integer): string;

var i,k:integer;

begin

SetLength(KoSy,1900,2600); // Die Längen des Array werden gesetzt

for i := Low(KoSy) to High(KoSy) do // und alle Speicherplätze mit false belegt

for k := Low(KoSy[i]) to High(KoSy[i]) do

KoSy[i,k] := false;

AssignFile(Orte\_Datei,ExtractFilePath(ParamStr(0)) + 'Ortskoordinaten/Orte\_KoSy.dat'); // Die Datei, in der die Ortskoordinaten hinterlegt sind,

aktueller\_record := index-1; // wird geöffnet (innerhalb dieser Prozedur wird die Datei auch wieder geschlossen

if FileExists(ExtractFilePath(ParamStr(0)) + 'Ortskoordinaten/Orte\_KoSy.dat') then // damit durch kurze Öffnungzeiten, keine Blockierung der Datei

begin // zustande kommt! So können mehrer Programme "gleichzeitig" draufzugreifen

Reset(Orte\_Datei);

end else exit;

Seek(Orte\_Datei,aktueller\_record); // Der Zeiger wird vor den zu Öffnenden Datensatz gesetzt (s.L 69)

if FileSize(Orte\_Datei) > 0 then

begin

Read(Orte\_Datei,ROrte);

with ROrte do

begin // Diese Koordinate wird im KoSy auf true gesetzt

KoSy[KoSy\_x,KoSy\_y] := true;

for i := Low(KoSy) to High(KoSy) do

for k := Low(KoSy[i]) to High(KoSy[i]) do

if KoSy[i,k] = true then

begin

Ort := Point(i,k); // Die Koordinaten werden in der globalen Variable Ort gespeichert

break;

end;

Canvas.MoveTo(Ort.x,Ort.y); // Diese Koordinate wird Canvas.MoveTo zugeordnet

result := Ortsname;

end;

end;

CloseFile(Orte\_Datei); // die Datei wird wieder geschlossen

end;

procedure TImageOrten.MouseUp(Button: TMouseButton; //MouseUp

Shift: TShiftState; X, Y: Integer);

var

dif\_x, dif\_y : real;

dif\_hoch:real;

a,b : integer;

EntfernungTemp:real;

begin

if geklickt = false then

begin

dif\_hoch := (2600 / Height); // Verhältniss zwischen der Bildgröße und der tatsächlichen Auflösung, die vom Bildschrimformat abhängt

X := round(dif\_hoch \* X); // x und y werden nach diesem Verhältnis neu berechnet

Y := round(dif\_hoch \* Y);

Canvas.Pen.Width := 8;

Canvas.Pen.Color := clRed;

Canvas.LineTo(X,Y); // eine Linie wird vom Ort zum gedrückten Maus gezogen

dif\_x := X - Ort.x; // die Differenz der x und y koordinaten

dif\_y := Y - Ort.y;

EntfernungTemp := sqrt( (dif\_x \* dif\_x) + (dif\_y \* dif\_y)); // Die Pixelentfernung wird mit dem Satz des Phytagoras errechnet

Entfernung := round(EntfernungTemp \* ( 613 / 1791 )); // Der Bildspezifische Maßstab wird in die Entfernung mit einbezogen

if Entfernung < 100 then b := 2 else b := 1; // Punktevergabe...

if Entfernung < 50 then b := 3;

if Entfernung < 5 then b := 4;

if ROrte.Schwierigkeit = 'schwer' then a := 3; // In der Datei sind neben den Ortsnamen und den Koordinaten auch die schwierigkeiten hinterlegt

if ROrte.Schwierigkeit = 'mittel' then a := 2;

if ROrte.Schwierigkeit = 'leicht' then a := 1;

Punkte := round(a\*b\*(100/Entfernung)); //

Canvas.Brush.Style := bsClear;

Canvas.Pen.Width := 6;

Canvas.Pen.Color := clGreen;

Canvas.Ellipse(Ort.x-29,Ort.y-29,Ort.x+29,Ort.y+29); // 10 km Radius Hilfszeichnungen

Canvas.Pen.Color := clYellow;

Canvas.Ellipse(Ort.x-146,Ort.y-146,Ort.x+146,Ort.y+146); // 50 km Radius

Canvas.Pen.Color := clRed;

Canvas.Ellipse(Ort.x-292,Ort.y-292,Ort.x+292,Ort.y+292); // 100 km Radius

geklickt := true;

end;

end;

procedure Register;

begin

RegisterComponents('Übung', [TImageOrten]);

end;

end.

Entwicklung des Designs des Anmeldefensters (18.10.13)

An diesem Tag hat Steffen das Design des Anmeldefensters entworfen.

Einbau der Verschlüsselung in das Programm (01.11.13)

An diesem Tag hat Steffen, die von ihm entwickelte, Verschlüsselung in das Programm integriert. Hierbei musste besondere Vorsicht gelten, da, wie schon erwähnt, der Datenschutz bei Schülern groß geschrieben werden muss.

Entwicklung der Anmeldung von Schülern (15.11.13)

An diesem Tag hat Steffen die Anmeldung der Schüler entwickelt.

Hier ist besonders zu beachten, dass diese Nutzer des Programms sehr eingeschränkte Rechte haben müssen, da sie keinerlei Einsicht auf Daten Dritter haben dürfen, wie beispielsweise der Lernstand, oder die Zeit, die ein anderer mit dem Programm verbracht hat.

Einbau weiterer Modi in das Anmeldefenster (29.11.13)

An diesem Tag hat Steffen weitere Modi, wie z.B. eine Sicherheitsabfrage in das Anmeldefenster integriert, die sicherstellen sollen, dass niemand einen fremden Account benutzt.

Arbeit am Maskottchen von Olga (01.9.13-22.11.13)

Einzelne Schritte:

* Olga hat die vorher überlegten Ideen versucht in die Realität umzusetzen.
* Zuerst mit dem Bleistift eine Skizze gezeichnet. Diese dann in den Computer eingescannt.
* Dann am Computer die Skizze weiter bearbeitet. Fehler wurden wieder mit Hand verbessert. ( Insgesamt ca. 4 Tage Arbeitszeit ).
* Arbeit an den verschiedenen Perspektiven auf dieselbe Art und Weise.
* Überlegung über die Umsetzung der verschiedenen Animationen. Dabei wurde jedes einzelne Frame neu gezeichnet. Alleine für 4 Sekunden Gehen wurden 74 Frames erstellt.
* Das Programmieren der verschiedenen Animationen wurde auf unterschiedliche Weisen ausprobiert (TAnimate, TMediaPlayer), da viele nicht funktioniert haben hat man sich letztendlich auf die Benutzung eines/einer Timers/Schleife zurückgegriffen.
* Weitere Ideen für Animationen werden folgen. (Programmieren mit Hilfe von Arne)

Lehrerconsole wurde hochgeladen von Steffen(09.12.13).

An diesem Tag hat Steffen die von ihm erstellte Lehrerconsole hochgeladen, für die er zuständig war. Diese hat er neben dem mitarbeiten am restlichen Programm in Alleinarbeit erstellt. Zu beachten ist, dass nur die wichtigsten Fortschritte in der Dokumentation dokumentiert sind. Tage an denen keine wichtigen Fortschritte gemacht wurden, sondern „alltägliche“ Arbeit gemacht wurde, wurden nicht dokumentiert.

Lexikon (09.12.13)

An diesem Tag hat Arne das Lexikon erstellt, in dem all die Begriffe stehen, die für den Unterricht essentiell sind und die, die Schüler nicht durch reines Allgemeinwissen können.

Prgramm/Lexikon/Bearbeiten.dpr

|  |  |  |
| --- | --- | --- |
| ... | ... | @@ -0,0 +1,13 @@ |
|  | 1 | +program Bearbeiten; |
|  | 2 | + |
|  | 3 | +uses |
|  | 4 | + Forms, |
|  | 5 | + LexikonBearbeiten in 'LexikonBearbeiten.pas' {Form1}; |
|  | 6 | + |
|  | 7 | +{$R \*.RES} |
|  | 8 | + |
|  | 9 | +begin |
|  | 10 | + Application.Initialize; |
|  | 11 | + Application.CreateForm(TForm1, Form1); |
|  | 12 | + Application.Run; |
|  | 13 | +end. |

Prgramm/Lexikon/LexikonBearbeiten.pas

|  |  |  |
| --- | --- | --- |
| ... | ... | @@ -0,0 +1,307 @@ |
|  | 1 | unit LexikonBearbeiten; |
|  | 2 |  |
|  | 3 | interface |
|  | 4 |  |
|  | 5 | uses |
|  | 6 | Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, |
|  | 7 | StdCtrls, ComCtrls; |
|  | 8 |  |
|  | 9 | type |
|  | 10 | TForm1 = class(TForm) |
|  | 11 | EdtStichwort: TEdit; |
|  | 12 | Label1: TLabel; |
|  | 13 | Label2: TLabel; |
|  | 14 | EdtIndex: TEdit; |
|  | 15 | Label3: TLabel; |
|  | 16 | BtnErsetzen: TButton; |
|  | 17 | BtnNeu: TButton; |
|  | 18 | Label4: TLabel; |
|  | 19 | BtnWeiter: TButton; |
|  | 20 | BtnZurueck: TButton; |
|  | 21 | BtnSpeichern: TButton; |
|  | 22 | REdtText: TRichEdit; |
|  | 23 | REdtKopie: TRichEdit; |
|  | 24 | procedure FormCreate(Sender: TObject); |
|  | 25 | procedure BtnErsetzenClick(Sender: TObject); |
|  | 26 | procedure BtnNeuClick(Sender: TObject); |
|  | 27 | procedure Laden(Datensatz:integer); |
|  | 28 | procedure schreiben(Datensatz:integer); |
|  | 29 | procedure BtnWeiterClick(Sender: TObject); |
|  | 30 | procedure BtnZurueckClick(Sender: TObject); |
|  | 31 | procedure BtnSpeichernClick(Sender: TObject); |
|  | 32 | procedure speichern; |
|  | 33 | procedure Verschluesseln; |
|  | 34 | procedure Entschluesseln; |
|  | 35 | procedure ErzeugeGa; |
|  | 36 | procedure addition(x:integer); |
|  | 37 | procedure FormClose(Sender: TObject; var Action: TCloseAction); |
|  | 38 | private |
|  | 39 | { Private-Deklarationen } |
|  | 40 | public |
|  | 41 | { Public-Deklarationen } |
|  | 42 | end; |
|  | 43 |  |
|  | 44 | var |
|  | 45 | Form1: TForm1; |
|  | 46 | Menge: integer; |
|  | 47 |  |
|  | 48 | ga, ga2: string; |
|  | 49 | lenA : integer; |
|  | 50 | kt, gt : string; |
|  | 51 | lenT, p : integer; |
|  | 52 | c : char; |
|  | 53 | schl: string; |
|  | 54 | LenSchl: integer; |
|  | 55 | Schluessel: String; |
|  | 56 |  |
|  | 57 | Const |
|  | 58 | ka = ',-./0123456789:;Ô?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz'; |
|  | 59 |  |
|  | 60 | implementation |
|  | 61 |  |
|  | 62 | {$R \*.DFM} |
|  | 63 |  |
|  | 64 | procedure TForm1.FormCreate(Sender: TObject); |
|  | 65 | begin |
|  | 66 | REdtKopie.Lines.LoadfromFile('Lexikon.txt'); // Lexikondatei wird ins REdtKopie geladen |
|  | 67 |  |
|  | 68 | Schluessel := 'abcd'; |
|  | 69 | Entschluesseln; // und entschlüsselt. |
|  | 70 |  |
|  | 71 | Menge := StrToInt(REdtKopie.Lines[0]); // in der ersten Linie is die menge der Datensätze hintelegt |
|  | 72 | Label4.Caption := IntToStr(Menge); |
|  | 73 | Laden(Menge); // der letzte Datensatz wird geladen |
|  | 74 | end; |
|  | 75 |  |
|  | 76 | procedure TForm1.Laden(Datensatz:integer); |
|  | 77 | var Linie,j:integer; |
|  | 78 | temp:string; |
|  | 79 | begin |
|  | 80 |  |
|  | 81 | EdtIndex.Text := IntToStr(Datensatz); |
|  | 82 | temp := ''; |
|  | 83 | Linie := 1; |
|  | 84 | repeat |
|  | 85 | temp := REdtKopie.Lines[Linie]; //Die Linien werden ausgelesen |
|  | 86 | inc(Linie); |
|  | 87 | until temp = IntToStr(Datensatz); // bis der richtige Datensatz gefunden wurde |
|  | 88 | EdtStichwort.Text := REdtKopie.Lines[Linie]; // das Stichwort wird aus der nächsten Zeile gelesen |
|  | 89 |  |
|  | 90 | inc(Linie); |
|  | 91 | temp := REdtKopie.Lines[Linie]; |
|  | 92 | j := 0; |
|  | 93 | repeat |
|  | 94 | if j = 0 then REdtText.Lines[j] := temp // Die Absätze werden gelesen |
|  | 95 | else REdtText.Lines.add(temp); |
|  | 96 | inc(Linie); |
|  | 97 | temp := REdtKopie.Lines[Linie]; |
|  | 98 | inc(j); |
|  | 99 | until (temp = IntToStr(Datensatz+1)) or (temp = ''); |
|  | 100 | end; |
|  | 101 |  |
|  | 102 |  |
|  | 103 | procedure TForm1.BtnErsetzenClick(Sender: TObject); |
|  | 104 | begin |
|  | 105 | schreiben(StrToInt(EdtIndex.Text)); |
|  | 106 | end; |
|  | 107 |  |
|  | 108 |  |
|  | 109 | procedure TForm1.schreiben(Datensatz:integer); |
|  | 110 | var temp,temp2:string; |
|  | 111 | Linie,Linie2,i,dif:integer; |
|  | 112 | begin |
|  | 113 | if Datensatz < Menge then // Wenn der zuschreibende Datensatz nicht der letzte ist, |
|  | 114 | begin |
|  | 115 |  |
|  | 116 | Linie := 0; |
|  | 117 | repeat |
|  | 118 | temp := REdtKopie.Lines[Linie]; |
|  | 119 | inc(Linie); |
|  | 120 | until temp = IntToStr(Datensatz); |
|  | 121 | Linie2 := Linie; |
|  | 122 | repeat |
|  | 123 | temp := REdtKopie.Lines[Linie2]; |
|  | 124 | inc(Linie2); |
|  | 125 | until temp = IntToStr(Datensatz+1); |
|  | 126 |  |
|  | 127 | dif := Linie2-Linie+1; |
|  | 128 | repeat |
|  | 129 | temp := REdtKopie.Lines[Linie+dif-1]; // wird er von den folgenden überschrieben |
|  | 130 | if (length(temp) > 3) then REdtKopie.Lines[Linie] := temp |
|  | 131 | else if temp <> '' then REdtKopie.Lines[Linie] := IntToStr(StrToInt(temp)-1); |
|  | 132 | inc(Linie); |
|  | 133 | until temp = ''; // und neu hinten ran gesetzt. |
|  | 134 | REdtKopie.Lines[Linie-1] := IntToStr(Menge); |
|  | 135 | EdtIndex.Text := IntToStr(Menge); |
|  | 136 | REdtKopie.Lines[Linie] := EdtStichwort.Text; |
|  | 137 | inc(Linie); |
|  | 138 |  |
|  | 139 | i := 0; |
|  | 140 | temp := REdtKopie.Lines[Linie]; |
|  | 141 | temp2 := REdtText.Lines[i]; |
|  | 142 | repeat |
|  | 143 | if temp <> '' then |
|  | 144 | if temp2 <> '' then REdtKopie.Lines[Linie] := temp2 |
|  | 145 | else REdtKopie.Lines.Delete(Linie) |
|  | 146 | else REdtKopie.Lines.Add(temp2); |
|  | 147 | inc(i); |
|  | 148 | inc(Linie); |
|  | 149 | temp := REdtKopie.Lines[Linie]; |
|  | 150 | temp2 := REdtText.Lines[i]; |
|  | 151 | until (temp = '') and (temp2 = ''); |
|  | 152 |  |
|  | 153 | end else // wenn es der letzte Datensatz ist, |
|  | 154 | begin |
|  | 155 |  |
|  | 156 | Linie := 1; |
|  | 157 | repeat |
|  | 158 | temp := REdtKopie.Lines[Linie]; |
|  | 159 | inc(Linie); |
|  | 160 | until temp = IntToStr(Menge); |
|  | 161 | Linie2 := Linie; |
|  | 162 | repeat |
|  | 163 | temp := REdtKopie.Lines[Linie2]; |
|  | 164 | inc(Linie2); |
|  | 165 | until temp = ''; |
|  | 166 |  |
|  | 167 | REdtKopie.Lines[Linie] := EdtStichwort.Text; |
|  | 168 | inc(Linie); |
|  | 169 |  |
|  | 170 | i := 0; |
|  | 171 | temp := REdtKopie.Lines[Linie]; |
|  | 172 | temp2 := REdtText.Lines[i]; |
|  | 173 | repeat |
|  | 174 | if temp <> '' then |
|  | 175 | if temp2 <> '' then REdtKopie.Lines[Linie] := temp2 // wird der letzte einfach überschrieben |
|  | 176 | else REdtKopie.Lines.Delete(Linie) |
|  | 177 | else REdtKopie.Lines.Add(temp2); |
|  | 178 | inc(i); |
|  | 179 | inc(Linie); |
|  | 180 | temp := REdtKopie.Lines[Linie]; |
|  | 181 | temp2 := REdtText.Lines[i]; |
|  | 182 | until (temp = '') and (temp2 = ''); |
|  | 183 | end; |
|  | 184 | end; |
|  | 185 |  |
|  | 186 |  |
|  | 187 | procedure TForm1.BtnWeiterClick(Sender: TObject); |
|  | 188 | begin |
|  | 189 | if StrToInt(EdtIndex.Text) < Menge then |
|  | 190 | begin |
|  | 191 | REdtText.Lines.Clear; |
|  | 192 | Laden(StrToInt(EdtIndex.Text)+1); |
|  | 193 | end; |
|  | 194 | end; |
|  | 195 |  |
|  | 196 | procedure TForm1.BtnZurueckClick(Sender: TObject); |
|  | 197 | begin |
|  | 198 | if StrToInt(EdtIndex.Text) > 1 then |
|  | 199 | begin |
|  | 200 | REdtText.Lines.Clear; |
|  | 201 | Laden(StrToInt(EdtIndex.Text)-1); |
|  | 202 | end; |
|  | 203 | end; |
|  | 204 |  |
|  | 205 |  |
|  | 206 | procedure TForm1.BtnNeuClick(Sender: TObject); |
|  | 207 | begin |
|  | 208 | inc(Menge); |
|  | 209 | EdtIndex.Text := IntToStr(Menge); // die Menge wird um einen erhöht und |
|  | 210 | EdtStichwort.Text := ''; |
|  | 211 | REdtText.Lines.Clear; // die Edtit werden geleert. |
|  | 212 |  |
|  | 213 | BtnErsetzen.Enabled := false; |
|  | 214 | BtnWeiter.Enabled := false; |
|  | 215 | BtnZurueck.Enabled := false; |
|  | 216 | BtnNeu.Enabled := false; |
|  | 217 | BtnSpeichern.Enabled := true; |
|  | 218 | end; |
|  | 219 |  |
|  | 220 |  |
|  | 221 | procedure TForm1.BtnSpeichernClick(Sender: TObject); |
|  | 222 | begin |
|  | 223 | speichern; |
|  | 224 | end; |
|  | 225 |  |
|  | 226 | procedure TForm1.speichern; |
|  | 227 | var temp:string; |
|  | 228 | i:integer; |
|  | 229 | begin |
|  | 230 | REdtKopie.Lines[0] := IntToStr(Menge); |
|  | 231 | REdtKopie.Lines.add(IntToStr(Menge)); |
|  | 232 | REdtKopie.Lines.add(EdtStichwort.Text); |
|  | 233 |  |
|  | 234 | i := 0; |
|  | 235 | temp := REdtText.Lines[i]; // ein neuer Datensatz wird ans Ende der Liste geschrieben |
|  | 236 | repeat |
|  | 237 | REdtKopie.Lines.add(temp); |
|  | 238 | inc(i); |
|  | 239 | temp := REdtText.Lines[i]; |
|  | 240 | until temp = ''; |
|  | 241 |  |
|  | 242 | BtnErsetzen.Enabled := true; |
|  | 243 | BtnWeiter.Enabled := true; |
|  | 244 | BtnZurueck.Enabled := true; |
|  | 245 | BtnNeu.Enabled := true; |
|  | 246 | BtnSpeichern.Enabled := false; |
|  | 247 | end; |
|  | 248 |  |
|  | 249 | procedure TForm1.Verschluesseln; |
|  | 250 | var i:integer; |
|  | 251 | begin |
|  | 252 | kt := REdtKopie.Text; |
|  | 253 | lenT := length(kt); |
|  | 254 | gt := ''; |
|  | 255 | for i := 1 to lenT do |
|  | 256 | begin |
|  | 257 | ErzeugeGa; |
|  | 258 | addition(i); |
|  | 259 | c := kt[i]; |
|  | 260 | p := pos(c,ka); |
|  | 261 | if p <> 0 then gt := gt + copy (ga2,p,1) |
|  | 262 | else gt := gt + c; |
|  | 263 | end; |
|  | 264 | REdtKopie.Text := gt; |
|  | 265 | end; |
|  | 266 |  |
|  | 267 | procedure TForm1.Entschluesseln; |
|  | 268 | var i:integer; |
|  | 269 | begin |
|  | 270 | gt := REdtKopie.Text; |
|  | 271 | lenT := length(gt); |
|  | 272 | kt := ''; |
|  | 273 | for i := 1 to lenT do |
|  | 274 | begin |
|  | 275 | ErzeugeGa; |
|  | 276 | addition(i); |
|  | 277 | c := gt[i]; |
|  | 278 | p := pos(c,ga2); |
|  | 279 | if p <> 0 then kt := kt + copy(ka,p,1) |
|  | 280 | else kt := kt + c;; |
|  | 281 | end; |
|  | 282 | REdtKopie.Text := kt; |
|  | 283 | end; |
|  | 284 |  |
|  | 285 | procedure TForm1.ErzeugeGa; |
|  | 286 | var Wert1, Wert2: integer; |
|  | 287 | i: integer; |
|  | 288 | schlZahl: integer; |
|  | 289 | begin |
|  | 290 | ga := ''; |
|  | 291 | schlZahl := ord(schluessel[1]); |
|  | 292 | lenA := length(ka); |
|  | 293 | For i := 1 to lenA do |
|  | 294 | begin |
|  | 295 | c := ka[i]; |
|  | 296 | Wert1 := ord(c); |
|  | 297 | Wert2 := ((Wert1-44)\*schlZahl mod 79)+44; |
|  | 298 | ga := ga + chr(Wert2); |
|  | 299 | end; |
|  | 300 | end; |
|  | 301 |  |
|  | 302 | procedure TForm1.addition (x: integer); |
|  | 303 | var c2: char; |
|  | 304 | p2, zaehler: integer; |
|  | 305 | begin |
|  | 306 | lenSchl := length (schluessel); |
|  | 307 | zaehler := x mod lenSchl +1; |

Pflichtenheft und Dokumentation (11.10.13-25.12.13)

In diesem Zeitraum haben Niklas und Dario an der Dokumentation und dem Pflichtenheft gearbeitet. Sie werden beides dem weiteren Verlauf fortgehend anpassen, damit die Dokumentation und das Pflichtenheft immer auch dem neuesten Stand sind und der Auftraggeber später die Entwicklung des Programms nachprüfen kann.

Weltkugel (11.10.13-29.12.13)

In dieser Zeit hat Artur sich um ein weiteres Minispiel gekümmert. Die Benennung der Weltkugel nach Kontinente, Ozeane, usw.

Gruppe: Steffen, Arne, Olga, Artur, Niklas, Dario