|  |
| --- |
| Kjell Sonnenberg  Sebastian Hagen  Heiko Hoffmann  Dominik Sander  Klasse: QIIb  Gymnasium Glinde  Oher Weg 24  21509 Glinde |
| Codedokumentation |
| Aufbau und Funktionen der Game-Engine |
|  |
| Fach: Informatik |
| Betreuende Lehrkraft: Herr Buhk |
| Projektlaufzeit: 02.09.2017 – 14.12.2017 |

Hallo

Hallo

Hallo

Inhalt

[Inhalt 1](#_Toc501377928)

[Einführung 2](#_Toc501377929)

[1 API-Part: Flows 3](#_Toc501377930)

[1.1 Unit: GameFlow 4](#_Toc501377931)

[TGameFlowCustomItem 5](#_Toc501377932)

[TGameFlowList 13](#_Toc501377933)

[1.2 Unit: FlowTree 19](#_Toc501377934)

[1.3 Unit: CustomFlow 19](#_Toc501377935)

[1.4 Unit: ConversationFlow 20](#_Toc501377936)

[1.5 Unit: AnimationFlow 20](#_Toc501377937)

[2 API-Part: Frames 21](#_Toc501377938)

[2.1 Unit: GameFrame 21](#_Toc501377939)

[TGameFrame 21](#_Toc501377940)

[3 API-Part: GUIs 26](#_Toc501377941)

[3.1 Unit: GameGUI 26](#_Toc501377942)

[TGameGUI 26](#_Toc501377943)

[3.2 Unit: GameGUIFrame 30](#_Toc501377944)

[TGameGUIFrame 30](#_Toc501377945)

[3.3 Unit: ConversationGUI 31](#_Toc501377946)

[3.4 Unit: InventoryGUI 31](#_Toc501377947)

[4 API-Part: Objects 32](#_Toc501377948)

[4.1 Unit: GameObject 33](#_Toc501377949)

[TGameBase 33](#_Toc501377950)

[TGameObject 36](#_Toc501377951)

[4.2 Unit: ForegroundObj 41](#_Toc501377952)

[TForegroundObj 41](#_Toc501377953)

[4.3 Unit: Hotspot 44](#_Toc501377954)

[THotspot 44](#_Toc501377955)

[4.4 Unit: Item 50](#_Toc501377956)

[4.5 Unit: Player 50](#_Toc501377957)

[4.6 Unit: Background 50](#_Toc501377958)

[5 API-Part: Sonstige 51](#_Toc501377959)

[5.1 Unit: GameManager 52](#_Toc501377960)

[TGameManager 52](#_Toc501377961)

Einführung

Die Game-Engine ist ein auf das FMX-Framework basierendes Framework, mit dem das Erstellen von Spielen (insb. Point-and-Click- Adventures) mit wenig Programmieraufwand ermöglicht werden soll. Realisiert wird dieses Ziel dadurch, dass Objekte und Prozesse im Spiel schon im Design-Time-Editor bearbeitet werden können, sodass die eigenhändige Programmierung auf ein Minimum reduziert wird. Dabei ist die Engine leicht erweiterbar, sodass durchaus auch komplexere Aufgaben vom Anwender programmiert werden können.

Die Basis der Entwicklung eines Adventures bieten sog. GameFrames. Diese Frames sind der sichtbare Teil des Adventures, in ihnen findet am Ende das Spielgeschehen statt. Auf ihnen werden alle Objekte gesetzt, mit denen der Spieler interagieren kann (Items, Personen, etc.). Die Eigenschaften dieser Objekte (Größe, Position, etc.) können schon zur Design-Zeit eingestellt werden. Zur Laufzeit wird immer nur ein GameFrame im Anwendungsfenster gezeigt.

Prozesse im Spiel werden mit sog. FlowLists modelliert. Diese können Items unterschiedlichen Typs aufnehmen. Die sog. FlowItems haben je nach Typ unterschiedliche Aufgaben (z. B. Abspielen von Sounds, etc.). Beim Ausführen einer FlowList werden alle Items der Liste nacheinander ausgeführt. Dabei sind FlowLists und Items ebenfalls schon zur Design-Zeit editierbar.

Die grafischen Benutzeroberflächen außerhalb des eigentlichen Spiels (z. B. das Hauptmenü) werden durch die sog. GameGUIs dargestellt. GameGUIs können schon zur Design-Zeit designt werden und ebenfalls zur Design-Zeit im Anwendungsfenster (nicht in den Frames) platziert werden.

Callback-Methoden – und Events sorgen dafür, dass der Anwender auf Ereignisse im Spiel reagieren kann. Callback-Events können ebenfalls zur Design-Zeit erstellt werden und sind letztendlich Methodenzeiger, denen eine Methode zugewiesen wird. Diese Methode wird beim Eintreten des Events dann ausgeführt. Beim Erstellen des Events zur Design-Zeit wird automatisch eine Methode erstellt, in der der Anwender Code schreiben kann, welcher beim Eintreten des Events ausgeführt wird.

Wie abgesprochen werden hier nur die für den Aufbau und das Verständnis der Game-Engine wichtigen Basisklassen detailliert dokumentiert. Die detaillierte Dokumentation einer Klasse beinhaltet alle öffentlichen Methoden und Eigenschaften sowie protected Eigenschaften und die protected Methoden, die vom Anwender aufgerufen bzw. überladen werden sollen. Alle hier dokumentierten Klassen und Units befinden sich im Ordner „GameObjects“, die Quelldateien außerhalb dieses Ordners erstellen lediglich die Anwendung (Adventure.cpp) und stellen das Programmfenster bereit (Unit GameForm).

Hinweis: Als Unit werden alle Dateien mit demselben Namen aber unterschiedlicher Dateiendung benannt. Dateien mit der Endung „\*.cpp“ beinhalten die Implementierungen von Klassen und Funktionen; Dateien mit der Endung „\*.h“ enthalten die Klassendeklarationen. Dateien mit der Endung „\*.fmx“ enthalten die zur Design-Zeit eingestellten Werte von Frames und enthaltenen Objekten.

1. API-Part: Flows

In diesem Teil der API (im Unterordner Flows) sind die FlowLists- und Items implementiert. In den Units „GameFlow“ und „FlowTree“ sind die Basisklassen für alle FlowLists- und Items definiert.

|  |  |  |
| --- | --- | --- |
| **Klassenname** | **Unit** | **Funktion** |
| TGameFlowList | GameFlow | Basisklasse aller FlowLists (*siehe TGameFlowList*) |
| TGameFlowCustomItem | GameFlow | Basisklasse aller FlowListItems (*siehe TGameFlowCustomItem*) |
| TReferenceFlowItem | GameFlow | Basisklasse aller FlowItems, die ein GameObject referenzieren (*siehe GameFlow*) |
| TGameFlowItem | GameFlow | Von TGameFlowCustomItem abgeleitete Klasse, die alle Eigenschaften in den \_\_published-Bereich veröffentlicht |
| IGameRunnable | GameFlow | Interface, was für alle ausführbaren Elemente implementiert wird (*siehe GameFlow*) |
| TAudioFlowItem | AudioFlow | Spielt Audiodateien ab. Die Untertitel (Caption) werden als Sprechblase beim referenzierten Objekt (Object) angezeigt. |
| TAnimationFlowItem | AnimationFlow | FlowItem, mit denen Animationen (Bildabfolgen) einfach erstellt werden können (*siehe AnimationFlow*) |
| TConversationFlowItem | ConversationFlow | FlowItem ist Teil einer Konversation (*siehe ConversationFlow*) |
| TCustomFlowItem | CustomFlow | Basisklasse zum Erstellen von eigenen FlowItems, die aus schon vorhandenen Items bestehen (*siehe CustomFlow*) |
| TEventFlowItem | EventFlows | Basisklasse für alle FlowItems, dessen Ausführung keine Zeit benötigt (z. B. Setzten einer Eigenschaft) |
| TCustomEventFlowItem | EventFlows | Anwender kann mit dem Item eigene Events programmieren, indem er das OnEvent-Callback programmiert. |
| TStopFlowItem | EventFlows | Stoppt ein laufendes FlowItem. StopIndex gibt den Index des zu stoppenden FlowItems an. |
| TGiveItemFlowItem | EventFlows | Lädt das Item in der Eigenschaft „Item“ in das Inventar. |
| TLoadLocationFlowItem | EventFlows | Öffnet das Frame mit dem Klassennamen von „Location“. „OpenFlowName“ ist der Name der FlowList, die beim Öffnen des Frames ausgeführt werden soll (*siehe TGameManager::OpenFrame*) |
| TChangeStateFlowItem | EventFlows | Ändert den State vom Hotspot mit dem Namen „HotspotName“ im Frame „FrameName“ zum State „State“ |
| TChangeLocalStateFlowItem | EventFlows | Ändert den State von „Object“ zum State „State“ |
| TChangeAnimationFlowItem | EventFlows | Ändert die Animation von „Object“ zu „Animation“ |
| TLoadPicFlowItem | EventFlows | Lädt das Bild „Picture“ in das GameObject „Object“ |
| TOpenConvFlowItem | EventFlows | Startet die Konversation mit der Person „Object“ |

|  |  |  |
| --- | --- | --- |
| TMoveObjFlowItem | EventFlows | Bewegt das GameObject „Object“ um „LeftChange“ und „TopChange“ nach Links bzw. Oben |
| TChangePropertyFlowItem | EventFlows | Ändert die Eigenschaft „PropertyName“ des GameObjects „Object“ auf „PropertyValue“ |
| TVariadicFlowList | FlowLists | Liste, die alle FlowItems enthalten kann |
| TEventFlowList | FlowLists | Liste, die nur EventFlowItems enthalten kann |
| TFlowTreeList | FlowTree | Baumstruktur, die verästelte FlowLists enthalten (*siehe FlowTree*) |
| TFlowTreeItem | FlowTree | Item des FlowTrees (*siehe FlowTree*) |
| TSleepFlowItem | SleepFlow | FlowItem, was die Zeit „Sleep“ verstreichen lässt, bevor die Ausführung der FlowList fortgesetzt wird |

* 1. Unit: GameFlow

In dieser Unit befinden sich die Basisklassen für das FlowList-System. Neben der Basisklasse der FlowLists (TGameFlowList) und der FlowItems (TGameFlowCustomItem) sind noch TReferenceFlowItem, das Interface IGameRunnable und Collections für Name-FlowList-Paare implementiert.

IGameRunnable legt die Methoden für ausführbare Objekte fest. Beim Implementieren dieser Methoden ist zu beachten, dass die Start-Methode die Ausführung von Beginn an startet, wenn das Objekt gestoppt oder noch nicht gestartet wurde, aber die Ausführung nach dem Pausieren des Objekts von dem Punkt an fortführt, an dem es pausiert wurde.

TReferenceFlowItem ist eine von TGameFlowCustomItem abgeleitete Template-Klasse. Der Template-Parameter ist der Typ der Eigenschaft „Object“. Beim Erzeugen des Items wird „Object“ das Stammobjekt der FlowList des Items zugeordnet, wenn das Stammobjekt zum Typ der Eigenschaft passt.

TFlowListNameItem ist ein CollectionItem, welches FlowLists einen Namen zuordnet, die übergeordneten Collections sind vom Typ TFlowListNameCollection; diese können auf ihre Items über den Index oder den Namen zugreifen.

TFlowListNameSelItem und dessen TFlowListNameSelCollection sind von diesen Klassen abgeleitet und implementieren das Auswählen eines der enthaltenen Items über den Namen. Diese Listen sind die Basis für die Animations- oder States-Listen. Template-Wrapper (TFlowListCollectionTmpl, TFlowListNameCollectionTmpl, TFlowListNameSelCollectionTmpl) vereinfachen das Ableiten dieser Listen, da so nicht bei jeder neuen abgeleiteten Klasse, die Items-Eigenschaft auf die neue Item-Klasse neu gecastet werden muss, sondern einfach als Template-Parameter übergeben werden kann.

TGameFlowCustomItem

Signatur

class PACKAGE TGameFlowCustomItem : public TCustomListItem,public IGameRunnable

Funktion

TGameFlowCustomItem ist die abstrakte Basisklasse für alle Items von ausführbaren Listen in der Game-Engine. Abgeleitete Klassen müssen die DoStart-, DoStop- und DoPause-Methode sowie die IsRunning- und IsPaused-Methode überschreiben. Dabei soll der Klasse eine Aktion zugeordnet werden, die mit DoStart gestartet, mit DoStop gestoppt und mit DoPause pausiert werden soll.

Viele Eigenschaften dieser Klasse sind im protected-Bereich deklariert, können aber je nach Bedarf in abgeleiteten Klassen \_\_published gesetzt werden, um eine Bearbeitung der Eigenschaften zur Design-Zeit zu ermöglichen. Dabei können (und sollen) auch weitere Eigenschaften diesem Bereich hinzugefügt werden, um eine vollständige Einstellung des Items zur Design-Zeit zu ermöglichen.

Methoden

|  |  |
| --- | --- |
| Started | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Starten des FlowListItems aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Started(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Stopped | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Stoppen des FlowListItems aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Stopped(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Paused | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Pausieren des FlowListItems aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Paused(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Finished | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Aufruf der Finish-Funktion des FlowListItems aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Finished(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Start | |
| Funktion | Startet das FlowListItem. Wenn das Item das erste Item der Liste ist oder das Item davor seinen Fokus verlieren kann, wird CurrItem der übergeordneten FlowList auf dieses Item gesetzt, DoStart, die Callback-Methode und das Callback-Event aufgerufen. Sollte eine Exception auftreten, wird die übergeordnete FlowList gestoppt. Ansonsten führt die Methode nichts aus. Wenn die Eigenschaft IsBlocking auf false gesetzt wurde, wird die Start-Methode vom nächsten Item aufgerufen. Die Prüfung auf den Fokus verhindert, dass durch einen manuellen Aufruf der Start-Funktion eines Items der Verlauf der FlowList verändert wird. |
| Signatur | void \_\_fastcall Start(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Stop | |
| Funktion | Stoppt das FlowListItem. Wenn das FlowListItem pausiert wurde, oder gerade ausgeführt wird, ruft die Methode die DoStop-Methode, die Callback-Funktion Stopped und das Callback-Event OnStopped auf.  Wenn das Item den Fokus verlieren kann, wird, wenn dieses Item das Letzte in der FlowList ist, die Finish-Methode der FlowList aufgerufen, sonst wird die Start-Methode des nächsten Items aufgerufen. Die Prüfung auf den Fokus verhindert, dass durch einen manuellen Aufruf der Stop-Funktion eines Items der Verlauf der FlowList verändert wird. Stop sollte in abgeleiteten Klassen nie direkt aufgerufen werden. Stattdessen soll die Finish-Methode verwendet werden. |
| Signatur | void \_\_fastcall Stop(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Pause | |
| Funktion | Pausiert das FlowListItem. Ruft die DoPause-Methode, die Callback-Funktion Paused und das Callback-Event OnPaused auf. |
| Signatur | void \_\_fastcall Pause(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Jump | |
| Funktion | Beim Aufruf dieser Methode soll die Aktion, die vom Item ausgeführt wird, übersprungen werden, damit das Item seine nächste Aktion ausführen kann. Standardmäßig wird beim Aufruf von Jump die Stop-Methode aufgerufen. In TCustomFlowItem wird diese Methode überschrieben, um das aktuelle FlowItem dieses Items zu überspringen. |
| Signatur | virtual void \_\_fastcall Jump(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Finish | |
| Funktion | Beendet das Ausführen des FlowListItems. Ruft die Callback-Methode Finished und das Callback-Event OnFinished auf. Wenn die Eigenschaft Repeat auf True gesetzt wurde, ruft die Methode die DoStop-Methode und danach die DoStart-methode auf, um die Aktion des FlowListItems zu wiederholen. Sollte beim Starten eine Exception auftreten, wird die übergeordnete FlowList gestoppt und die Exception weitergegeben.  Wenn Repeat auf False gesetzt ist, wird lediglich die Stop-Methode des Items aufgerufen.  In abgeleiteten Klassen sollte immer die Finish-Methode aufgerufen werden, wenn die Aktion, die das Item ausführen soll, beendet ist, um das Wiederholen der Aktion zu ermöglichen. |
| Signatur | void \_\_fastcall Finish(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| HasFocus | |
| Funktion | Gibt an, ob das Item den Fokus besitzt. |
| Signatur | bool \_\_fastcall HasFocus(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt true zurück, wenn das Item der Eigenschaft CurrItem der FlowList das Item ist, sonst false. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| CanLooseFocus | |
| Funktion | Gibt an, ob das Item den Fokus verlieren kann / darf. |
| Signatur | bool \_\_fastcall CanLooseFocus(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt true zurück, wenn das Item den Fokus hat, die übergeordnete FlowList ausgeführt wird und das Item asynchron ausgeführt wird (IsBlocking ist false) oder nicht mehr ausgeführt wird, sonst False. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsRunning | |
| Funktion | Diese abstrakte Methode muss in abgeleiteten Klassen überschrieben werden und soll angeben, ob das Item gerade ausgeführt wird. |
| Signatur | virtual bool \_\_fastcall IsRunning(void)=0; |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn das Item gerade ausgeführt wird, sonst False |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsPaused | |
| Funktion | Diese abstrakte Methode muss in abgeleiteten Klassen überschrieben werden und soll angeben, ob das Item pausiert wurde. |
| Signatur | virtual bool \_\_fastcall IsPaused (void)=0; |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn das Item pausiert wurde, sonst False |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoStart | |
| Funktion | Diese abstrakte Methode muss in abgeleiteten Klassen überschrieben werden und soll die Aktion des Items starten. Wenn das Item pausiert wurde, soll diese Methode die Aktion von dem Punkt, an dem das Item pausiert wurde, ausführen. |
| Signatur | virtual void \_\_fastcall DoStart(void)=0; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoStop | |
| Funktion | Diese abstrakte Methode muss in abgeleiteten Klassen überschrieben werden und soll die Aktion des Items stoppen. |
| Signatur | virtual void \_\_fastcall DoStop(void)=0; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoPause | |
| Funktion | Diese abstrakte Methode muss in abgeleiteten Klassen überschrieben werden und soll die Aktion des Items pausieren, sodass beim Aufruf der DoStart-Methode das Item von dem Punkt, wo es pausiert wurde, weiter ausgeführt wird. |
| Signatur | virtual void \_\_fastcall DoPause(void)=0; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| FlowList | |
| Signatur | \_\_property TGameFlowList\* FlowList = {read=GetFlowList}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf die übergeordnete FlowList zurück. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| NextItem | |
| Signatur | \_\_property TGameFlowCustomItem\* NextItem = {read=GetNextItem}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf das nächste Item in der Liste zurück. Wenn dieses Item das Letzte in der Liste ist, wird der Nullzeiger zurückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LastItem | |
| Signatur | \_\_property TGameFlowCustomItem\* LastItem = {read=GetLastItem}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf das vorherige Item in der Liste zurück. Wenn dieses Item das Erste in der Liste ist, wird der Nullzeiger zurückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnStarted | |
| Signatur | \_\_property TNotifyEvent OnStarted = {read=onStarted,write=onStarted}; |
| Sichtbarkeit | protected |
| Lesen | Gibt das Callback-Event zurück, welches beim Starten des FlowListItems ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Starten des FlowListItems ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnStopped | |
| Signatur | \_\_property TNotifyEvent OnStopped = {read=onStopped,write=onStopped}; |
| Sichtbarkeit | protected |
| Lesen | Gibt das Callback-Event zurück, welches beim Stoppen des FlowListItems ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Stoppen des FlowListItems ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnPaused | |
| Signatur | \_\_property TNotifyEvent OnPaused = {read=onPaused,write=onPaused}; |
| Sichtbarkeit | protected |
| Lesen | Gibt das Callback-Event zurück, welches beim Pausieren des FlowListItems ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Pausieren des FlowListItems ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnFinished | |
| Signatur | \_\_property TNotifyEvent OnFinished = {read=onFinished,write=onFinished}; |
| Sichtbarkeit | protected |
| Lesen | Gibt das Callback-Event zurück, welches beim Aufruf der Methode Finish des FlowListItems ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Starten der Aufruf der Methode Finish ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Repeat | |
| Signatur | property bool Repeat = {read=repeat,write=repeat}; |
| Sichtbarkeit | protected |
| Lesen | Gibt True zurück, wenn sich das Item wiederholt, wenn es bei der Ausführung am Ende angelangt ist, sonst False. |
| Schreiben | Wird die Property auf True gesetzt, wiederholt sich das Item, wenn es bei der Ausführung am Ende angelangt ist, sonst nicht. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsBlocking | |
| Signatur | \_\_property bool IsBlocking = {read=isBlocking,write=isBlocking, default=true};\_\_ |
| Sichtbarkeit | protected |
| Lesen | Gibt True zurück, wenn das Item asynchron ausgeführt werden soll, sonst false. Standardmäßig ist die Eigenschaft auf True gesetzt. |
| Schreiben | Wird die Property auf True gesetzt, wird das Item asynchron ausgeführt, d. h. dass das Ausführen des nächsten Items in der Liste nicht nach Beendigung der Ausführung dieses Items erfolgt, sondern direkt beim Start dieses Items. Wenn dieses Item das Letzte in der Liste ist, hat IsBlocking keinen Effekt. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LockScreen | |
| Signatur | \_\_property bool LockScreen = {read=lockScreen,write=lockScreen}; |
| Sichtbarkeit | protected |
| Lesen | Gibt True zurück, wenn das aktuelle GameFrame gelocked werden soll, sonst false. |
| Schreiben | Wird die Property auf True gesetzt, wird das aktuelle GameFrame gelocked, d. h. dass die Benutzereingaben (Maus, Tastatur) im GameFrame ignoriert werden, solange dieses Item ausgeführt wird. Sollte während der Ausführung eine Exception auftreten, oder das Item gestoppt werden, wird das GameFrame Benutzereingaben wieder akzeptieren. |
| Exceptions | -/- |

TGameFlowList

Signatur

class PACKAGE TGameFlowList : public TVectorList, public IGameRunnable

Funktion

TGameFlowList ist die Basisklasse für alle ausführbaren Listen in der Game-Engine. Den Aufbau der Liste kann man sich wie ein Flussdiagramm vorstellen, wobei ein Item ein Element des Flussdiagrammes repräsentiert. Beim Starten der FlowList werden alle Items nacheinander ausgeführt, was man sich wie das Abarbeiten des Flussdiagramms von oben nach unten vorstellen kann. Die Items führen je nach Klassentyp unterschiedliche Aktionen aus. Dabei können auch Items verschiedener Klassen in die Liste eingesetzt werden. Eine FlowList kann gestartet, gestoppt und pausiert werden. Die Bearbeitung der FlowList erfolgt meist zur Design-Zeit in der Entwicklerumgebung, sie kann aber auch zur Laufzeit bearbeitet werden.

Methoden

|  |  |
| --- | --- |
| Started | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Starten der Flowlist aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Started(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Stopped | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Stoppen der Flowlist aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Stopped(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Paused | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Pausieren der Flowlist aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Paused(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Finished | |
| Funktion | Callback-Funktion mit leerem Funktionsrumpf. Wird unmittelbar nach dem Aufruf der Finish-Funktion der Flowlist aufgerufen und kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall Finished(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SetAllowedClasses | |
| Funktion | Fügt die Metaklassen der ClassList hinzu, bei denen der Rückgabewert der Methode IsClassAllowed mit der Metaklasse als Parameter True ist.  Überschreibt die Methode aus der Basisklasse TCustomListItem. |
| Signatur | virtual void \_\_fastcall SetAllowedClasses(TClassList \*ClassList); |
| Sichtbarkeit | protected |
| Parameter | ClassList: Zeiger auf eine Liste von Metaklassen. Nur Objekte von item-Klassen dessen Metaklasse sich in dieser Liste befindet, können der FlowList hinzugefügt werden. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsClassAllowed | |
| Funktion | Prüft, ob die Klasse, die der Parameter Class beschreibt, von der Klasse, die der Rückgabewert von der Methode MinimumClass beschreibt, abgeleitet ist. Die Methode kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual bool \_\_fastcall IsClassAllowed(TClass Class); |
| Sichtbarkeit | protected |
| Parameter | Class: Metaklassenobjekt der zu prüfenden Klasse |
| Rückgabewert | Gibt True zurück, wenn die Klasse, die der Parameter Class beschreibt, in der FlowList vorkommen darf, sonst False. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| MinimumClass | |
| Funktion | Gibt das Metaklassenobjekt zurück, welche die Klasse beschreibt, von der alle Items der Flowlist abgeleitet sein müssen. Die Methode kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual TClass \_\_fastcall MinimumClass(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | Gibt das Metaklassenobjekt zurück, welches die Klasse TCustomFlowListItem beschreibt. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Start | |
| Funktion | Startet die FlowList. Wenn die FlowList pausiert wurde, startet sie an dem Punkt, an dem sie pausiert wurde; wenn die FlowList gestoppt ist oder noch nicht ausgeführt wurde, startet sie von Beginn an. Wird die FlowList gerade ausgeführt, führt die Methode nichts aus.  Ruft die Callback-Funktion Started und das Callback-Event OnStarted auf. |
| Signatur | void \_\_fastcall Start(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | * Gibt EFlowItemError weiter, sollte diese Exception beim Starten eines Flow­Items geworfen werden * Wirft EFlowListError, wenn eine andere Exception beim Starten eines Flow­Items geworfen wird |

|  |  |
| --- | --- |
| Stop | |
| Funktion | Stoppt die FlowList. Wenn die FlowList pausiert wurde oder gerade ausgeführt wird, ruft die Methode von jedem FlowItem der FlowList die Stop-Methode auf. Ruft die Callback-Funktion Stopped und das Callback-Event OnStopped auf. |
| Signatur | void \_\_fastcall Stop(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Pause | |
| Funktion | Pausiert die FlowList und ruft von jedem FlowItem der FlowList die Pause-Methode auf. Ruft die Callback-Funktion Paused und das Callback-Event OnPaused auf. |
| Signatur | void \_\_fastcall Pause(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Finish | |
| Funktion | Stoppt die FlowList. Wenn Repeat auf True gesetzt ist, wird die FlowList von Anfang an wieder ausgeführt. Ruft die Callback-Funktion Finished und das Callback-Event OnFinished auf. |
| Signatur | void \_\_fastcall Pause(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsRunning | |
| Funktion | -/- |
| Signatur | bool \_\_fastcall IsRunning(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn die FlowList ausgeführt wird, sonst False |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsPaused | |
| Funktion | -/- |
| Signatur | bool \_\_fastcall IsPaused (void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn die FlowList pausiert wurde, sonst False |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| Items | |
| Signatur | \_\_property TGameFlowCustomItem\* Items[int Index] = {read=GetItem,write=SetItem}; |
| Sichtbarkeit | public |
| Lesen | Castet den Lesezugriff auf die Property Items der Basisklasse TCustomList auf den Typ TGameFlowCustomItem |
| Schreiben | Castet den Schreibzugriff auf die Property Items der Basisklasse TCustomList auf den Typ TGameFlowCustomItem |
| Exceptions | Siehe Property Items in TCustomList |

|  |  |
| --- | --- |
| CurrItem | |
| Signatur | \_\_property TGameFlowCustomItem\* CurrItem = {read=GetCurrItem}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf das Item in der Liste zurück, welches gerade den Fokus hat. Wenn die Liste gestoppt ist, wird der Nullzeiger zurückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GameObject | |
| Signatur | \_\_property TGameObject\* GameObject = {read=GetGameObject}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf das GameObject zurück, dem diese FlowList gehört oder einen Nullzeiger, wenn die Liste keinen Eigentümer hat. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnStarted | |
| Signatur | \_\_property TNotifyEvent OnStarted = {read=onStarted,write=onStarted}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches beim Starten der FlowList ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Starten der FlowList ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnStopped | |
| Signatur | \_\_property TNotifyEvent OnStopped = {read=onStopped,write=onStopped}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches beim Stoppen der FlowList ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Stoppen der FlowList ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnPaused | |
| Signatur | \_\_property TNotifyEvent OnPaused = {read=onPaused,write=onPaused}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches beim Pausieren der FlowList ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Pausieren der FlowList ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnFinished | |
| Signatur | \_\_property TNotifyEvent OnFinished = {read=onFinished,write=onFinished}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches beim Aufruf der Methode Finish der FlowList ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Starten der Aufruf der Methode Finish ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Repeat | |
| Signatur | \_\_property bool Repeat = {read=repeat,write=repeat}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt True zurück, wenn sich die Liste wiederholt, wenn sie bei der Ausführung am Ende angelangt ist, sonst False. |
| Schreiben | Wird die Property auf True gesetzt, wiederholt sich die Liste, wenn sie bei der Ausführung am Ende angekommen ist, sonst nicht. |
| Exceptions | -/- |

Hallo

* 1. Unit: FlowTree

Diese Unit beinhaltet die Klassen, um einen Dialogbaum darstellen zu können. TFlowTreeItem erbt von TVectorTreeItem und ist der Typ eines Items in der Dialogstruktur, welches seinerseits wieder Items seines Typs beinhalten kann. So können verästelte Dialogbäume mit unterschiedlichen Ergebnissen erstellt werden.

Diese Items haben eine Überschrift (Text) der auf der ConversationGUI als Dialogoption angezeigt wird, eine FlowList (FlowList), die beim Auswählen des Items ausgeführt wird, und zwei Flags – HideItem und Hidden – die steuern, ob die Dialogoption nach der Auswahl deaktiviert wird bzw. ob die Dialogoption auswählbar ist. Wenn das Flag „OpenGUIOnFinish“ auf True gesetzt ist, wird die ConversationGUI nach dem Ende der Ausführung der FlowList wieder geöffnet, damit der Spieler weitere Dialogoptionen auswählen kann, sonst nicht. Die Eigenschaft Hidden wird im Spielstand gespeichert, damit die nach einmaliger Auswahl deaktivierten Dialogoptionen auch über das Spiel hinweg deaktiviert bleiben.

TFlowTreeList erbt von TVectorTree; dessen geerbte Eigenschaft Root ist der Startpunkt des Dialogbaumes. Die FlowTreeItems mit Root als übergeordnetem Element sind die erste Auswahlebene des Dialogbaumes.

* 1. Unit: CustomFlow

Diese Unit beinhaltet die Basisklasse TCustomFlowItem, mit der eigene FlowItems, die vorhandene FlowItems beinhalten, programmiert werden können. Ein Beispiel für ein solches Item ist TConversationFlowItem, welches aus mehreren EventFlowItems und AudioFlowItems besteht.

Um ein solches Item zu erstellen, muss von TCustomFlow eine Klasse abgeleitet werden, die die abstrakte Methode *AddFlowItems* implementiert. Der Parameter AddProc ist ein Methodenzeiger, mit dessen Aufruf das Metaklassenobjekt des FlowItems übergeben muss, was man hinzufügen möchte.

Einfacher ist jedoch der Aufruf der AddItem-Methode, der als Template-Parameter den hinzuzufügenden FlowItem-Typ und als Funktionsparameter den Funktionszeiger AddProc annimmt. Dadurch wird ein FlowItem des übergebenen Typs dem CustomFlow hinzugefügt und dieses Item zurückgegeben, um weitere Einstellungen an dem Item vornehmen zu können.

Die AddItem-Methode kann auch mehrmals aufgerufen werden, um eine Kombination mehrerer vorhandener FlowItems zu einem neuen FlowItem zu ermöglichen.

* 1. Unit: ConversationFlow

In dieser Unit befinden sich die Klassen für ein TConversationFlowItem. Items dieses Typs können den Dialogbeitrag einer Person erstellen. Dazu wird der Name einer Idle-Animation und einer Talk-Animation der Person benötigt, sowie die Person selbst und die Sätze, die die Person sagen soll.

Die Talk-Animation (TalkAnimation) ist die Animation, die der Person (Person) zugewiesen wird, wenn sie redet, also beim Start dieses Items. Die Idle-Animation (IdleAnimation) wird der Person zugewiesen, wenn sie alle ihre Sätze (ConvParts) gesagt hat. Danach ist die Ausführung des Items beendet.

Die Eigenschaft ConvParts ist eine Collection von CollectionItems des Typs TConvPart. Diese haben die Eigenschaften AudioFile und Caption, mit denen Audiodateien und zugehörige Untertitel gesetzt werden können, welche die Person dann nacheinander sagt. Die Untertitel werden als Sprechblase über der Person solange angezeigt, bis die Audiodatei fertig abgespielt ist, oder – wenn keine Audiodatei zugewiesen wurde – ist die Anzeigedauer abhängig von der Länge des Untertitels.

* 1. Unit: AnimationFlow

In dieser Unit befinden sich die Klassen TAnimationBaseFlowItem und TAnimationFlowItem, mit denen sich Bildabfolgen leicht erstellen lassen. In TAnimationBaseFlowItem sind alle Eigenschaften im protected-Bereich deklariert; der Anwender sollte beim Erstellen eigener AnimationFlowItem-Klassen von dieser Klasse ableiten, um flexibel die Eigenschaften der Basisklasse veröffentlichen zu können.

TAnimationFlowItem veröffentlicht alle Eigenschaften der Basisklasse in den \_\_published-Bereich, sodass sie zur Design-Zeit eingestellt werden können.

Die Eigenschaft „Object“ wird auf das GameObject gesetzt, auf dem die Animation durchgeführt werden soll. Die Eigenschaft „FPS“ gibt an, wie viele Bilder pro Sekunde in das GameObject geladen werden sollen. Die Eigenschaft „Animation“ gibt die Animation an, die durchgeführt werden soll. Animationen sind Unterordner im Ressourcenordner des GameObjects in „Object“. Die Bilder (PNG-Format) die in diesen Unterordnern liegen, werden für die Animation verwendet und nacheinander (sortiert nach Namen der Bilder im Ordner) abgespielt.

TAnimationBaseFlowItem ist von TCustomFlowItem abgeleitet und besteht aus mehreren LoadPicFlowItems und SleepFlowItems im Wechsel, je nachdem wie viele Bilder sich im Animationen-Ordner befinden. Durch das Überschreiben der Methoden AddBeforePic und AddAfterPic können vor bzw. nach dem Laden eines neuen Bildes aus der Animation FlowItems hinzugefügt werden, die dann vor bzw. nach dem Laden des Bildes ausgeführt werden.

1. API-Part: Frames

Dieser API-Part beinhaltet (im Unterordner Frames) neben der Basisklasse TGameFrame alle Frames des Spiels. Durch das Erstellen eines von TGameFrame abgeleiteten Frames und dessen Instanziierung im Game-Manager können Orte in der Spielwelt erstellt werden.

|  |  |  |
| --- | --- | --- |
| **Klassenname** | **Unit** | **Funktion** |
| TGameFrame | GameFrame | Basisklasse für alle GameFrames (*siehe TGameFrame*) |

* 1. Unit: GameFrame

TGameFrame

Signatur

class PACKAGE TGameFrame : public TFrame, public IXmlSerializable

Funktion

TGameFrame ist die Basisklasse für alle Frames. Nur auf von GameFrames abgeleiteten Frames können GameObjects (von TGameObject abgeleitete Klassen) platziert werden. Die Klasse implementiert das Interface IXmlSerializable, um den Spielstand in eine XML-Datei speichern zu können.

GameFrames können geöffnet und geschlossen werden. Beim Öffnen werden alle GameObjects gestartet, sodass FlowLists ausgeführt werden können und das GameFrame auf der in SetParent übergebenen Komponente angezeigt. Dabei wird das vorher geöffnete GameFrame geschlossen, sodass immer nur ein GameFrame geöffnet ist. Beim Schließen des Frames wird der Spielstand gespeichert und alle GameObjects gestoppt, sodass keine FlowLists mehr ausgeführt werden.

Die GameFrame-Instanzen werden vom Game-Manager verwaltet. GameFrames werden bei Ihrer Erstellung im Manager registriert und bei ihrer Zerstörung wieder deregistriert. Dabei kann nur eine Instanz pro Frame-Klasse erstellt werden. Der Versuch, eine weitere Instanz zu erzeugen, führt zu einer Exception.

Methoden

|  |  |
| --- | --- |
| DoOpen | |
| Funktion | Callback-Methode mit leerem Funktionsrumpf. Wird ausgeführt, wenn das GameFrame geöffnet wurde. Diese Methode kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall DoOpen(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoClose | |
| Funktion | Callback-Methode mit leerem Funktionsrumpf. Wird ausgeführt, wenn das GameFrame geschlossen wurde. Diese Methode kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall DoClose(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LockedScreenClicked | |
| Funktion | Callback-Methode mit leerem Funktionsrumpf. Wird ausgeführt, wenn der Benutzer einen Mausklick beim gesperrten Frame ausführt. Diese Methode kann in abgeleiteten Klassen überschrieben werden. |
| Signatur | virtual void \_\_fastcall LockedScreenClicked(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ToXmlNode | |
| Funktion | Führt von allen GameObjects die ToXmlNode-Methode aus, um die zu speichernden Spieldaten in einem gemeinsamen Knoten für den Frame zu vereinen. |
| Signatur | \_di\_IXMLNode \_\_fastcall ToXmlNode(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt einen XML-Knoten mit dem zu speichernden Spielstand des Frames zurück. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FromXmlNode | |
| Funktion | Führt von allen GameObjects die FromXmlNode -Methode aus, um die gespeicherten Spieldaten als Unterknoten zu übergeben, sodass die einzelnen GameObjects die Daten lesen können. |
| Signatur | void \_\_fastcall FromXmlNode(\_di\_IXMLNode Node); |
| Sichtbarkeit | public |
| Parameter | Node: Der XML-Knoten, der die Daten für den Frame, sowie Unterknoten für die einzelnen auf dem Frame platzierten GameObjects, beinhaltet. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Player | |
| Funktion | Diese Funktion gibt die Instanz der Spielfigur auf dem Frame zurück. |
| Signatur | TPlayer\* \_\_fastcall Player(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Eine Instanz von TPlayer, die die Spielfigur in dem aktuell geöffneten Frame ist. |
| Exceptions | Wirft eine Exception, sollte keine Spielfigur auf dem Frame gesetzt worden sein. |

|  |  |
| --- | --- |
| IsPlayer | |
| Funktion | Diese Funktion prüft, ob sich in dem Frame eine Spielfigur befindet. |
| Signatur | bool \_\_fastcall IsPlayer(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, sollte sich eine Spielfigur auf dem Frame befinden, sonst False. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LockScreen | |
| Funktion | Diese Methode sperrt den Frame, sodass keine Maus – oder Tastatureingaben vom Benutzer mehr von Objekten im Frame verarbeitet werden. |
| Signatur | TGameFlowCustomItem\* Sender=NULL, float Opacity=0 |
| Sichtbarkeit | public |
| Parameter | Sender: Gibt das FlowItem an, welches den Frame gesperrt hat. Beim Klicken auf den Frame wird die Jump-Methode dieses Items aufgerufen. Dieser Parameter muss nicht zwingend übergeben werden.  Opacity: Gibt die Durchsichtigkeit von 0 bis 1 des schwarzen Sperrbildschirms an (Standardmäßig 0, also durchsichtig). Damit kann das Verdunkeln des Frames beim Sperren realisiert werden. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| UnlockScreen | |
| Funktion | Entsperrt den Frame wieder, sodass Maus- und Tastatureingaben wieder verarbeitet werden. Wenn der Frame nicht gesperrt ist, hat der Aufruf dieser Methode keine Auswirkungen. |
| Signatur | void \_\_fastcall UnlockScreen(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Open | |
| Funktion | Öffnet den Frame. Dabei wird das momentan geöffnete Frame geschlossen, dieses Frame auf der in SetParent übergebenen Komponente angezeigt und der Spielstand für dieses Frame geladen, um danach alle auf dem Frame befindlichen GameObjects zu starten. |
| Signatur | void \_\_fastcall Open(String OpenFlowName); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Instance | |
| Funktion | Gibt die Instanz der vorangestellten Frame-Klasse zurück. |
| Signatur | \_\_classmethod virtual TGameFrame\* \_\_fastcall Instance(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Instanz der vorangestellten Frame -Klasse |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsInstance | |
| Funktion | Prüft, ob eine Instanz der vorangestellten Frame -Klasse im Game-Manager vorliegt. |
| Signatur | \_\_classmethod virtual bool \_\_fastcall IsInstance(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn eine Instanz vorliegt, sonst False |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| IsOpen | |
| Signatur | \_\_property bool IsOpen = {read=GetIsOpen}; |
| Sichtbarkeit | public |
| Lesen | Gibt True zurück, ob das Frame geöffnet ist, sonst False |
| Schreiben | -/- |
| Exceptions | -/- |

1. API-Part: GUIs

Dieser Part der API beinhaltet (im Unterordner GUIs) alle GUIs und deren Oberflächen.

|  |  |  |
| --- | --- | --- |
| **Klassenname** | **Unit** | **Funktion** |
| TGameGUI | GameGUI | Basisklasse aller GUIs (*siehe TGameGUI*) |
| TGameGUIFrame | GameGUIFrame | Basisklasse aller GUI-Oberflächen (*siehe TGameGUIFrame*) |
| TConversationGUI | ConversationGUI | GUI zum Auswählen der Dialogoptionen (*siehe ConversationGUI*) |
| TConversationGUIFrame | ConversationGUIFrame | Oberfläche der TConversationGUI |
| TInventoryGUI | InventoryGUI | GUI für das Inventar des Spielers (*siehe InventoryGUI*) |
| TInventoryGUIFrame | InventoryGUIFrame | Oberfläche der TInventoryGUI |
| TMainMenuGUI | MainMenuGUI | GUI für das Hauptmenü |
| TMainMenuGUIFrame | MainMenuGUIFrame | Oberfläche der TMainMenuGUI |
| TNotesGUI | NotesGUI | GUI für die Notizen. Diese werden im Spielstand gespeichert. |
| TNotesGUIFrame | NotesGUIFrame | Oberfläche der TNotesGUI |

* 1. Unit: GameGUI

TGameGUI

Signatur

class PACKAGE TGameGUI : public TGameBase

Funktion

TGameGUI ist die Basisklasse für alle GUIs in der Game-Engine. Die Klasse implementiert das Öffnen und Schließen der GUI.

Zudem ermöglicht die Klasse ein einfaches Erstellen neuer GUIs zur Design-Zeit, indem im Konstruktor der Basisklasse ein Objekt einer von TGameGUIFrame abgeleiteten Klasse übergeben werden muss. Dieses GUI-Frame kann zur Design-Zeit bearbeitet werden und wird dann über die gesamte Fläche der GUI angezeigt. Abgeleitete Klassen sollten den Standardkonstruktor für Komponenten (nur mit Parameter Owner [Typ TComponent]) implementieren um die GUI in die Komponentenpalette der Entwicklerumgebung installieren zu können.

Die GameGUI-Instanzen werden vom Game-Manager verwaltet. GameGUIs werden zur Laufzeit bei ihrer Erstellung im Manager registriert und bei ihrer Zerstörung wieder deregistriert. Dabei kann nur eine Instanz pro GUI-Klasse erstellt werden. Der Versuch, eine weitere Instanz zu erzeugen, führt zu einer Exception.

Konstruktoren

|  |  |
| --- | --- |
| Owner, GUIFrame | |
| Funktion | Setzt das übergebene GUI-Frame in die Komponente |
| Signatur | \_\_fastcall TGameGUI(TComponent \*Owner, TGameGUIFrame \*GUIFrame); |
| Sichtbarkeit | public |
| Parameter | Owner: Eigentümer der Komponente  GUIFrame: Instanz einer von TGameGUIFrame abgeleiteten Klasse, die auf der Komponente angezeigt wird. Die Freigabe der Instanz erfolgt automatisch im Destruktor der GUI |
| Exceptions | -/- |

Methoden

|  |  |
| --- | --- |
| DoOpen | |
| Funktion | Implementiert das weiche Öffnen der GUI (Transparenz wird schrittweise erhöht).  Methode kann in abgeleiteten Klassen überschrieben werden, um Funktionalität beim Öffnen der GUI hinzuzufügen. |
| Signatur | virtual void \_\_fastcall DoOpen(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoClose | |
| Funktion | Implementiert das weiche Schließen der GUI (Transparenz wird schrittweise erhöht). Methode kann in abgeleiteten Klassen überschrieben werden, um Funktionalität beim Schließen der GUI hinzuzufügen. |
| Signatur | virtual void \_\_fastcall DoClose(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Open | |
| Funktion | Öffnet die GUI. Wenn die GUI schon geöffnet wurde, führt diese Funktion nichts aus. |
| Signatur | void \_\_fastcall Open(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Close | |
| Funktion | Schließt die GUI. Wenn die GUI schon geschlossen wurde, führt diese Funktion nichts aus. |
| Signatur | void \_\_fastcall Close(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Reset | |
| Funktion | Diese Methode wird aufgerufen, wenn ein neues Spiel gestartet wird und die GUIs zurückgesetzt werden müssen. TInventoryGUI überschreibt diese Methode, um bei einem neuen Spiel das Inventar zu leeren. |
| Signatur | virtual void \_\_fastcall Reset(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Instance | |
| Funktion | Gibt die Instanz der vorangestellten GUI-Klasse zurück. |
| Signatur | \_\_classmethod virtual TGameGUI\* \_\_fastcall Instance(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Instanz der vorangestellten GUI-Klasse |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsInstance | |
| Funktion | Prüft, ob eine Instanz der vorangestellten GUI-Klasse im Game-Manager vorliegt. |
| Signatur | \_\_classmethod virtual bool \_\_fastcall IsInstance(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Gibt True zurück, wenn eine Instanz vorliegt, sonst False |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| IsOpen | |
| Signatur | \_\_property bool IsOpen = {read=isOpen,write=SetIsOpen,default=true}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt True zurück, wenn die GUI geöffnet ist, sonst False |
| Schreiben | Öffnet die GUI mit True als Wert, bei False wird die GUI geschlossen |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GUIFrame | |
| Signatur | \_\_property TGameGUIFrame\* GUIFrame = {read=FGUIFrame}; |
| Sichtbarkeit | public |
| Lesen | Gibt das auf der GUI angezeigte Frame zurück. |
| Schreiben | -/- |
| Exceptions | -/- |

* 1. Unit: GameGUIFrame

TGameGUIFrame

Signatur

class PACKAGE TGameGUIFrame : public TFrame

Funktion

TGameGUIFrame ist die Basisklasse für alle GUI-Oberflächen, die auf den zugehörigen GUIs angezeigt werden sollen. Somit kann die GUI-Oberfläche schon zur Design-Zeit erstellt werden. Abgeleitete Klassen können z. B. beim Klicken auf einen Button über die GUIComp-Eigenschaft eine entsprechende Methode der GUI aufrufen, damit diese auf den Button-Klick reagieren kann.

Diese Klasse implementiert das „weiche“ Öffnen und Schließen der GUI durch eine FloatAnimation, die die Transparenz des Frames in 0.2s im fließenden Übergang von transparent auf sichtbar und umgekehrt stellen kann. Genutzt wird diese Funktionalität in der Open- bzw. Close-Methode der übergeordneten GUI.

Konstruktoren

|  |  |
| --- | --- |
| Owner | |
| Funktion | Erstellt das Frame, was auf der übergeordneten GUI angezeigt werden soll |
| Signatur | \_\_fastcall TGameGUIFrame(TComponent\* Owner) |
| Sichtbarkeit | public |
| Parameter | Owner: Muss eine von TGameGUI abgeleitete Klasse sein. |
| Exceptions | Wirft eine EArgumentException, wenn Owner keine von TGameGUI abgeleitete Klasse ist. |

Eigenschaften

|  |  |
| --- | --- |
| GUIComp | |
| Signatur | \_\_property TGameGUI\* GUIComp = {read=GetGUIComp}; |
| Sichtbarkeit | public |
| Lesen | Gibt die übergeordnete GUI-Instanz zurück |
| Schreiben | -/- |
| Exceptions | -/- |

Hallo

* 1. Unit: ConversationGUI

In dieser Unit befinden sich die Klassen für die ConversationGUI. Ein TConversationGUIItem repräsentiert eine Dialogoption auf der ConversationGUI. Diese Items zeigen die Überschriften der zur Auswahl stehenden FlowTreeItems an.

Die Klasse TItemColors fast drei Schriftfarben (Normal, Hovered, Pressed) zu einem Typ zusammen. Normal ist dabei die Schriftfarbe, wenn eine Dialogoption nur angezeigt wird, Hovered die Schriftfarbe, wenn sich die Maus über der Dialogoption befindet, und Pressed, wenn mit der Maus auf diese Dialogoption gedrückt wird.

TConversationGUI ist die Klasse für die ConversationGUI. Mit der Methode LoadFromConvTree können Dialogbäume in die GUI geladen werden, mit SelectConv kann eine bestimmte Dialogoption ausgewählt werden. Die Eigenschaft ConvTreeList gibt den aktuellen Dialogbaum zurück, die Eigenschaft CurrTreeItem das TreeItem, dessen untergeordnete Items gerade zur Dialogauswahl stehen. Nach dem Laden des Dialogbaumes ist diese Eigenschaft das Root-Objekt des Baumes.

Die Eigenschaft ItemColors setzt die Schriftfarben der Dialogoptionen für die Zustände Normal, Hovered und Pressed.

* 1. Unit: InventoryGUI

In dieser Unit befinden sich die Klassen für das InventoryGUI. TInventoryItem erbt von TImageButton und stellt ein Item im Inventar grafisch auf der InventoryGUI dar. Beim Erstellen eines InventoryItems muss das Item vom Typ IGameItemData im Konstruktor übergeben werden. Anhand dessen wird das ItemIcon auf dem InventoryItem angezeigt, hinterlegt von der Normal- / Hovered- / Pressed-Textur, die in der übergeordneten InventoryGUI eingestellt sind.

Die InventoryGUI als Solches ist vom Typ TInventoryGUI. Diese Klasse bietet Methoden zum Hinzufügen und Entfernen von Items aus dem Inventar sowie zum Auswählen eines Items aus dem Inventar, mit dem man mit Hotspots interagieren kann (ItemDropEvents, siehe THotspot).

Welche Items sich im Inventar befinden wird im Spielstand gespeichert, sodass der Inventarinhalt beim Beenden des Spiels erhalten bleibt.

Die Methoden OpenMenu und OpenNodes öffnen das Hauptmenü bzw. die NotesGUI und werden beim Klick auf die entsprechenden Buttons in der Inventarleiste ausgeführt.

Die Eigenschaften ItemNormal, ItemHovered und ItemPressed sind Bitmaps, die vom Anwender zur Design-Zeit gesetzt werden können und die Hintergrundtextur der einzelnen enthaltenen Items je nach Zustand (Normal, Hovered, Pressed) bilden.

Hallo

1. API-Part: Objects

In diesem Teil der API (im Unterordner Objects) sind alle Objekte definiert, die auf GameFrames gesetzt werden müssen. Diese Objekte sind am Ende das, was man grafisch von der Game-Engine im Spiel sieht.

|  |  |  |
| --- | --- | --- |
| **Klassenname** | **Unit** | **Funktion** |
| TGameBase | GameObject | Basisklasse aller Objekte, die auf den Ressourcenordner zugreifen (*siehe TGameBase*) |
| TGameObject | GameObject | Basisklasse aller Objekte, die auf GameFrames platziert werden müssen (*siehe TGameObject*) |
| TForegroundObj | ForegroundObj | Basisklasse aller Objekte mit Animationen (*siehe TForegroundObj*) |
| THotspot | Hotspot | Basisklasse aller im GameFrame anklickbaren Elemente (*siehe THotspot*) |
| TGameItem | Item | Klasse für Items, die im GameFrame platziert sind und aufgesammelt werden können (*siehe Item*) |
| TPerson | Person | Klasse für Personen, mit denen man eine Konversation führen kann. „ConvTree“ stellt den Dialogbaum dar. |
| TPlayer | Player | Klasse für die Spielfigur (*siehe Player*) |
| TBackground | Background | Klasse für den Hintergrund des GameFrames (*siehe Background*) |
| TExit | Exit | Klasse für die Ausgänge, mit denen man andere Frames betreten kann. |

* 1. Unit: GameObject

In dieser Unit sind die Basisklassen für alle sichtbaren Objekte in der Game-Engine (außer der Frames) definiert.

TGameBase

Signatur

class PACKAGE TGameBase : public TControl, public IXmlSerializable

Funktion

TGameBase ist die Basisklasse aller Komponenten der Game-Engine. Sie verwaltet die Ressourcenordner der Komponenten (erstellt Ordner beim Platzieren einer Komponente, löscht Ordner beim Entfernen der Komponente, aktualisiert Ordnernamen beim Ändern des Komponentennamens). Zudem implementiert sie die Basis für die XML-Serialisierung der Objekte sowie eine Methode zum Zurückgeben eines zur Design- und Laufzeit gültigen Pfadnamens des Anwendungsordners. Abgeleitete Klassen müssen die Methode GetResourcePath implementieren, die den Pfad zum Ressourcenordner des Objekts zurückgibt.

Konstruktoren

|  |  |
| --- | --- |
| Owner | |
| Funktion | Standardkonstruktor des FMX-Frameworks. Wenn nicht anders beschrieben, implementiert jede von dieser Klasse abgeleiteten Klasse nur dieses Konstruktor. |
| Signatur | \_\_fastcall TGameObject(TComponent \*Owner); |
| Sichtbarkeit | public |
| Parameter | Owner: Eigentümer der Komponente |
| Exceptions | -/- |

Methoden

|  |  |
| --- | --- |
| GetResourcePath | |
| Funktion | Abstrakte Methode muss in abgeleiteten Klassen überschrieben werden. Sie gibt den Ressourcenpfad zurück, in dem sich z. B. Bilder und Sounddateien für dieses Objekt befinden. Dabei sollte als Basispfad der Rückgabewert von ResourceRoot verwendet werden. Der Ressourcenorder wird automatisch erstellt |
| Signatur | virtual String \_\_fastcall GetResourcePath(void)=0; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | Gibt den Ressourcenpfad für dieses Objekt zurück. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ToXmlNode | |
| Funktion | Implementiert die Methode aus dem IXmlSerializable Interface.  Erstellt einen XML-Knoten mit dem Klassennamen als Knotennamen und gibt diesen zurück. |
| Signatur | TGameXmlNode \_\_fastcall ToXmlNode(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Leerer XML-Knoten mit Klassennamen als Namen. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FromXmlNode | |
| Funktion | Implementiert die Methode aus dem IXmlSerializable Interface. Prüft, ob der Komponentenname mit dem Knotennamen übereinstimmt. |
| Signatur | void \_\_fastcall FromXmlNode(TGameXmlNode Node); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EXmlNodeNameError-Exception, wenn der Komponentenname nicht mit dem Knotennamen übereinstimmt. |

|  |  |
| --- | --- |
| Path | |
| Funktion | Gibt den Pfad des Anwendungsordners zurück. Diese Funktion ist notwendig, da die Working Directory zur Design-Zeit in den Packages nicht der Pfad ist, in dem die Anwendung später erstellt wird. Um einen gültigen Pfad zur Design- und Laufzeit zu erhalten, muss Path verwendet werden. |
| Signatur | String \_\_fastcall Path(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Wenn die Funktion zur Design-Zeit aufgerufen wird, gibt sie den Funktionswert von DesignPath zurück, zur Laufzeit gibt sie den Funktionswert von RunPath zurück. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ResourceRoot | |
| Funktion | Gibt dem Pfad zum Ressourcenordner zurück. |
| Signatur | String \_\_fastcall ResourceRoot(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Setzt sich zusammen aus dem Rückgabewert von Path und dem hartkodierten Namen des Ressourcenordners („Resources“). |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DesignPath | |
| Funktion | Diese statische Funktion gibt den hartkodierten Pfad zum Anwendungsordner zurück. Zur Design-Zeit ist in den Package der Name und Ort der Anwendung unbekannt, daher ist er in dieser Funktion hart kodiert. |
| Signatur | static String \_\_fastcall DesignPath(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | hartkodierter Pfad zum Anwendungsordner |
| Exceptions | -/- |

|  |  |
| --- | --- |
| RunPath | |
| Funktion | Diese statische Funktion gibt den relativen Pfad zum Anwendungsordner zurück. Der Pfad ist aber nur zur Laufzeit gültig, zur design-Zeit muss DesignPath verwendet werden. |
| Signatur | static String \_\_fastcall RunPath(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | relativer Pfad zum Anwendungsordner („./“) |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| ResourcePath | |
| Signatur | \_\_property String ResourcePath = {read=GetResourcePath}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt den Ressourcenpfad zurück |
| Schreiben | -/- |
| Exceptions | -/- |

TGameObject

Signatur

class PACKAGE TGameObject : public TGameBase

Funktion

TGameObject ist die Basisklasse aller Objekte, die auf GameFrames platziert werden. Objekte mit dieser Klasse als Basisklasse werden zur Laufzeit automatisch skaliert, wenn sich die Dimensionen des Objektes, auf dem es platziert ist, ändern (wie bei TScaledLayout). Zudem wird die Möglichkeit, Bilder in die Objekte zu laden sowie Methoden zum Starten und Stoppen der Objekte eingeführt, die u. A. vom übergeordneten GameFrame beim Öffnen bzw. Schließen aufgerufen werden. Die Methode GetResourcePath wurde überschrieben; der Ressourcenpfad der Objekte ist nun Abhängig vom Namen des Objektes und seines übergeordneten Frames.

Konstruktoren

|  |  |
| --- | --- |
| Owner | |
| Funktion | Gibt den Parameter „Owner“ an die Mutterklasse weiter und prüft, ob „Owner“ ein von TGameFrame abgeleitetes Objekt ist. |
| Signatur | \_\_fastcall TGameObject(TComponent \*Owner); |
| Sichtbarkeit | public |
| Parameter | Owner: Eigentümer der Komponente; übergeordnetes GameFrame |
| Exceptions | Wirft eine EArgumentException, wenn Owner nicht von TGameFrame abstammt |

Methoden

|  |  |
| --- | --- |
| DoStart | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Starten des Objektes zu reagieren. |
| Signatur | virtual void \_\_fastcall DoStart(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoStop | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Stoppen des Objektes zu reagieren. |
| Signatur | virtual void \_\_fastcall DoStop(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| PicLoaded | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf den Aufruf der LoadPic-Methode zu reagieren. |
| Signatur | virtual void \_\_fastcall PicLoaded(TImage \*ImgObj); |
| Sichtbarkeit | protected |
| Parameter | ImgObj ist ein Zeiger auf ein TImage-Objekt. Nach dem Laden des Bildes kann so das Bild weiter bearbeitet werden. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FlowListStarted | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Starten einer diesem Objekt untergeordneten FlowList zu reagieren. |
| Signatur | virtual void \_\_fastcall FlowListStarted(TGameFlowList \*List); |
| Sichtbarkeit | protected |
| Parameter | List ist die FlowList, die gestartet wurde |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FlowListStopped | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Stoppen einer diesem Objekt untergeordneten FlowList zu reagieren. |
| Signatur | virtual void \_\_fastcall FlowListStopped (TGameFlowList \*List); |
| Sichtbarkeit | protected |
| Parameter | List ist die FlowList, die gestoppt wurde |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FlowListPaused | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Pausieren einer diesem Objekt untergeordneten FlowList zu reagieren. |
| Signatur | virtual void \_\_fastcall FlowListPaused (TGameFlowList \*List); |
| Sichtbarkeit | protected |
| Parameter | List ist die FlowList, die pausiert wurde |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FlowListFinished | |
| Funktion | Methode mit leerem Funktionsrumpf. Kann in abgeleiteten Klassen überschrieben werden, um auf das Beenden einer diesem Objekt untergeordneten FlowList zu reagieren. |
| Signatur | virtual void \_\_fastcall FlowListFinished (TGameFlowList \*List); |
| Sichtbarkeit | protected |
| Parameter | List ist die FlowList, die beendet (ans Ende ihrer Ausführung angelangt) ist |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SetLeft | |
| Funktion | Setzt die x-Koordinate der Position des Objektes. Diese Methode ersetzt zusammen mit der SetTop-Methode die Eigenschaft Position der TControl-Basisklasse. Durch das Überschreiben der Methode kann direkt auf ein Ändern der Position reagiert werden. |
| Signatur | virtual void \_\_fastcall SetLeft(float Value); |
| Sichtbarkeit | protected |
| Parameter | Value ist der Wert, auf den die x-Koordinate der Position gesetzt werden soll. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SetTop | |
| Funktion | Setzt die y-Koordinate der Position des Objektes. Diese Methode ersetzt zusammen mit der SetLeft-Methode die Eigenschaft Position der TControl-Basisklasse. Durch das Überschreiben der Methode kann direkt auf ein Ändern der Position reagiert werden. |
| Signatur | virtual void \_\_fastcall SetTop(float Value); |
| Sichtbarkeit | protected |
| Parameter | Value ist der Wert, auf den die y-Koordinate der Position gesetzt werden soll. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Start | |
| Funktion | Startet das Objekt. Wenn IsOwnerRunnig true zurückgibt, wird DoStart aufgerufen, ansonsten führt die Funktion nichts aus. |
| Signatur | void \_\_fastcall Start(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Stop | |
| Funktion | Stoppt das Objekt. Ruft die DoStop-Methode und die Stop-Methode aller untergeordneten FlowLists auf. |
| Signatur | void \_\_fastcall Stop(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsRunning | |
| Funktion | Gibt an, ob das Objekt gerade ausgeführt wird. |
| Signatur | bool \_\_fastcall IsRunning(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Wenn das Objekt ausgeführt wird, gibt die Methode True zurück, sonst False. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsOwnerRunning | |
| Funktion | Gibt an, ob der Eigentümer des Objekts gestartet wurde. |
| Signatur | bool \_\_fastcall IsOwnerRunning(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Zur Laufzeit ist der Eigentümer immer ein GameFrame, es wird True zurückgegeben, wenn das übergeordnete GameFrame geöffnet ist. Zur Design-Zeit ist der Eigentümer nur dann ein GameFrame, wenn es nicht im Formular-Designer bearbeitet wird, sondern manuell im Package-Quelltext erstellt wurde. Zur Design-Zeit soll IsOwnerRunning nur true zurückgeben, wenn das übergeordnete Frame kein Nachkomme von TGameFrame ist und das Laden aller Objekte in den Formular-Designer abgeschlossen ist. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LoadPic | |
| Funktion | Lädt ein Bild in das Objekt, sodass das Bild an die Höhe (oder Breite, je nach Einstellung der Eigenschaft FixedSide) des Objekts angepasst wird, und die Breite des Objekts (oder Höhe, je nach Einstellung der Eigenschaft FixedSide) an die Breite (oder Höhe) des Bildes unter Beibehaltung des Seitenverhältnisses des Bildes angepasst wird. |
| Signatur | void \_\_fastcall LoadPic(String FileName); |
| Sichtbarkeit | protected |
| Parameter | FileName gibt den Pfad zur Bilddatei an. |
| Rückgabewert | -/- |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| FixedSide | |
| Signatur | \_\_property TFixedSide FixedSide = {read=FFixedSide,write=FFixedSide}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Seite (Width, Height) zurück, die beim Laden eines Bildes nicht verändert wird, um die Dimensionen des Objekts auf das Bild anzupassen. |
| Schreiben | Setzt die Seite (Width, Height), die beim Laden eines Bildes nicht verändert wird, um die Dimensionen des Objekts auf das Bild anzupassen. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OriginalWidth | |
| Signatur | \_\_property float OriginalWidth = {read=FOriginalWidth, write=SetOriginalWidth}; |
| Sichtbarkeit | \_\_published |
| Lesen | Der Wert der Eigenschaft ist, wenn das Objekt nicht skaliert ist, gleich mit dem Wert der Eigenschaft Width. Durch das Skalieren verändert sich der Wert von Width, aber nicht von OriginalWidth. |
| Schreiben | Durch das Setzten der Eigenschaft OriginalWidth kann der Anwender die Breite setzen, ohne die Skalierung zu berücksichtigen. Dadurch wird der Wert von Width so verändert, dass das Skalierungsverhältnis berücksichtig wird. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OriginalHeight | |
| Signatur | \_\_property float OriginalHeight = {read=FOriginalHeight, write=SetOriginalHeight}; |
| Sichtbarkeit | \_\_published |
| Lesen | Der Wert der Eigenschaft ist, wenn das Objekt nicht skaliert ist, gleich mit dem Wert der Eigenschaft Height. Durch das Skalieren verändert sich der Wert von Height, aber nicht von OriginalHeight. |
| Schreiben | Durch das Setzten der Eigenschaft OriginalHeight kann der Anwender die Breite setzen, ohne die Skalierung zu berücksichtigen. Dadurch wird der Wert von Height so verändert, dass das Skalierungsverhältnis berücksichtig wird. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Left | |
| Signatur | \_\_property float Left = {read=GetLeft,write=SetLeft}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die x-Koordinate der Position des Objekts zurück. |
| Schreiben | Ruft die überschreibbare Methode SetLeft auf. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Top | |
| Signatur | \_\_property float Top = {read=GetTop,write=SetTop}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die y-Koordinate der Position des Objekts zurück. |
| Schreiben | Ruft die überschreibbare Methode SetTop auf. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Background | |
| Signatur | \_\_property TBackground\* Background = {read=GetBackground}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Zeiger auf das Background-Objekt zurück, auf dem dieses Objekt platziert ist. |
| Schreiben | -/- |
| Exceptions | -/- |

* 1. Unit: ForegroundObj

TForegroundObj

Signatur

class PACKAGE TForegroundObj : public TGameObject

Funktion

TForegroundObj implementiert die Funktionen zum Animieren und zum Anzeigen von Text eines Game-Objects. Zur Design-Zeit kann man in der Animations-Liste Animationen erstellen, die ausgeführt werden, wenn sie ausgewählt werden. Eine Animation hat einen Namen, eine FlowList, die beim Auswählen dieser Animation ausgeführt wird, und speichert, wenn das Bool-Flag „SaveStatePos“ der Animation gesetzt ist, bestimmte Eigenschaften, die beim Auswählen dieser Animation wieder auf den gespeicherten Wert gesetzt werden. Die Eigenschaften, die im PropertySet „SavedProperties“ der Animations-Liste enthalten sind, können zur Design-Zeit bearbeitet werden und werden für jede Animation gespeichert. Die Animation kann man auswählen, wenn man die Eigenschaft „Selected“ der Animations-Liste auf den Namen der auszuwählenden Animation setzt.

So kann man z. B. eine Animation erstellen und die Breite des Objektes auf einen kleinen Wert setzen und eine zweite Animation erstellen, bei der das Objekt breiter sein soll. Bei der Auswahl der Animationen wird dann die Breite des Objektes auf den jeweils gespeicherten Wert gesetzt.

Objekte aus dieser Klasse sollten lediglich direkt verwendet werden, wenn man damit animierte Objekte im Hintergrund (z. B. blinkende Lampe) darstellen möchte oder als Objekt im Vordergrund, hinter dem sich die Spielfigur bewegt (z. B. Theke, hinter der sich die Spielfigur bewegt). Für anklickbare Objekte (Items, Personen, etc.) sind die abgeleitete Klasse THotspot und dessen Nachkommen vorgesehen.

Methoden

|  |  |
| --- | --- |
| DisplayCaption | |
| Funktion | Zeigt eine Sprechblase über dem Objekt mit Text an |
| Signatur | void \_\_fastcall DisplayCaption(String Caption); |
| Sichtbarkeit | public |
| Parameter | Caption: Text, der in der Sprechblase angezeigt wird. Die Sprechblase schließt sich, wenn ein leerer String übergeben wird. |
| Rückgabewert | -/- |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| Animations | |
| Signatur | \_\_property TGameAnimationList\* Animations = {read=FAnimations, write=SetAnimations}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Liste der Animationen für dieses Objekt zurück. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LoadFlow | |
| Signatur | \_\_property TEventFlowList\* LoadFlow = {read=FLoadFlow, write=SetLoadFlow}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die EventFlowList zurück, die beim Start des Objekts ausgeführt werden soll. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Animation | |
| Signatur | \_\_property String Animation = {read=GetAnimation,write=SetAnimation,stored=false}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Animation aus der Liste der Animationen an, die gerade ausgeführt wird. |
| Schreiben | Führt die angegebene Animation aus der Liste der Animationen aus. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| AnimationSavedProperties | |
| Signatur | \_\_property TOwnedPropertySet\* AnimationSavedProperties = {read=GetStateSavedProperties,write=SetStateSavedProperties,stored=false}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt eine Liste von Eigenschaften zurück; In der Liste enthaltene Eigenschaften des Objektes werden beim Erstellen einer Animation zur Design-Zeit mit der Animation gespeichert, sodass diese beim Ausführen der Animation auf den gespeicherten Wert gesetzt werden. |
| Schreiben | Kopiert die in der übergebenen Liste enthaltenen Eigenschaften in die Liste des Objektes. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| TextOffsetX | |
| Signatur | \_\_property float TextOffsetX = {read=FTextOffsetX,write=SetTextOffsetX}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Verschiebung der Sprechblase von der Mitte des Objektes aus in x-Richtung zurück. |
| Schreiben | Setzt die Verschiebung der Sprechblase von der Mitte des Objektes aus in x-Richtung |
| Exceptions | -/- |

|  |  |
| --- | --- |
| TextOffsetY | |
| Signatur | \_\_property float TextOffsetY = {read=FTextOffsetY,write=SetTextOffsetY}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Verschiebung der Sprechblase von der oberen Kante des Objektes aus in y-Richtung zurück. |
| Schreiben | Setzt die Verschiebung der Sprechblase von der oberen Kante des Objektes aus in y-Richtung |
| Exceptions | -/- |

|  |  |
| --- | --- |
| CaptionColor | |
| Signatur | \_\_property TColor CaptionColor = {read=GetCaptColor,write=SetCaptColor,stored=false}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die gesetzte Schriftfarbe des Sprechblasentextes zurück. |
| Schreiben | Setzt die Schriftfarbe des Sprechblasentextes. |
| Exceptions | -/- |

* 1. Unit: Hotspot

THotspot

Signatur

class PACKAGE THotspot : public TForegroundObj

Funktion

THotspot ist die Basisklasse für alle anklickbaren Objekte im Frame. Dabei unterscheiden Hotspots zwischen drei verschiedenen Mausklicks. Ein Linksklick ohne ausgewähltes Item aus dem Inventar ist ein Action-Event. Ein Linksklick mit ausgewähltem Item aus dem Inventar ist ein ItemDrop-Event. Ein Rechtsklick ist ein Inspect-Event.

Das Action-Event führt die zum Objekttyp zugehörige Aktion aus. Bei Personen wird ein Gespräch angefangen, Items werden aufgesammelt, Ausgänge werden genutzt um ein anderes Frame zu betreten.

Das Inspect-Event ist dazu da, das Objekt zu beobachten. Der Spieler soll so erfahren, wofür das Objekt eigentlich gut sein soll (z. B. könnte die Spielfigur etwas über das inspizierte Objekt sagen).

Das ItemDrop-Event ist dazu da, um Kombinationen zwischen Items und Hotspots zu ermöglichen. Auf unterschiedliche Items soll das Objekt unterschiedlich reagieren (z. B. Man klickt mit Item A auf eine Person; diese sagt, sie benötigt den Gegenstand nicht. Man klickt mit Item B auf diese Person; diese benötigt Item B und gibt dem Spieler im Gegenzug Item C).

Bei allen Events können die Hotspots unterschiedlich reagieren, je nachdem, im welchem Zustand (State) sie sich befinden.

Man kann zur Design-Zeit States der States-Liste des Hotspots hinzufügen. Der Zustand (Eigenschaft ‚State‘) des Hotspots kann dann immer auf einen dieser States gesetzt werden. Der aktuelle State des Hotspots wird beim Schließen des übergeordneten Frames im Spielstand gespeichert und beim Öffnen des Frames geladen.

States sind dazu da, dass Hotspots zu unterschiedlichen Zeitpunkten unterschiedlich reagieren (z. B. Person ist schlecht gelaunt [State A] und will nicht mit der Spielfigur reden. Man gibt der Person einen Schokorigel, die Laune der Person bessert sich [State B] und die Person redet mit ihrem Gegenüber).

Beim Klick auf den Hotspot wird sich die Spielfigur zunächst zum Hotspot bewegen, bevor das zugehörige Event ausgeführt wird. Dabei lassen sich die Endposition (PlayerOffset) und die Blickrichtung (PlayerFacing) der Spielfigur sowie die Animation der Spielfigur (PlayerAnimation), die nach dem Ankommen an der Endposition ausgeführt werden soll, einstellen.

Zudem implementiert diese Klasse Methoden, die ein Icon in der Mitte des Hotspots anzeigen und wieder verstecken, um dem Spieler zu zeigen, auf welche Elemente im Frame er überhaupt klicken kann.

Methoden

|  |  |
| --- | --- |
| RunEventState | |
| Funktion | Führt das Event (Action, Inspect oder ItemDrop), welches durch einen Mausklick ausgelöst wurde, aus. So kann man das Event verzögert ausführen (z. B. nach einem Linksklick läuft die Spielfigur erst zur Tür, bevor das Event Öffnen ausgeführt wird). Die Methode ist nur für den internen Gebrauch gedacht und sollte vom Anwender nicht aufgerufen werden. |
| Signatur | void \_\_fastcall RunEventState(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ShowHint | |
| Funktion | Zeigt ein Icon in der Mitte des Hotspots an. Beim Druck auf die Leertaste wird diese Methode für alle Hotspots im Frame ausgeführt, um dem Spieler eine Hilfestellung zu geben, wo er was anklicken kann. |
| Signatur | void \_\_fastcall ShowHint(TBitmap \*HintImg); |
| Sichtbarkeit | public |
| Parameter | HintImg: Bitmap, die das Icon enthält, welches angezeigt werden soll |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| HideHint | |
| Funktion | Das Hilfestellungs-Icon wird wieder versteckt (*siehe ShowHint*). Diese Methode wird für alle Hotspots nach dem Loslassen der Leertaste ausgeführt. |
| Signatur | void \_\_fastcall HideHint(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Inspect | |
| Funktion | Callback-Methode für das Inspect-Event mit leerem Funktionsrumpf. Diese Methode kann in abgeleiteten Klassen überschrieben werden, um auf dieses Event zu reagieren. |
| Signatur | virtual void \_\_fastcall Inspect(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Action | |
| Funktion | Callback-Methode für das Action-Event mit leerem Funktionsrumpf. Diese Methode kann in abgeleiteten Klassen überschrieben werden, um auf dieses Event zu reagieren. |
| Signatur | virtual void \_\_fastcall Action(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ItemDrop | |
| Funktion | Callback-Methode für das ItemDrop-Event mit leerem Funktionsrumpf. Diese Methode kann in abgeleiteten Klassen überschrieben werden, um auf dieses Event zu reagieren. |
| Signatur | virtual void \_\_fastcall ItemDrop(void){}; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| StateSelActnFlow | |
| Signatur | \_\_property TGameFlowList\* StateSelActnFlow = {read=GetStateSelActnFlow}; |
| Sichtbarkeit | protected |
| Lesen | Gibt die FlowList des Elements aus der ActionFlows-Liste zurück, dessen Name zum State des Hotspots passt. Wenn kein Name mit dem State übereinstimmt, wird nach Elementen mit dem Namen „#ALLSTATES#“ gesucht. Wird eines gefunden, wird die FlowList dieses Elements, ansonsten der Nullzeiger zurückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| StateSelInspctFlow | |
| Signatur | \_\_property TGameFlowList\* StateSelInspctFlow = {read=GetStateSelInspctFlow}; |
| Sichtbarkeit | protected |
| Lesen | Gibt die FlowList des Elements aus der InspectFlows -Liste zurück, dessen Name zum State des Hotspots passt. Wenn kein Name mit dem State übereinstimmt, wird nach Elementen mit dem Namen „#ALLSTATES#“ gesucht. Wird eines gefunden, wird die FlowList dieses Elements, ansonsten der Nullzeiger zurückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| InspectFlows | |
| Signatur | \_\_property TFlowListGameList\* InspectFlows = {read=FInspectFlows, write=SetInspectFlows}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die InspectFlows-Liste für dieses Objekt zurück. Elemente dieser Liste haben einen Namen und eine FlowList; beim Eintritt des Inspect-Events wird die FlowList des Elements ausgeführt, dessen Name zum State des Hotspots passt (*siehe StateSelInspctFlow)*. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ActionFlows | |
| Signatur | \_\_property TFlowListGameList\* ActionFlows = {read=FActionFlows, write=SetActionFlows}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die ActionFlows -Liste für dieses Objekt zurück. Elemente dieser Liste haben einen Namen und eine FlowList; beim Eintritt des Action-Events wird die FlowList des Elements ausgeführt, dessen Name zum State des Hotspots passt (*siehe StateSelActnFlow)*. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| States | |
| Signatur | \_\_property THotspotStateList\* States = {read=stateList,write=SetStates}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Liste der States für diesen Hotspot zurück. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| State | |
| Signatur | \_\_property String State = {read=GetState,write=SetState,stored=false}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt den Namen das aktuellen States zurück. |
| Schreiben | Setzt den State des Objekts auf den zugewiesenen Namen. Dabei wird die FlowList des zugehörigen Elements in der States-Liste ausgeführt. |
| Exceptions | Wirft eine EArgumentException, wenn der übergebene Name mit keinem Namen der Elemente in der States-Liste übereinstimmt. |

|  |  |
| --- | --- |
| ItemDropEvents | |
| Signatur | \_\_property TItemDropEvents\* ItemDropEvents = {read=FItemDropEvents,write=SetItemDropEvents}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die ItemDropEvents -Liste für dieses Objekt zurück. Elemente dieser Liste haben einen Namen, einen Item-Namen und eine FlowList; beim Eintritt des ItemDrop-Events wird die FlowList des Elements ausgeführt, dessen Name zum State des Hotspots und dessen ItemName zum auf dem Hotspot gedroppten Item passt. |
| Schreiben | Leerer Funktionsrumpf; lediglich implementiert, um Eigenschaft zur Design-Zeit bearbeiten zu können. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| PlayerOffset | |
| Signatur | \_\_property float PlayerOffset = {read=FPlayerOffset, write=FPlayerOffset}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Verschiebung in x-Richtung der Position der Spielfigur an, wenn sie sich nach dem Auslösen eines Events zum Hotspot bewegt. |
| Schreiben | Standardmäßig ist die Position der Spielfigur vor dem Hotspot so eingestellt, dass die Mitte der Spielfigur auf der linken Kante des Hotspots liegt. Durch das Setzen dieser Eigenschaft kann man diese Position nach links (negative Werte) oder nach rechts (positive Werte) verschieben. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| PlayerFacing | |
| Signatur | \_\_property TFacingDirection PlayerFacing = {read=FPlayerFacing, write=FPlayerFacing}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt die Richtung der Spielfigur an, in die sie blicken soll, nachdem sie sich zum Hotspot bewegt hat. |
| Schreiben | Setzt den Wert für die Blickrichtung der Spielfigur, nachdem sie sich zum Hotspot bewegt hat. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| PlayerAnimation | |
| Signatur | \_\_property String PlayerAnimation = {read=FPlayerAnimation, write=FPlayerAnimation}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt den Namen der Animation der Spielfigur an, die ausgeführt werden soll, nachdem sich die Spielfigur zum Hotspot bewegt hat und bevor das Event ausgeführt wird. |
| Schreiben | Setzt den Wert für den Namen der Player-Animation. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnInspect | |
| Signatur | \_\_property TGameEvent OnInspect = {read=onInspect, write=onInspect}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches Inspect-Event ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Inspect-Event ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnAction | |
| Signatur | \_\_property TGameEvent OnAction = {read=onAction, write=onAction}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches Action-Event ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim Action -Event ausgelöst werden soll. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OnItemDrop | |
| Signatur | \_\_property TGameEvent OnItemDrop = {read=onItemDrop, write=onItemDrop}; |
| Sichtbarkeit | \_\_published |
| Lesen | Gibt das Callback-Event zurück, welches ItemDrop-Event ausgelöst werden soll, oder den Nullzeiger, sollte kein Event gesetzt worden sein. |
| Schreiben | Setzt das Callback-Event, welches beim ItemDrop -Event ausgelöst werden soll. |
| Exceptions | -/- |

Hallo

* 1. Unit: Item

In dieser Unit befinden sich die Klassen, die für das Item-System des Adventures benötigt werden. Die Klasse TGameItem ist von THotspot abgeleitet und stellt das Item grafisch in der Spielwelt dar. Diese Item-Hotspots werden bei einem Linksklick nach Ausführung des ActionFlows aufgesammelt und kommen in das Inventar.

Items haben zwei Darstellungsformen: In der Spielwelt und als Icon im Inventar. TGameItem implementiert das Interface IGameItemData, welches die Basisdaten eines Items (Name, Inventar-Textur) kapselt. Zur Laufzeit werden die Items als Typ IGameItemData im Game-Manager registriert.

* 1. Unit: Player

In dieser Unit befindet sich die Player-Klasse. TPlayer ist von TForegroundObj abgeleitet und modelliert die Spielfigur, die vom Spieler gesteuert wird.

Der Name der Bewegungs– und der Ruheanimation muss zur Design-Zeit auf den Namen der entsprechenden Animation in der Animations-Liste gesetzt werden, damit die Spielfigur sich bewegen kann.

Die Eigenschaft YOffsetPerPic gibt die Anzahl der Pixel an, mit der sich die Spielfigur in der Bewegungsanimation pro Bild bewegen soll.

Die Eigenschaften MinPosition und MaxPosition geben die minimale bzw. maximale Position an, an die sich die Spielfigur bewegen kann.

TextureDirection gibt die Blickrichtung des Players in den Texturen an; FacingDirection gibt die Blickrichtung des Players an.

Die Spielfigur bewegt sich, wenn irgendwo auf das Frame geklickt wurde, zu dem Punkt, an dem der Spieler geklickt hat, oder, wenn auf einen Hotspot geklickt wurde, zum Hotspot.

* 1. Unit: Background

In dieser Unit befindet sich die Background-Klasse, die auf jedem GameFrame standardmäßig über das gesamte Frame platziert ist und z. B. das Hintergrundbild des Frames anzeigt.

Des Weiteren ermöglicht TBackground das Erstellen von sog. OpenFlows. Beim Öffnen eines Frames kann mit dem Parameter OpenFlowName angegeben werden, welche FlowList beim Öffnen des Frames ausgeführt werden soll. Dies ist z. B. wichtig beim Modellieren von Räumen mit mehreren Türen. Betritt die Person den Frame über Raum A, dann wir die Tür zu Raum A zugeschlagen. Betritt die Person den Frame über Raum B, dann wir die Tür zu Raum B zugeschlagen. Solche Szenarien können mit OpenFlows modelliert werden.

jghg

1. API-Part: Sonstige

In diesem Part der API befinden sich (ohne Ordnerzuordnung) Klassen, die zwar für die Game-Engine entwickelt wurden, aber auch (mit Ausnahmen des Game-Managers) außerhalb in anderen Projekten verwendet werden können. Daher wird nur auf den Game-Manager detailliert eingegangen.

|  |  |  |
| --- | --- | --- |
| **Klassenname** | **Unit** | **Funktion** |
| TCustomList | CustomList | Basisklasse für heterogene, serialisierbare, im Design-Time-Editor bearbeitbare Listen. Items dieser Liste sind von TCustomListItem abgeleitet. |
| TCustomListItem | CustomList | Item-Basisklasse für TCustomList. |
| TCustomTree | CustomTree | Basisklasse für heterogene, serialisierbare, im Design-Time-Editor bearbeitbare Baumstrukturen. |
| TCustomTreeItem | CustomTree | Item-Basisklasse für TCustomTree. Item kann seinerseits wieder mehrere Items in seiner Liste speichern, so kommt die Baumstruktur zustande. |
| TEventList | EventList | Liste für Callback-Events. Durch Aufruf der CallEvent-Methode können alle Events der Liste gleichzeitig aufgerufen werden. |
| TGameCursor | GameCursor | Singleton, der die Cursor-Icons verwaltet |
| TImageButton | ImageButton | Klasse für Buttons, die mit Bildern vom Anwender designt werden können |
| TPropertyList | PropertyClasses | Klasse, die veröffentlichte Properties eines Objektes als serialisierbare Liste darstellt. |
| TPropertySet | PropertyClasses | Klasse, die Property-Namen als String in einer Liste darstellt. |
| IXmlSerializable | XmlSerialize | Interface für alle Klassen, die als XML-Knoten serialisiert werden können. |

* 1. Unit: GameManager

In dieser Unit befinden sich die Klassen, die für das Verwalten des Spiels, des Spielstandes und der Log-Datei verantwortlich sind.

TLogFile bietet die Funktion, Text in eine Log-Datei, die sich im Spieldatenordner befindet, zu schreiben.

TStringFlags ermöglicht das Setzen und Lesen von Flags im String-Format. Diese Flags können im XML-Format serialisiert werden.

TGameSave vereinfacht den Umgang mit XML-Knoten und das Lesen und Schreiben dieser in bzw. aus einer Datei.

Die Spielstandsdatei und die Logdatei werden im Ordner *%appdata%/Roaming/Resurrection* gespeichert.

TGameManager

Signatur

class PACKAGE TGameManager

Funktion

TGameManager ist die Basisklasse für den GameManager, der den Spielstand, die Frame- und GUI-Instanzen sowie die Liste von Items, die globalen Flags und die Basiseinstellungen für das Spiel verwaltet. Des Weiteren werden im GameManager auf Tastaturereignisse reagiert.

Die Spielstandsdatei ist eine XML-Datei, die durch den GameSave verwaltet wird. Mit dem GameSave kann man XML-Knoten auslesen und überschreiben und neue Knoten erstellen.

Die globalen Flags werden vom GlobalFlags-Objekt verwaltet. Dort kann der Anwender Flags in Form eines Name-Werte Paares erstellen, lesen und schreiben. Die Namen und Werte sind Strings, die in den Spielstand geschrieben und beim Laden des Spiels ausgelesen werden, sodass die Daten erhalten bleiben.

Basiseinstellungen für das Spiel, wie das Festlegen einer GUI als Hauptmenü und eines Frames als Standardframe, werden im Konstruktor übernommen.

Weitere Einstellungen wie das Setzen der den Frames übergeordneten Komponente, auf der diese beim Öffnen angezeigt werden, oder das Bearbeiten des HintIcons sind ebenfalls möglich.

Bei der Programmierung eines Spiels auf dieser Engine muss der Anwender eine von dieser Klasse abgeleiteten Klasse erstellen und die CreateFrames-Methode überschreiben. Zudem kann der Anwender so den GameManager um Funktionalität wie z. B. eine multiple Spielstandsverwaltung erweitern.

Wichtig ist, dass, bevor die GUIs erstellt werden, ein Manager instanziiert wird. Wenn die GUIs zur Design-Zeit im Formular erstellt wurden, muss der Manager in der Main-Funktion der Anwendung vor dem Aufruf der Run-Funktion der Anwendung erfolgen.

Konstruktoren

|  |  |
| --- | --- |
| MainMenuGUI, DefaultFrame | |
| Funktion | Initialisiert MainMenuGUI und DefaultFrame mit den übergebenen Objekten. Setzt den internen Manager-Zeiger auf dieses Objekt. Es kann nur ein Manager (eine Instanz einer von dieser Klasse abgeleiteten Klasse) erzeugt werden. |
| Signatur | \_\_fastcall TGameManager(TClass MainMenuGUI, TClass DefaultFrame); |
| Sichtbarkeit | public |
| Parameter | MainMenuGUI: Metaklasse der GUI, die als Hauptmenü fungieren soll  DefaultFrame: Metaklasse des Frames, was bei einem neuen Spiel als erstes Frame angezeigt werden soll |
| Exceptions | Wirft eine EArgumentException, wenn schon ein Manager erzeugt wurde. |

Methoden

|  |  |
| --- | --- |
| CreateFrames | |
| Funktion | Abstrakte Methode. In abgeleiteten Klassen müssen in dieser Methode alle Frames des Spiels erzeugt werden. |
| Signatur | virtual void \_\_fastcall CreateFrames(void)=0; |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| RegisterItems | |
| Funktion | Methode mit leerem Funktionsrumpf. In abgeleiteten Klassen können hier durch das Erstellen von TGameItemReg-Instanzen Items registriert werden, die nicht als Item-Objekt in der Welt aufgesammelt werden können. |
| Signatur | virtual void \_\_fastcall RegisterItems(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoStartGame | |
| Funktion | Schließt das Hauptmenü und öffnet das Frame, in dem das Spiel beendet wurde. Gibt es noch keinen Spielstand (neues Spiel), wird das Default-Frame geöffnet.  Abgeleitete Klassen können die Methode überschreiben, um auf das Starten des Spiels zu reagieren. |
| Signatur | virtual void \_\_fastcall DoStartGame(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoStopGame | |
| Funktion | Schließt das aktuelle Frame und alle GUIs und öffnet das Hauptmenü.  Abgeleitete Klassen können die Methode überschreiben, um auf das Stoppen des Spiels zu reagieren. |
| Signatur | virtual void \_\_fastcall DoStopGame(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DoNewGame | |
| Funktion | Löscht den aktuellen Spielstand.  Abgeleitete Klassen können die Methode überschreiben, um auf das Erstellen eines neuen Spielstands zu reagieren. |
| Signatur | virtual void \_\_fastcall DoNewGame(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LoadGame | |
| Funktion | Lädt alle Daten (Global Flags, Frames, GUIs) aus dem Spielstand. |
| Signatur | virtual void \_\_fastcall LoadGame(void); |
| Sichtbarkeit | protected |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| KeyDown | |
| Funktion | Diese Methode wird beim Drücken einer Taste der Tastatur während des Spiels aufgerufen. Wenn die Leertaste gedrückt ist, werden die Hinweis-Icons der Hotspots auf dem aktuell geöffneten Frame angezeigt.  Abgeleitete Klassen können diese Methode überschreiben, um auf Tastatureingaben zu reagieren. |
| Signatur | virtual void \_\_fastcall KeyDown(TGameFrame \*Sender, wchar\_t Key); |
| Sichtbarkeit | protected |
| Parameter | Sender: Das GameFrame, in dem ein Key gedrückt wurde  Key: Der gedrückte Buchstabe (als WideChar-Charakter) |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| KeyUp | |
| Funktion | Diese Methode wird beim Loslassen einer Taste der Tastatur während des Spiels aufgerufen. Wenn die Leertaste losgelassen wird, werden die Hinweis-Icons der Hotspots auf dem aktuell geöffneten Frame wieder versteckt.  Abgeleitete Klassen können diese Methode überschreiben, um auf Tastatureingaben zu reagieren. |
| Signatur | virtual void \_\_fastcall KeyUp(TGameFrame \*Sender, wchar\_t Key); |
| Sichtbarkeit | protected |
| Parameter | Sender: Das GameFrame, in dem ein Key losgelassen wurde  Key: Der losgelassene Buchstabe (als WideChar-Charakter) |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SetGameSave | |
| Funktion | Weist dem GameSave des Managers eine neue GameSave-Instanz zu. Damit können z. B. in abgeleiteten Klassen die Funktion für mehrere Spielstände implementiert werden. |
| Signatur | void \_\_fastcall SetGameSave(TGameSave GameSave); |
| Sichtbarkeit | protected |
| Parameter | GameSave: neue Instanz der Klasse TGameSave |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| RegisterFrame | |
| Funktion | Registriert das übergebene Frame. Das Frame wird der internen Frame-Liste hinzugefügt, sodass die Instanz der Frame-Klasse über den Klassennamen erhalten werden kann (Eigenschaft ‚Frames‘). GameFrames rufen bei ihrer Erstellung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall RegisterFrame(TGameFrame \*Frame); |
| Sichtbarkeit | public |
| Parameter | Frame: Ein noch nicht registriertes GameFrame |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn das Frame schon registriert ist. |

|  |  |
| --- | --- |
| UnregisterFrame | |
| Funktion | Deregistriert das übergebene Frame. Das Frame wird aus der internen Frame-Liste gelöscht. GameFrames rufen bei ihrer Zerstörung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall UnregisterFrame(TGameFrame \*Frame); |
| Sichtbarkeit | public |
| Parameter | Frame: Ein registriertes GameFrame |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn das Frame nicht registriert ist. |

|  |  |
| --- | --- |
| RegisterGUI | |
| Funktion | Registriert die übergebene GUI. Die GUI wird der internen GUI-Liste hinzugefügt, sodass die Instanz der GUI-Klasse über den Klassennamen erhalten werden kann (Eigenschaft ‚GUIs‘). GameGUIs rufen bei ihrer Erstellung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall RegisterGUI(TGameGUI \*GUI); |
| Sichtbarkeit | public |
| Parameter | GUI: Eine noch nicht registrierte GameGUI |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn die GUI schon registriert ist. |

|  |  |
| --- | --- |
| UnregisterGUI | |
| Funktion | Deregistriert die übergebene GUI. Die GUI wird aus der internen GUI -Liste gelöscht. Game GUIs rufen bei ihrer Zerstörung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall UnregisterGUI(TGameGUI \*GUI); |
| Sichtbarkeit | public |
| Parameter | GUI: Eine registrierte GameGUI |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn die GUI nicht registriert ist. |

|  |  |
| --- | --- |
| RegisterItem | |
| Funktion | Registriert das übergebene Item. das Item wird der internen Item-Liste hinzugefügt, sodass die ItemData-Instanz über den Itemnamen erhalten werden kann (Eigenschaft ‚Items‘). GameItems rufen bei ihrer Erstellung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall RegisterItem(IGameItemData \*Item); |
| Sichtbarkeit | public |
| Parameter | Item: Ein noch nicht registriertes Item |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn das Item schon registriert ist. |

|  |  |
| --- | --- |
| UnregisterItem | |
| Funktion | Deregistriert das übergebene Item. Das Item wird aus der internen Item-Liste gelöscht. GameItems rufen bei ihrer Zerstörung automatisch diese Methode auf, sodass ein Aufruf durch den Anwender nicht nötig ist. |
| Signatur | void \_\_fastcall UnregisterItem(IGameItemData \*Item); |
| Sichtbarkeit | public |
| Parameter | Item: Ein registriertes Item |
| Rückgabewert | -/- |
| Exceptions | Wirft eine EArgumentException, wenn das Item nicht registriert ist. |

|  |  |
| --- | --- |
| OpenFrame | |
| Funktion | Öffnet ein GameFrame und lädt dessen Daten aus dem Spielstand. |
| Signatur | void \_\_fastcall OpenFrame(String FrameClassName, String OpenFlowName); |
| Sichtbarkeit | public |
| Parameter | FrameClassName: Klassenname des Frames, das geöffnet werden soll.  OpenFlowName: Name der FlowList, welche beim Öffnen und beim Laden des Frames ausgeführt werden sollen. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SaveGame | |
| Funktion | Speichert alle Daten (Frames, GUIs, GlobalFlags) in den Spielstand. Beim Schließen eines Frames wird diese Methode aufgerufen. |
| Signatur | virtual void \_\_fastcall SaveGame(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| StartGame | |
| Funktion | Startet das Spiel. Wenn die Frames noch nicht erstellt sind, wird *CreateFrames* aufgerufen. Alle Daten aus dem Spielstand werden in das Spiel geladen und die Methode *DoStartGame* aufgerufen. Wenn das Spiel schon gestartet ist, führt diese Methode nichts aus. |
| Signatur | void \_\_fastcall StartGame(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| StopGame | |
| Funktion | Beendet das Spiel. Alle Daten aus dem Spiel werden in den Spielstand gespeichert und die Methode *DoStopGame* aufgerufen. Wenn das Spiel nicht gestartet ist, führt diese Methode nichts aus. |
| Signatur | void \_\_fastcall StopGame(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| NewGame | |
| Funktion | Erstellt ein neues Spiel. Ruft die Methode *DoNewGame* und die Methode *StartGame* auf. Wenn das Spiel schon gestartet wurde, führt diese Methode nichts aus. |
| Signatur | void \_\_fastcall NewGame(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LoadFrame | |
| Funktion | Lädt die Daten des übergebenen GameFrames aus dem Spielstand in dieses Frame. |
| Signatur | void \_\_fastcall LoadFrame(TGameFrame \*Frame); |
| Sichtbarkeit | public |
| Parameter | Frame: Das Frame, in das die Daten geladen werden sollen. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SaveFrame | |
| Funktion | Speichert die Daten des übergebenen GameFrames in den Spielstand. |
| Signatur | void \_\_fastcall SaveFrame(TGameFrame \*Frame); |
| Sichtbarkeit | public |
| Parameter | Frame: Das Frame, dessen Daten in den Spielstand gespeichert werden sollen. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LoadGUI | |
| Funktion | Lädt die Daten der übergebenen GUIs aus dem Spielstand in diese GUI. |
| Signatur | void \_\_fastcall LoadGUI(TGameGUI \*GUI); |
| Sichtbarkeit | public |
| Parameter | GUI: Die GUI, in das die Daten geladen werden sollen. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| SaveGUI | |
| Funktion | Speichert die Daten der übergebenen GUIs in den Spielstand. |
| Signatur | void \_\_fastcall SaveGUI(TGameGUI \*GUI); |
| Sichtbarkeit | public |
| Parameter | GUI: Die GUI, dessen Daten in den Spielstand gespeichert werden sollen. |
| Rückgabewert | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Manager | |
| Funktion | Gibt die Manager-Instanz zurück. |
| Signatur | static TGameManager\* \_\_fastcall Manager(void); |
| Sichtbarkeit | public |
| Parameter | -/- |
| Rückgabewert | Die Manager-Instanz (Instanz einer von TGameManager abgeleiteten Klasse) |
| Exceptions | Wirft eine EArgumentException, wenn noch kein Manager instanziiert wurde |

|  |  |
| --- | --- |
| IsCreated | |
| Funktion | Prüft, ob schon ein Manager instanziiert wurde |
| Signatur | static bool \_\_fastcall IsCreated(void); |
| Sichtbarkeit | public |
| Parameter |  |
| Rückgabewert | Gibt True zurück, wenn ein Manager instanziiert wurde, sonst False. |
| Exceptions | -/- |

Eigenschaften

|  |  |
| --- | --- |
| MainMenuGUI | |
| Signatur | \_\_property TClass MainMenuGUI = {read=FMainMenuGUI}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Metaklasse der Hauptmenü-GUI zurück |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| DefaultFrame | |
| Signatur | \_\_property TClass DefaultFrame = {read=FDefaultFrame}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Metaklasse des Frames zurück, welches bei einem neuen Spiel als erstes Frame angezeigt wird |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| IsStarted | |
| Signatur | \_\_property bool IsStarted = {read=FIsStarted}; |
| Sichtbarkeit | public |
| Lesen | Gibt True zurück, wenn das Spiel gestartet ist, sonst False |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OpenedFrameFlagName | |
| Signatur | \_\_property String OpenedFrameFlagName = {read=FOpenedFrameFlagName}; |
| Sichtbarkeit | public |
| Lesen | Gibt den Flag-Namen des GlobalFlags zurück, in dem der Name des zuletzt geöffneten Frames steht. Anhand dieses Flag-Wertes wird beim nächsten Spielstart das zuletzt geöfnete Frame geöffnet. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FrameParent | |
| Signatur | \_\_property TFmxObject \*FrameParent = {read=FFrameParent,write=FFrameParent}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Komponente zurück, auf der die Frames angezeigt werden sollen. |
| Schreiben | Setzt die Komponente, auf der die Frames angezeigt werden sollen. Vor dem Spielstart sollte diese Eigenschaft gesetzt werden, da sonst die Frames nicht angezeigt werden. |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GameSave | |
| Signatur | \_\_property TGameSave GameSave = {read=FGameSave}; |
| Sichtbarkeit | public |
| Lesen | Gibt das Objekt zurück, was die Spielstandsdatei bearbeitet. Mit diesem Objekt können XML-Knoten gelesen und überschrieben werden. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GlobalFlags | |
| Signatur | \_\_property TStringFlags& GlobalFlags = {read=GetGlobalFlags}; |
| Sichtbarkeit | public |
| Lesen | Gibt das Objekt zurück, was die globalen Flags speichert. In diese Flags können vom Anwender Strings gespeichert werden, die in den Spielstand gespeichert werden und beim nächsten Laden des Spiels wieder im GlobalFlags-Objekt vorhanden sind. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| LogFile | |
| Signatur | \_\_property TLogFile& LogFile = {read=GetLogFile}; |
| Sichtbarkeit | public |
| Lesen | Gibt das Objekt zurück, welches die Log-Datei verwaltet. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| HintIcon | |
| Signatur | \_\_property TBitmap\* HintIcon = {read=FHintIcon}; |
| Sichtbarkeit | public |
| Lesen | Gibt das Icon zurück, was in der Mitte der Hotspots beim Drücken der Leertaste angezeigt werden soll. Durch das Aufrufen von z. B. *LoadFromFile* oder *CopyFromBitmap* des Bitmap-Icons kann das Icon gesetzt werden. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| OpenedFrame | |
| Signatur | \_\_property TGameFrame\* OpenedFrame = {read=FOpenedFrame}; |
| Sichtbarkeit | public |
| Lesen | Gibt das aktuell geöffnete Frame zurück. Wenn gerade kein Frame geöffnet ist, wird der Nullzeiger zruückgegeben. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Frames | |
| Signatur | \_\_property TGameFrame\* Frames[String FrameClassNameOrIndex] = {read=GetFrame}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Frame-Instanz entsprechend des Index zurück. |
| Schreiben | -/- |
| Index | FrameClassNameOrIndex: String, der entweder den Klassennamen des Frames, oder eine Zahl als Index der internen Frame-Liste enthält |
| Exceptions | Wirft eine EArgumentException, sollte der Klassenname zu keinem Frame in der Liste passen. |

|  |  |
| --- | --- |
| FrameExists | |
| Signatur | \_\_property bool FrameExists[String FrameClassName] = {read=GetFrameExists}; |
| Sichtbarkeit | public |
| Lesen | Gibt True zurück, sollte ein Frame mit dem im Index übergeben Klassennamen instanziiert sein, sonst False |
| Schreiben | -/- |
| Index | FrameClassName: Name der Frame-Klasse |
| Exceptions | -/- |

|  |  |
| --- | --- |
| FrameCount | |
| Signatur | \_\_property int FrameCount = {read=GetFrameCount}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Anzahl der instanziierten Frames zurück. Mit FrameCount kann man in einer For-Schleife durch alle Frames iterieren. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GUIs | |
| Signatur | \_\_property TGameGUI\* GUIs[String GUIClassNameOrIndex] = {read=GetGUI}; |
| Sichtbarkeit | public |
| Lesen | Gibt die GUI-Instanz entsprechend des Index zurück. |
| Schreiben | -/- |
| Index | GUIClassNameOrIndex: String, der entweder den Klassennamen der GUI, oder eine Zahl als Index der internen GUI-Liste enthält |
| Exceptions | Wirft eine EArgumentException, sollte der Klassenname zu keiner GUI in der Liste passen. |

|  |  |
| --- | --- |
| GUIExists | |
| Signatur | \_\_property bool GUIExists[String GUIClassName] = {read=GetGUIExists}; |
| Sichtbarkeit | public |
| Lesen | Gibt True zurück, sollte eine GUI mit dem im Index übergeben Klassennamen instanziiert sein, sonst False |
| Schreiben | -/- |
| Index | GUIClassName: Name der GUI-Klasse |
| Exceptions | -/- |

|  |  |
| --- | --- |
| GUICount | |
| Signatur | \_\_property int GUICount = {read=GetGUICount}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Anzahl der instanziierten GUIs zurück. Mit GUICount kann man in einer For-Schleife durch alle GUIs iterieren. |
| Schreiben | -/- |
| Exceptions | -/- |

|  |  |
| --- | --- |
| Items | |
| Signatur | \_\_property IGameItemData\* Items[String ItemNameOrIndex] = {read=GetItem}; |
| Sichtbarkeit | public |
| Lesen | Gibt die ItemData-Instanz entsprechend des Index zurück. |
| Schreiben | -/- |
| Index | ItemNameOrIndex: String, der entweder den Itemnamen oder eine Zahl als Index der internen Item-Liste enthält |
| Exceptions | Wirft eine EArgumentException, sollte der Itemname zu keinem Item in der Liste passen. |

|  |  |
| --- | --- |
| ItemExists | |
| Signatur | \_\_property bool ItemExists[String ItemName] = {read=GetItemExists}; |
| Sichtbarkeit | public |
| Lesen | Gibt True zurück, sollte ein Item mit dem im Index übergeben Itemnamen instanziiert sein, sonst False |
| Schreiben | -/- |
| Index | ItemName: Name des Items |
| Exceptions | -/- |

|  |  |
| --- | --- |
| ItemCount | |
| Signatur | \_\_property int ItemCount = {read=GetItemCount}; |
| Sichtbarkeit | public |
| Lesen | Gibt die Anzahl der instanziierten Items zurück. Mit ItemCount kann man in einer For-Schleife durch alle Items iterieren. |
| Schreiben | -/- |
| Exceptions | -/- |