ENTWURFSDOKUMENT

(V. 1.0)

KNOT^3

PSE WS 2013/14

Auftraggeber:
Karlsruher Institut für Technologie
Institut für Betriebs- und Dialogsysteme
Prof. Dr.-Ing. C. Dachsbacher

Betreuer:
Dipl.-Inf. Thorsten Schmidt
Dipl.-Inf. M. Retzlaff

Auftragnehmer:
Tobias Schulz, Maximilian Reuter, Pascal Knodel,
Gerd Augsburg, Christina Erler, Daniel Warzel

17. Dezember 2013

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Knobel- und Konstruktionsspiel Knot^3, welches im Auftrag des IBDS Dachsbacher ausgearbeitet und wie im Pflichtenheft spezifiziert angefertigt wird.

Kapitel 2

Aufbau

2.1 Architektur

Die grundlegende Architektur des Spiels basiert auf der Spielkomponenten-Infrastruktur des XNA-Frameworks, die mit Spielzuständen kombiniert wird. Die abstrakten Klassen GameStateComponent und DrawableGameStateComponent erben von den von XNA bereitgestellten Klassen GameComponent und DrawableGameComponent implementieren zusätzlich die Schnittstelle IGameStateComponent. Sie unterscheiden sich von den XNA-Basisklassen dadurch, dass sie immer eine Referenz auf einen bestimmten Spielzustand halten und nur in Kombination mit diesem zu verwenden sind.

Die Spielzustände erben von der abstrakten Basisklasse GameScreen und halten eine Liste von IGameStateComponent-Objekten. Wird ein Spielzustand aktiviert, indem von einem anderen Spielzustand aus zu ihm gewechselt wird oder indem er der Startzustand ist, dann weist er seine Liste von IGameStateComponent-Objekten dem Components-Attribut der Game-Klasse zu, die von der vom XNA-Framework bereitgestellten abstrakten Klasse Game erbt. So ist zu jedem Zeitpunkt während der Laufzeit des Spiels ein Spielzustand aktiv, der die aktuelle Liste von Spielkomponenten verwaltet.

Die Spielkomponenten, die nicht gezeichnet werden und nur auf Eingaben reagieren, haben nur eine Update()-Methode und erben von GameStateComponent. Dies sind vor allem verschiedene Input-Handler, welche Tastatur- und Mauseingaben verarbeiten und beispielsweise die Kameraposition und das Kameratarget ändern oder Spielobjekte bewegen.

Spielkomponenten, die neben der Update()-Methode auch eine Draw()-Methode besitzen, erben von DrawableGameStateComponent. Dies sind vor allem die Elemente, aus denen die grafische Benutzeroberfläche zusammengesetzt ist, deren abstrakte Basisklasse Widget darstellt. [weitere Erklärungen zu Widgets...]

Alle Spielobjekte implementieren die Schnittstelle IGameObject. Die abstrakte Klasse GameModel repräsentiert dabei ein Spielobjekt, das aus einem 3D-Modell besteht, und hält zu diesem Zweck eine Referenz auf ein Objekt der Klasse Model aus dem XNA-Framework sowie weitere Eigenschaften wie Position, Drehung und Skalierung.

Spielobjekte sind keine Komponenten, sondern werden in einer Spielwelt zusammengefasst, die durch die Klasse World repräsentiert wird. Die Spielwelt ist ein DrawableGameStateComponent und ruft in ihren Update()- und Draw()-Methoden jeweils die dazugehörigen Methoden aller in ihr enthaltenen Spielobjekte auf.

Shadereffekte werden durch die abstrakte Klasse RenderEffect und die von ihr abgeleiteten Klassen gekapselt. Ein RenderEffect enthält ein Rendertarget vom Typ RenderTarget2D als Attribut und im-

plementiert jeweils eine Begin()- und eine End-Methode. In der Methode Begin() wird das aktuell von XNA genutzte Rendertarget auf einem Stack gesichert und das Rendertarget des Effekts wird als aktuelles Rendertarget gesetzt.

Nach dem Aufruf von Begin() werden alle Draw()-Aufrufe von XNA auf dem gesetzten Rendertarget ausgeführt. Es wird also in eine im RenderTarget2D-Objekt enthaltene Bitmap gezeichnet. Dabei wird von den Draw()-Methoden der GameModels die DrawModel(GameModel)-Methode des RenderEffects aufgerufen, der die Modelle mit bestimmten Shadereffekten in die Bitmap zeichnet.

In der End()-Methode wird schließlich das auf dem Stack gesicherte, vorher genutzte Rendertarget wiederhergestellt und das Rendertarget des RenderEffects wird, unter Umständen verändert durch Post-Processing-Effekte, auf dieses übergeordnete Rendertarget gezeichnet.

2.2 Klassendiagramm

2.3 Verwendete Entwurfsmuster

Kapitel 3

Klassenübersicht

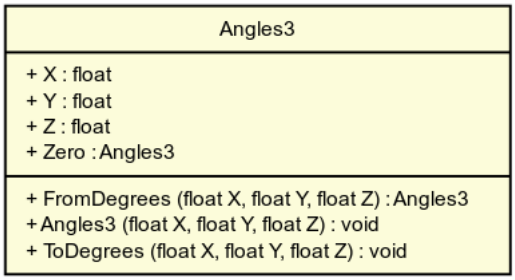
3.1 Klassen

3.1.1 Klasse Angles3

Beschreibung:

Diese Klasse repräsentiert die Rotationswinkel der drei Achsen.

Eigenschaften:



public float X

Der Rotationswinkel um die X-Achse.

public float Y

Der Rotationswinkel um die Y-Achse.

public float Z

Der Rotationswinkel um die Z-Achse.

public Angles3 Zero

Eine statische Property mit dem Wert X = 0, Y = 0, Z = 0.

Konstruktoren:

public Angles3 (float X, float Y, float Z)

Konstruiert ein neues Angles3-Objekt mit drei gegebenen Winkeln.

Methoden:

public Angles3 FromDegrees (float X, float Y, float Z)

Konvertiert Grad in Bogenmaß.

public void ToDegrees (float X, float Y, float Z)

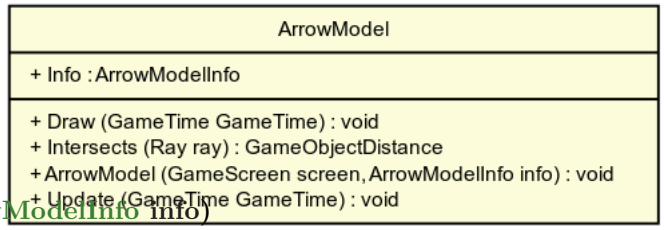
Konvertiert Bogenmaß in Grad.

3.1.2 Klasse ArrowModel

Beschreibung:

Diese Klasse repräsentiert ein 3D-Modell für einen Pfeil, der an selektierten Kanten erscheinen soll.

Eigenschaften:



public ArrowModelInfo Info

Das Info-Objekt, das die Position und Richtung des Pfeils enthält.

Konstruktoren:

public ArrowModel (GameScreen screen, ArrowModelInfo info)

Erstellt ein neues Pfeilmodell in dem angegebenen GameScreen mit einem bestimmten Info-Objekt, das Position und Richtung des Pfeils festlegt.

Methoden:

public void Draw (GameTime GameTime)

Zeichnet den Pfeil.

public GameObjectDistance Intersects (Ray ray)

Überprüft, ob der Mausstrahl den Pfeil schneidet.

public void Update (GameTime GameTime)

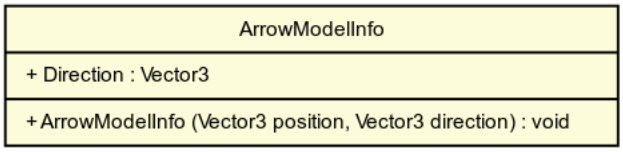
Wird für jeden Frame aufgerufen.

3.1.3 Klasse ArrowModelInfo

Beschreibung:

Ein Objekt dieser Klasse hält alle Informationen, die zur Erstellung eines Pfeil-3D-Modelles (ArrowModel) notwendig sind.

Eigenschaften:



public Vector3 Direction

Die Richtung, die der Pfeil zeigen soll.

Konstruktoren:

public ArrowModelInfo (Vector3 position, Vector3 direction)

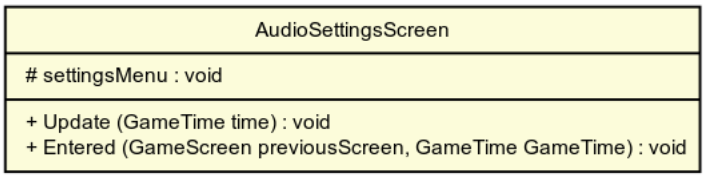
Erstellt ein neues ArrowModelInfo-Objekt an einer bestimmten Position im 3D-Raum, das in eine bestimmte Richtung zeigt.

3.1.4 Klasse AudioSettingsScreen

Beschreibung:

Der Spielzustand, der die Audio-Einstellungen darstellt.

Eigenschaften:



protected void settingsMenu

Das Menü, das die Einstellungen enthält.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

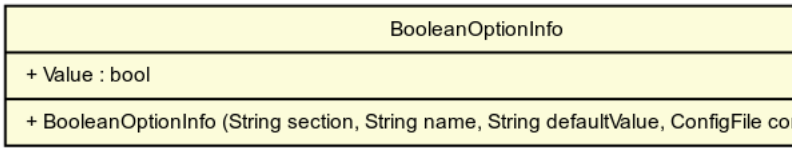
Fügt das Menü mit den Einstellungen in die Spielkomponentenliste ein.

3.1.5 Klasse BooleanOptionInfo

Beschreibung:

Diese Klasse repräsentiert eine Option, welche die Werte „Wahr“ oder „Falsch“ annehmen kann.

Eigenschaften:



public bool Value

Ein Property, das den aktuell abgespeicherten Wert zurückgibt.

Konstruktoren:

public BooleanOptionInfo (String section, String name, String defaultValue, ConfigFile configFile)

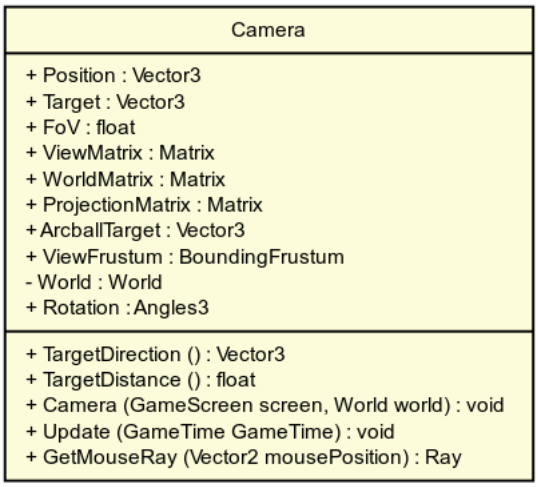
Erstellt eine neue Option, welche die Werte „Wahr“ oder „Falsch“ annehmen kann. Mit dem angegebenen Namen, in dem angegebenen Abschnitt der angegebenen Einstellungsdatei.

3.1.6 Klasse Camera

Beschreibung:

Jede Instanz der World-Klasse hält eine für diese Spielwelt verwendete Kamera als Attribut. Die Hauptfunktion der Kamera-Klasse ist das Berechnen der drei Matrizen, die für die Positionierung und Skalierung von 3D-Objekten in einer bestimmten Spielwelt benötigt werden, der View-, World- und Projection-Matrix. Um diese Matrizen zu berechnen, benötigt die Kamera unter Anderem Informationen über die aktuelle Kamera-Position, das aktuelle Kamera-Ziel und das Field of View.

Eigenschaften:



public Vector3 Position

Die Position der Kamera.

public Vector3 Target

Das Ziel der Kamera.

public float FoV

Das Sichtfeld.

public Matrix ViewMatrix

Die View-Matrix wird über die statische Methode CreateLookAt der Klasse Matrix des XNA-Frameworks mit Matrix.CreateLookAt (Position, Target, Vector3.Up) berechnet.

public Matrix WorldMatrix

Die World-Matrix wird mit Matrix.CreateFromYawPitchRoll und den drei Rotationswinkeln berechnet.

public Matrix ProjectionMatrix

Die Projektionsmatrix wird über die statische XNA-Methode Matrix.CreatePerspectiveFieldOfView berechnet.

public Vector3 ArcballTarget

Eine Position, um die rotiert werden soll, wenn der User die rechte Maustaste gedrückt hält und die Maus bewegt.

public BoundingFrustum ViewFrustum

Berechnet ein Bounding-Frustum, das benötigt wird, um festzustellen, ob ein 3D-Objekt sich im Blickfeld des Spielers befindet.

private World World

Eine Referenz auf die Spielwelt, für welche die Kamera zuständig ist.

public Angles3 Rotation

Die Rotationswinkel.

Konstruktoren:

public Camera (GameScreen screen, World world)

Erstellt eine neue Kamera in einem bestimmten GameScreen für eine bestimmte Spielwelt.

Methoden:

public Vector3 TargetDirection ()

Die Blickrichtung.

public float TargetDistance ()

Der Abstand zwischen der Kamera und dem Kamera-Ziel.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

public Ray GetMouseRay (Vector2 mousePosition)

Berechnet einen Strahl für die angegebenene 2D-Mausposition.

3.1.7 Klasse CelShadingEffect

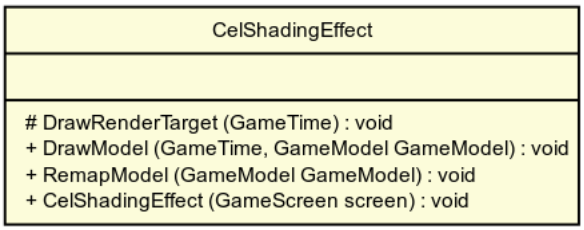
Beschreibung:

Ein Cel-Shading-Effekt.

Konstruktoren:

public CelShadingEffect (GameScreen screen)

Erstellt einen neuen Cel-Shading-Effekt für den angegebenen GameScreen.



Methoden:

protected void DrawRenderTarget (GameTime GameTime)

public void DrawModel (GameTime, GameModel GameModel)

Zeichnet ein 3D-Modell auf das Rendertarget.

public void RemapModel (GameModel GameModel)

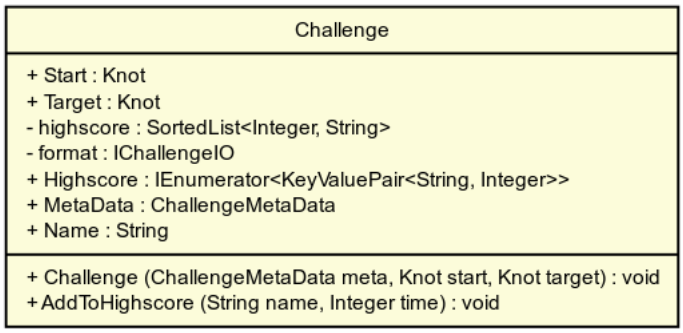
Weist dem 3D-Modell den Cel-Shader zu.

3.1.8 Klasse Challenge

Beschreibung:

Ein Objekt dieser Klasse repräsentiert eine Challenge.

Eigenschaften:



public Knot Start

Der Ausgangsknoten, den der Spieler in den Referenzknoten transformiert.

public Knot Target

Der Referenzknoten, in den der Spieler den Ausgangsknoten transformiert.

private SortedList<Integer, String> highscore

Eine sortierte Bestenliste.

private IChallengeIO format

Das Speicherformat der Challenge.

public IEnumerator<KeyValuePair<String, Integer>> Highscore

Ein öffentlicher Enumerator, der die Bestenliste unabhängig von der darunterliegenden Datenstruktur zugänglich macht.

public ChallengeMetaData MetaData

Die Metadaten der Challenge.

public String Name

Der Name der Challenge.

Konstruktoren:

public Challenge (ChallengeMetaData meta, Knot start, Knot target)

Erstellt ein Challenge-Objekt aus einem gegebenenen Challenge-Metadaten-Objekt. Erstellt ein Challenge-Objekt aus einer gegebenenen Challenge-Datei.

Methoden:

public void AddToHighscore (String name, Integer time)

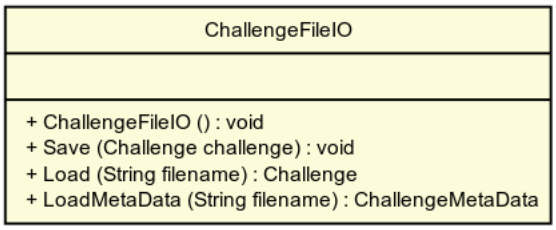
Fügt eine neue Bestzeit eines bestimmten Spielers in die Bestenliste ein.

3.1.9 Klasse ChallengeFileIO

Beschreibung:

Implementiert das Speicherformat für Challenges.

Konstruktoren:



public ChallengeFileIO ()

Erstellt ein ChallengeFileIO-Objekt.

Methoden:

public void Save (Challenge challenge)

Speichert eine Challenge in dem Dateinamen, der in dem Challenge-Objekt enthalten ist.

public Challenge Load (String filename)

Lädt eine Challenge aus einer angebenenen Datei.

public ChallengeMetaData LoadMetaData (String filename)

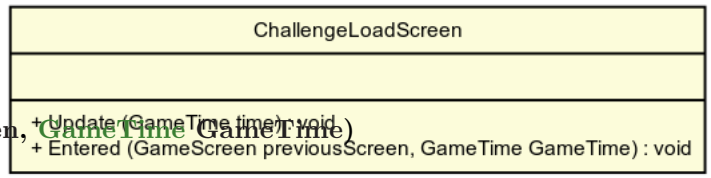
Lädt die Metadaten einer Challenge aus einer angegebenen Datei.

3.1.10 Klasse ChallengeLoadScreen

Beschreibung:

Der Spielzustand, der den Ladebildschirm für Challenges darstellt.

Methoden:



public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt das Menü mit den Spielständen in die Spielkomponentenliste ein.

3.1.11 Klasse ChallengeMetaData

Beschreibung:

Enthält Metadaten zu einer Challenge.

Eigenschaften:

public String Name

Der Name der Challenge.

public KnotMetaData Start

Der Ausgangsknoten, den der Spieler in den Referenzknoten transformiert.

public KnotMetaData Target

Der Referenzknoten, in den der Spieler den Ausgangsknoten transformiert.

public IChallengeIO Format

Das Format, aus dem die Metadaten der Challenge gelesen wurden oder null.

public String Filename

Der Dateiname, aus dem die Metadaten der Challenge gelesen wurden oder in den sie abgespeichert werden.

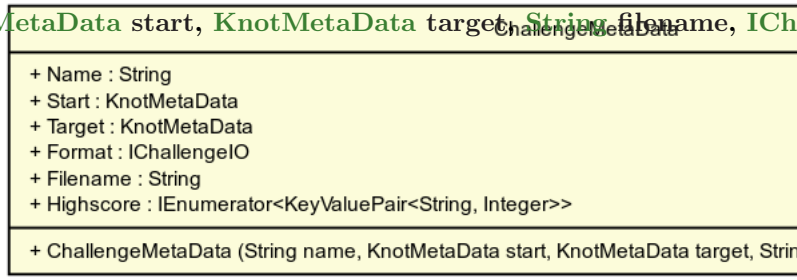
public IEnumerator<KeyValuePair<String, Integer>> Highscore

Ein öffentlicher Enumerator, der die Bestenliste unabhängig von der darunterliegenden Datenstruktur zugänglich macht.

Konstruktoren:

public ChallengeMetaData (String name, KnotMetaData start, KnotMetaData target, String filename, ICh

Erstellt ein Challenge-Metadaten-Objekt mit einem gegebenen Namen und den Metadaten des Ausgangs- und Referenzknotens.



3.1.12 Klasse ChallengeModeScreen

Beschreibung:

Der Spielzustand, der während dem Spielen einer Challenge aktiv ist und für den Ausgangs- und Referenzknoten je eine 3D-Welt zeichnet.

Eigenschaften:

public void PlayerKnot

Der Spielerknoten, der durch die Transformation des Spielers aus dem Ausgangsknoten entsteht.

public void ChallengeKnot

Der Referenzknoten.

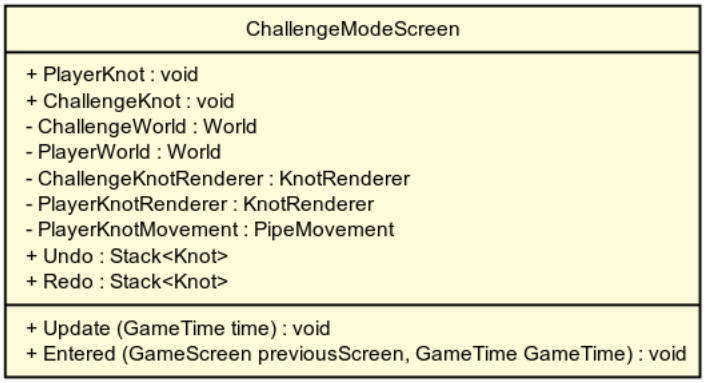
private World ChallengeWorld

Die Spielwelt in der die 3D-Modelle des dargestellten Referenzknotens enthalten sind.

private World PlayerWorld

Die Spielwelt in der die 3D-Modelle des dargestellten Spielerknotens enthalten sind.

private KnotRenderer ChallengeKnotRenderer



Der Controller, der aus dem Referenzknoten die 3D-Modelle erstellt.

private KnotRenderer PlayerKnotRenderer

Der Controller, der aus dem Spielerknoten die 3D-Modelle erstellt.

private PipeMovement PlayerKnotMovement

Der Inputhandler, der die Kantenverschiebungen des Spielerknotens durchführt.

public Stack<Knot> Undo

Der Undo-Stack.

public Stack<Knot> Redo

Der Redo-Stack.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt die 3D-Welten und den Inputhandler in die Spielkomponentenliste ein.

3.1.13 Klasse CheckBoxItem

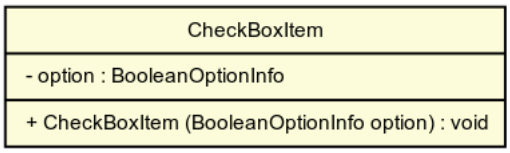
Beschreibung:

Ein Menüeintrag, der einen Auswahlkasten darstellt.

Eigenschaften:

private BooleanOptionInfo option

Die Option, die mit dem Auswahlkasten verknüft ist.



Konstruktoren:

public CheckBoxItem (BooleanOptionInfo option)

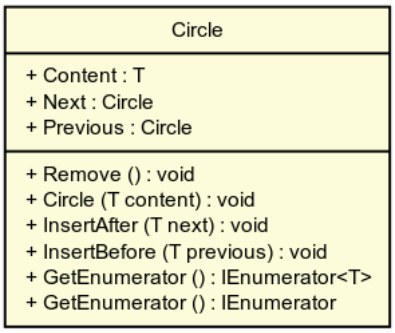
Erstellt einen Menüeintrag für die angegebene Option.

3.1.14 Klasse Circle

Beschreibung:

Eine doppelt verkettete Liste.

Eigenschaften:



public T Content

Der Wert dieses Listeneintrags.

public Circle Next

Der nächste Listeneintrag.

public Circle Previous

Der vorherige Listeneintrag.

Konstruktoren:

public Circle (T content)

Erstellt einen neuen Listeneintrag.

Methoden:

public void Remove ()

Entfernt diesen Listeneintrag und verknüft den vorherigen mit dem nächsten Eintrag.

public void InsertAfter (T next)

Fügt nach diesem Listeneintrag einen neuen Listeneintrag ein.

public void InsertBefore (T previous)

Fügt vor diesem Listeneintrag einen neuen Listeneintrag ein.

public IEnumerator<T> GetEnumerator ()

Gibt einen Enumerator über die Liste zurück.

public IEnumerator GetEnumerator ()

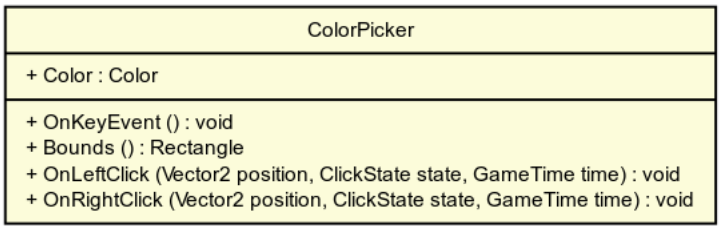
Gibt einen Enumerator über die Liste zurück.

3.1.15 Klasse ColorPicker

Beschreibung:

Ein Steuerelement der grafischen Benutzeroberfläche, das eine Auswahl von Farben ermöglicht.

Eigenschaften:



public Color Color

Die ausgewählte Farbe.

Methoden:

public void OnKeyEvent ()

Reagiert auf Tastatureingaben.

public Rectangle Bounds ()

Gibt die Ausmaße des ColorPickers zurück.

public void OnLeftClick (Vector2 position, ClickState state, GameTime time)

Bei einem Linksklick wird eine Farbe ausgewählt und im Attribut Color abgespeichert.

public void OnRightClick (Vector2 position, ClickState state, GameTime time)

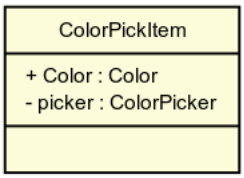
Bei einem Rechtsklick geschieht nichts.

3.1.16 Klasse ColorPickItem

Beschreibung:

Ein Menüeintrag, der eine aktuelle Farbe anzeigt und zum Ändern der Farbe per Mausklick einen ColorPicker öffnet.

Eigenschaften:



public Color Color

Die aktuelle Farbe.

private ColorPicker picker

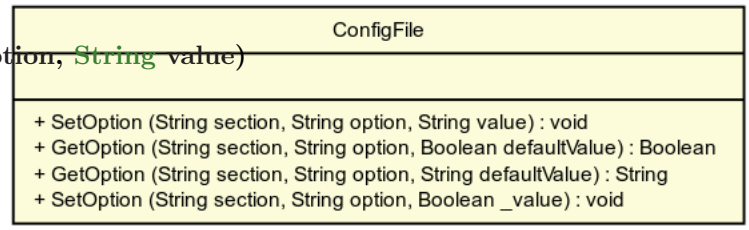
Der ColorPicker, der bei einem Mausklick auf den Menüeintrag geöffnet wird.

3.1.17 Klasse ConfigFile

Beschreibung:

Repräsentiert eine Einstellungsdatei.

Methoden:



public void SetOption (String section, String option, String value)

Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen Abschnitt auf den angegebenen Wert.

public Boolean GetOption (String section, String option, Boolean defaultValue)

Gibt den aktuell in der Datei vorhandenen Wert für die angegebene Option in dem angegebenen Abschnitt zurück.

public String GetOption (String section, String option, String defaultValue)

Gibt den aktuell in der Datei vorhandenen Wert für die angegebene Option in dem angegebenen Abschnitt zurück.

public void SetOption (String section, String option, Boolean value)

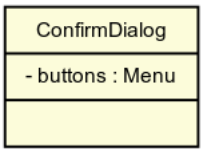
Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen Abschnitt auf den angegebenen Wert.

3.1.18 Klasse ConfirmDialog

Beschreibung:

Ein Dialog, der Schaltflächen zum Bestätigen einer Aktion anzeigt.

Eigenschaften:



private Menu buttons

Das Menü, das Schaltflächen enthält.

3.1.19 Klasse ControlSettingsScreen

Beschreibung:

Der Spielzustand, der die Steuerungs-Einstellungen darstellt.

Eigenschaften:

protected void settingsMenu

Das Menü, das die Einstellungen enthält.

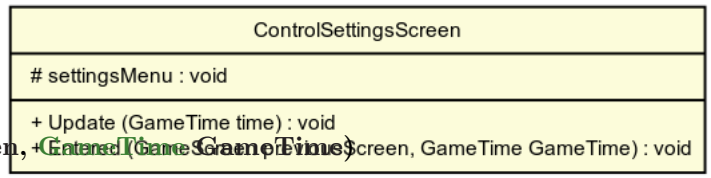
Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt das Menü mit den Einstellungen in die Spielkomponentenliste ein.



3.1.20 Klasse CreativeLoadScreen

Beschreibung:

Der Spielzustand, der den Ladebildschirm für Knoten darstellt.

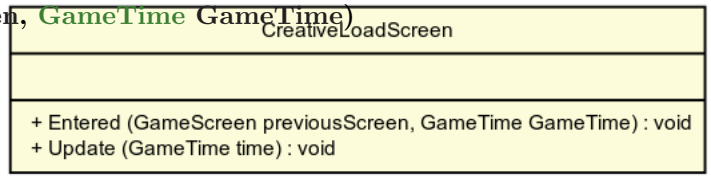
Methoden:

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt das Menü mit dem Spielständen in die Spielkomponentenliste ein.

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.



3.1.21 Klasse CreativeModeScreen

Beschreibung:

Der Spielzustand, der während dem Erstellen und Bearbeiten eines Knotens aktiv ist und für den Knoten eine 3D-Welt zeichnet.

Eigenschaften:

public void Knot

Der Knoten, der vom Spieler bearbeitet wird.

private World World

Die Spielwelt in der die 3D-Objekte des dargestellten Knotens enthalten sind.

private KnotRenderer KnotRenderer

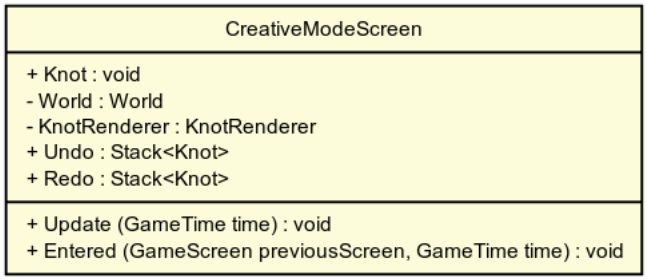
Der Controller, der aus dem Knoten die 3D-Modelle erstellt.

public Stack<Knot> Undo

Der Undo-Stack.

public Stack<Knot> Redo

Der Redo-Stack.



Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime time)

Fügt die 3D-Welt und den Inputhandler in die Spielkomponentenliste ein.

3.1.22 Klasse CreditsScreen

Beschreibung:

Der Spielzustand, der die Auflistung der Mitwirkenden darstellt.

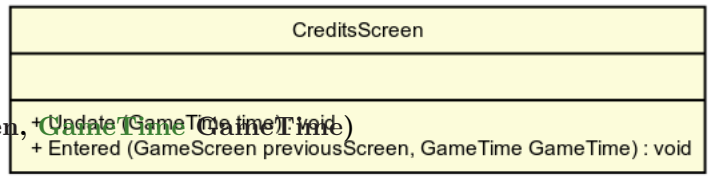
Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt das Menü mit den Mitwirkenden in die Spielkomponentenliste ein.



3.1.23 Klasse Dialog

Beschreibung:

Ein Dialog ist ein im Vordergrund erscheinendes Fenster, das auf Nutzerinteraktionen wartet.

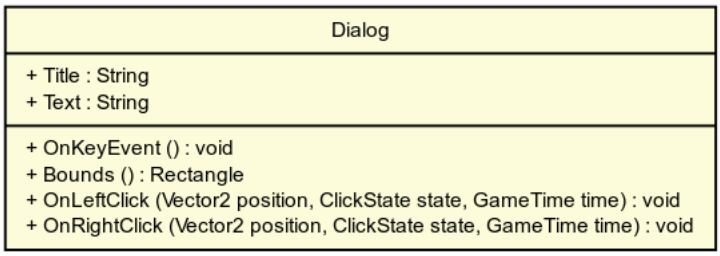
Eigenschaften:

public String Title

Der Fenstertitel.

public String Text

Der angezeigte Text.



Methoden:

public void OnKeyEvent ()

Durch Drücken der Entertaste wird die ausgewählte Aktion ausgeführt. Durch Drücken der Escape-Taste wird der Dialog abgebrochen. Mit Hilfe der Pfeiltasten kann zwischen den Aktionen gewechselt werden.

public Rectangle Bounds ()

Gibt die Ausmaße des Dialogs zurück.

public void OnLeftClick (Vector2 position, ClickState state, GameTime time)

Bei einem Linksklick geschieht nichts.

public void OnRightClick (Vector2 position, ClickState state, GameTime time)

Bei einem Rechtsklick geschieht nichts.

3.1.24 Klasse DistinctOptionInfo

Beschreibung:

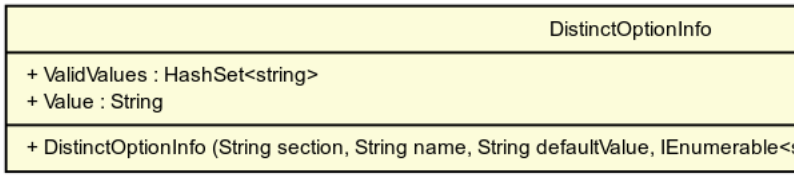
Diese Klasse repräsentiert eine Option, die eine distinkte Werteliste annehmen kann.

Eigenschaften:

public HashSet<string> ValidValues

public String Value

Ein Property, das den aktuell abgespeicherten Wert zurück gibt.



Konstruktoren:

public DistinctOptionInfo (String section, String name, String defaultValue, IEnumerable<string> validVal

Erstellt eine neue Option, die einen der angegebenen gültigen Werte annehmen kann, mit dem angegebenen Namen in dem angegebenen Abschnitt der angegebenen Einstellungsdatei.

3.1.25 Klasse DrawableGameScreenComponent

Beschreibung:

Eine zeichenbare Spielkomponente, die in einem angegebenen Spielzustand verwendet wird und eine bestimmte Priorität hat.

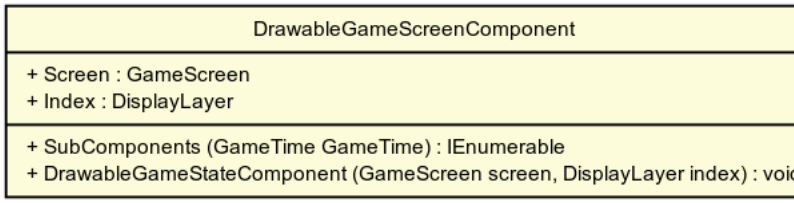
Eigenschaften:

public GameScreen Screen

Der zugewiesene Spielzustand.

public DisplayLayer Index

Die Zeichen- und Eingabepriorität.



Methoden:

public IEnumerable SubComponents (GameTime GameTime)

Gibt Spielkomponenten zurück, die in dieser Spielkomponente enthalten sind.

public void DrawableGameStateComponent (GameScreen screen, DisplayLayer index)

Erstellt eine neue zeichenbare Spielkomponente in dem angegebenen Spielzustand mit der angegebenen Priorität.

3.1.26 Klasse DropDownEntry

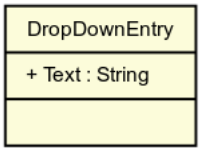
Beschreibung:

Repräsentiert einen Eintrag in einem Dropdown-Menü.

Eigenschaften:

public String Text

Der Text des Eintrags.



3.1.27 Klasse DropDownMenuItem

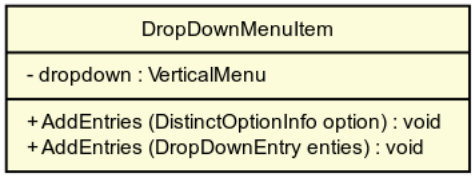
Beschreibung:

Ein Menüeintrag, der den ausgewählten Wert anzeigt und bei einem Linksklick ein Dropdown-Menü zur Auswahl eines neuen Wertes ein- oder ausblendet.

Eigenschaften:

private VerticalMenu dropdown

Das Dropdown-Menü, das ein- und ausgeblendet werden kann.



Methoden:

public void AddEntries (DistinctOptionInfo option)

Fügt Einträge in das Dropdown-Menü ein, die auf Einstellungsoptionen basieren. Fügt Einträge in das Dropdown-Menü ein, die nicht auf Einstellungsoptionen basieren.

public void AddEntries (DropDownEntry enties)

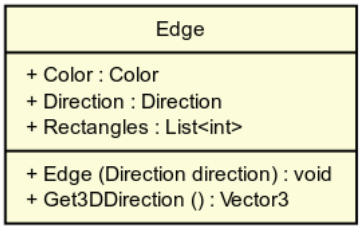
Fügt Einträge in das Dropdown-Menü ein, die auf Einstellungsoptionen basieren. Fügt Einträge in das Dropdown-Menü ein, die nicht auf Einstellungsoptionen basieren.

3.1.28 Klasse Edge

Beschreibung:

Eine Kante eines Knotens, die aus einer Richtung und einer Farbe, sowie optional einer Liste von Flächennummern besteht.

Eigenschaften:



public Color Color

Die Farbe der Kante.

public Direction Direction

Die Richtung der Kante.

public List<int> Rectangles

Die Liste der Flächennummern, die an die Kante angrenzen.

Konstruktoren:

public Edge (Direction direction)

Erstellt eine neue Kante mit der angegebenen Richtung.

Methoden:

public Vector3 Get3DDirection ()

Gibt die Richtung als normalisierten Vektor3 zurück.

3.1.29 Klasse FadeEffect

Beschreibung:

Ein Postprocessing-Effekt, der eine Überblendung zwischen zwei Spielzuständen darstellt.

Eigenschaften:

private Boolean IsFinished

Gibt an, ob die Überblendung abgeschlossen ist und das RenderTarget nur noch den neuen Spielzustand darstellt.

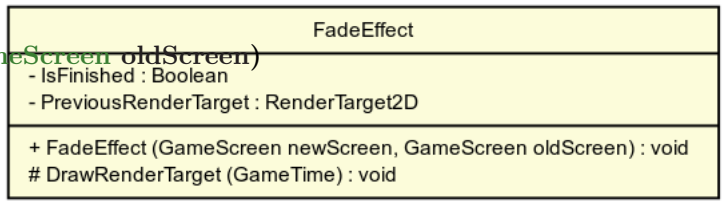
private RenderTarget2D PreviousRenderTarget

Der zuletzt gerenderte Frame im bisherigen Spielzustand.

Konstruktoren:

public FadeEffect (GameScreen newScreen, GameScreen oldScreen)

Erstellt einen Überblende-Effekt zwischen den angegebenen Spielzuständen.



Methoden:

protected void DrawRenderTarget (GameTime GameTime)

3.1.30 Klasse FileUtility

Beschreibung:

Eine Hilfsklasse für Dateioperationen.

Eigenschaften:

public String SettingsDirectory

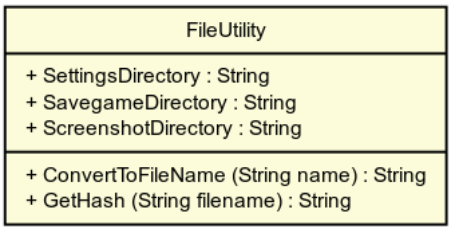
Das Einstellungsverzeichnis.

public String SavegameDirectory

Das Spielstandverzeichnis.

public String ScreenshotDirectory

Das Bildschirmfotoverzeichnis.



Methoden:

public String ConvertToFileName (String name)

Konvertiert einen Namen eines Knotens oder einer Challenge in einen gültigen Dateinamen durch Weglassen ungüliger Zeichen.

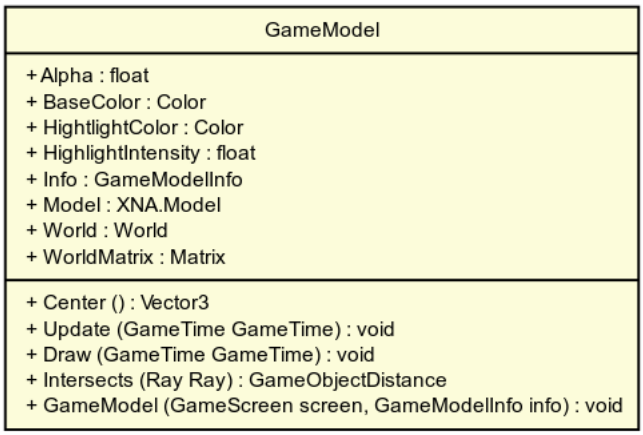
public String GetHash (String filename)

3.1.31 Klasse GameModel

Beschreibung:

Repräsentiert ein 3D-Modell in einer Spielwelt.

Eigenschaften:



public float Alpha

Die Transparenz des Modells.

public Color BaseColor

Die Farbe des Modells.

public Color HightlightColor

Die Auswahlfarbe des Modells.

public float HighlightIntensity

Die Intensität der Auswahlfarbe.

public GameModelInfo Info

Die Modellinformationen wie Position, Skalierung und der Dateiname des 3D-Modells.

public XNA.Model Model

Die Klasse des XNA-Frameworks, die ein 3D-Modell repräsentiert.

public World World

Die Spielwelt, in der sich das 3D-Modell befindet.

public Matrix WorldMatrix

Die Weltmatrix des 3D-Modells in der angegebenen Spielwelt.

Konstruktoren:

public GameModel (GameScreen screen, GameModelInfo info)

Erstellt ein neues 3D-Modell in dem angegebenen Spielzustand mit den angegebenen Modellinformationen.

Methoden:

public Vector3 Center ()

Gibt die Mitte des 3D-Modells zurück.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

public void Draw (GameTime GameTime)

Zeichnet das 3D-Modell in der angegebenen Spielwelt mit dem aktuellen Rendereffekt der Spielwelt.

public GameObjectDistance Intersects (Ray Ray)

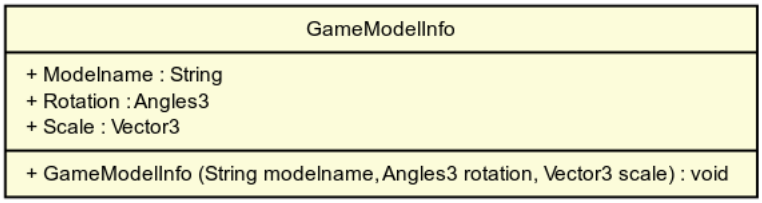
Überprüft, ob der Mausstrahl das 3D-Modell schneidet.

3.1.32 Klasse GameModelInfo

Beschreibung:

Enthält Informationen über ein 3D-Modell wie den Dateinamen, die Rotation und die Skalierung.

Eigenschaften:



public String Modelname

Der Dateiname des Modells.

public Angles3 Rotation

Die Rotation des Modells.

public Vector3 Scale

Die Skalierung des Modells.

Konstruktoren:

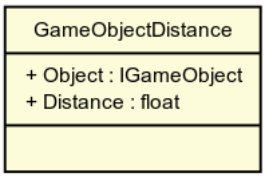
public GameModelInfo (String modelname, Angles3 rotation, Vector3 scale)

Erstellt ein neues Informations-Objekt eines 3D-Modells mit den angebenenen Informationen zu Dateiname, Rotation und Skalierung.

3.1.33 Klasse GameObjectDistance

Beschreibung:

Eigenschaften:



public IGameObject Object

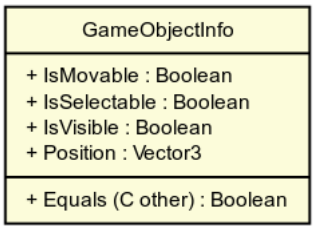
public float Distance

3.1.34 Klasse GameObjectInfo

Beschreibung:

Enthält Informationen über ein 3D-Objekt wie die Position, Sichtbarkeit, Verschiebbarkeit und Auswählbarkeit.

Eigenschaften:



public Boolean IsMovable

Die Verschiebbarkeit des Spielobjektes.

public Boolean IsSelectable

Die Auswählbarkeit des Spielobjektes.

public Boolean IsVisible

Die Sichtbarkeit des Spielobjektes.

public Vector3 Position

Die Position des Spielobjektes.

Methoden:

public Boolean Equals (C other)

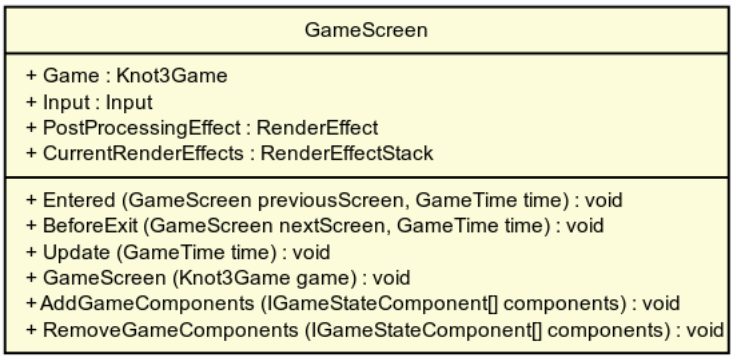
Vergleicht zwei Informationsobjekte für Spielobjekte.

3.1.35 Klasse GameScreen

Beschreibung:

Ein Spielzustand, der zu einem angegebenen Spiel gehört und einen Inputhandler und Rendereffekte enthält.

Eigenschaften:



public Knot3Game Game

Das Spiel, zu dem der Spielzustand gehört.

public Input Input

Der Inputhandler des Spielzustands.

public RenderEffect PostProcessingEffect

Der aktuelle Postprocessing-Effekt des Spielzustands

public RenderEffectStack CurrentRenderEffects

Ein Stack, der während dem Aufruf der Draw-Methoden der Spielkomponenten die jeweils aktuellen Rendereffekte enthält.

Konstruktoren:

public GameScreen (Knot3Game game)

Methoden:

public void Entered (GameScreen previousScreen, GameTime time)

Beginnt mit dem Füllen der Spielkomponentenliste des XNA-Frameworks und fügt sowohl für Tastatur- als auch für Mauseingaben einen Inputhandler für Widgets hinzu. Wird in Unterklassen von GameScreen reimplementiert und fügt zursätzlich weitere Spielkomponenten hinzu.

public void BeforeExit (GameScreen nextScreen, GameTime time)

Leert die Spielkomponentenliste des XNA-Frameworks.

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void AddGameComponents (IGameStateComponent[] components)

Fügt die angegebenen Spielkomponenten und deren über die Methode SubComponents() ermittelten Unterkomponenten der Spielkomponentenliste des XNA-Frameworks hinzu.

public void RemoveGameComponents (IGameStateComponent[] components)

Entfernt die angegebenen Spiekomponenten und deren Unterkomponenten von der Spielkomponentenliste des XNA-Frameworks.

3.1.36 Klasse GameScreenComponent

Beschreibung:

Eine Spielkomponente, die in einem angegebenen Spielzustand verwendet wird und eine bestimmte Priorität hat.

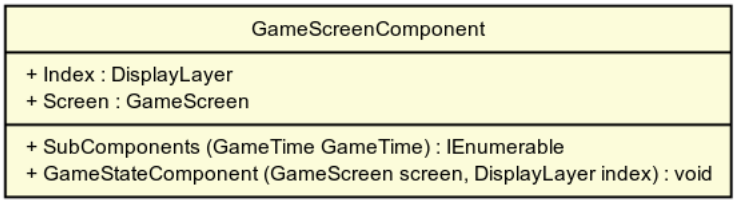
Eigenschaften:

public DisplayLayer Index

Die Zeichen- und Eingabepriorität.

public GameScreen Screen

Der zugewiesene Spielzustand.



Methoden:

public IEnumerable SubComponents (GameTime GameTime)

Gibt Spielkomponenten zurück, die in dieser Spielkomponente enthalten sind.

public void GameStateComponent (GameScreen screen, DisplayLayer index)

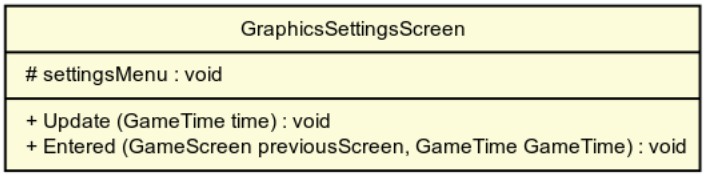
Erstellt eine neue Spielkomponente in dem angegebenen Spielzustand mit der angegebenen Priorität.

3.1.37 Klasse GraphicsSettingsScreen

Beschreibung:

Der Spielzustand, der die Grafik-Einstellungen darstellt.

Eigenschaften:



protected void settingsMenu

Das Menü, das die Einstellungen enthält.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

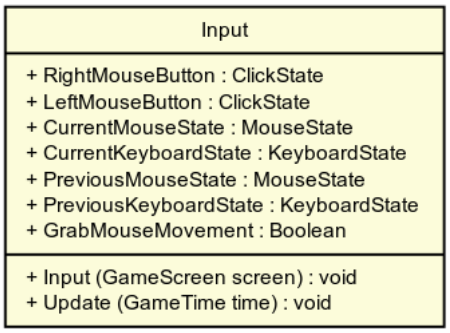
Fügt das Menü mit den Einstellungen in die Spielkomponentenliste ein.

3.1.38 Klasse Input

Beschreibung:

Stellt für jeden Frame die Maus- und Tastatureingaben bereit. Daraus werden die nicht von XNA bereitgestellten Mauseingaben berechnet.

Eigenschaften:



public ClickState RightMouseButton

Enthält den Klickzustand der rechten Maustaste.

public ClickState LeftMouseButton

Enthält den Klickzustand der linken Maustaste.

public MouseState CurrentMouseState

Enthält den Mauszustand von XNA des aktuellen Frames.

public KeyboardState CurrentKeyboardState

Enthält den Tastaturzustand von XNA des aktuellen Frames.

public MouseState PreviousMouseState

Enthält den Mauszustand von XNA des vorherigen Frames.

public KeyboardState PreviousKeyboardState

Enthält den Tastaturzustand von XNA des vorherigen Frames.

public Boolean GrabMouseMovement

Gibt an, ob die Mausbewegung für Kameradrehungen verwendet werden soll.

Konstruktoren:

public Input (GameScreen screen)

Erstellt ein neues Input-Objekt, das an den übergebenen Spielzustand gebunden ist.

Methoden:

public void Update (GameTime time)

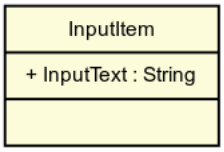
Wird für jeden Frame aufgerufen.

3.1.39 Klasse InputItem

Beschreibung:

Ein Menüeintrag, der Texteingaben vom Spieler annimmt.

Eigenschaften:



public String InputText

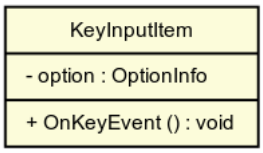
Beinhaltet den vom Spieler eingegebenen Text.

3.1.40 Klasse KeyInputItem

Beschreibung:

Ein Menüeintrag, der einen Tastendruck entgegennimmt und in der enthaltenen Option als Zeichenkette speichert.

Eigenschaften:



private OptionInfo option

Die Option in einer Einstellungsdatei.

Methoden:

public void OnKeyEvent ()

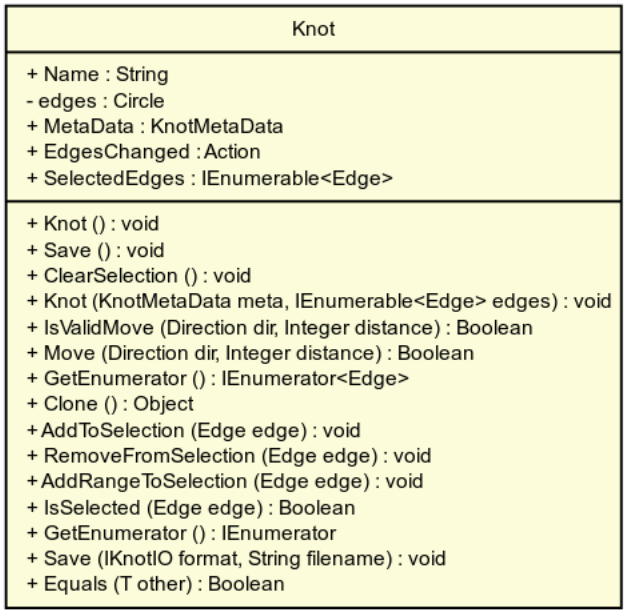
Speichert die aktuell gedrückte Taste in der Option.

3.1.41 Klasse Knot

Beschreibung:

Diese Klasse repräsentiert einen gültigen Knoten, bestehend aus einem Knoten-Metadaten-Objekt und einer doppelt-verketteten Liste von Kanten.

Eigenschaften:



public String Name

Der Name des Knotens, welcher auch leer sein kann. Beim Speichern muss der User in diesem Fall zwingend einen nichtleeren Namen wählen. Der Wert dieser Eigenschaft wird aus der „Name “-Eigenschaft des Metadaten-Objektes geladen und bei Änderungen wieder in diesem gespeichert. Beim Ändern dieser Eigenschaft wird automatisch auch der im Metadaten-Objekt enthaltene Dateiname verändert.

private Circle edges

Das Startelement der doppelt-verketteten Liste, in der die Kanten gespeichert werden.

public KnotMetaData MetaData

Die Metadaten des Knotens.

public Action EdgesChanged

Ein Ereignis, das in der Move-Methode ausgelöst wird, wenn sich die Struktur der Kanten geändert hat.

public IEnumerable<Edge> SelectedEdges

Enthält die aktuell vom Spieler selektierten Kanten in der Reihenfolge, in der sie selektiert wurden.

Konstruktoren:

public Knot ()

Erstellt einen minimalen Standardknoten. Das Metadaten-Objekt enthält in den Eigenschaften, die das Speicherformat und den Dateinamen beinhalten, den Wert „null “.

public Knot (KnotMetaData meta, IEnumerable<Edge> edges)

Erstellt einen neuen Knoten mit dem angegebenen Metadaten-Objekt und den angegebenen Kanten, die in der doppelt verketteten Liste gespeichert werden. Die Eigenschaft des Metadaten-Objektes, die die Anzahl der Kanten enthält, wird auf ein Delegate gesetzt, welches jeweils die aktuelle Anzahl der Kanten dieses Knotens zurückgibt.

Methoden:

public void Save ()

Speichert den Knoten unter dem Dateinamen in dem Dateiformat, das in dem Metadaten-Objekt angegeben ist. Enthalten entweder die Dateiname-Eigenschaft, die Dateiformat-Eigenschaft oder beide den Wert „null “, dann wird eine IOException geworfen.

public void ClearSelection ()

Hebt die aktuelle Kantenauswahl auf.

public Boolean IsValidMove (Direction dir, Integer distance)

Prüft, ob eine Verschiebung der aktuellen Kantenauswahl in die angegebene Richtung um die angegebene Distanz gültig ist.

public Boolean Move (Direction dir, Integer distance)

Verschiebt die aktuelle Kantenauswahl in die angegebene Richtung um die angegebene Distanz.

public IEnumerator<Edge> GetEnumerator ()

Gibt die doppelt-verkettete Kantenliste als Enumerator zurück.

public Object Clone ()

Erstellt eine vollständige Kopie des Knotens, inklusive der Kanten-Datentuktur und des Metadaten-Objekts.

public void AddToSelection (Edge edge)

Fügt die angegebene Kante zur aktuellen Kantenauswahl hinzu.

public void RemoveFromSelection (Edge edge)

Entfernt die angegebene Kante von der aktuellen Kantenauswahl.

public void AddRangeToSelection (Edge edge)

Fügt alle Kanten auf dem kürzesten Weg zwischen der zuletzt ausgewählten Kante und der angegebenen Kante zur aktuellen Kantenauswahl hinzu. Sind beide Wege gleich lang, wird der Weg in Richtung der ersten Kante ausgewählt.

public Boolean IsSelected (Edge edge)

Prüft, ob die angegebene Kante in der aktuellen Kantenauswahl enthalten ist.

public IEnumerator GetEnumerator ()

Gibt die doppelt-verkettete Kantenliste als Enumerator zurück.

public void Save (IKnotIO format, String filename)

Speichert den Knoten unter dem angegebenen Dateinamen in dem angebenenen Dateiformat.

public Boolean Equals (T other)

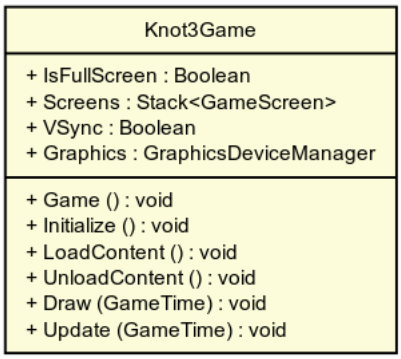
Prüft, ob die räumliche Struktur identisch ist, unabhängig von dem Startpunkt und der Richtung der Datenstruktur.

3.1.42 Klasse Knot3Game

Beschreibung:

Die zentrale Spielklasse, die von der „Game “-Klasse des XNA-Frameworks erbt.

Eigenschaften:



public Boolean IsFullScreen

Wird dieses Attribut ausgelesen, dann gibt es einen Wahrheitswert zurück, der angibt, ob sich das Spiel im Vollbildmodus befindet. Wird dieses Attribut auf einen Wert gesetzt, dann wird der Modus entweder gewechselt oder beibehalten, falls es auf den selben Wert gesetzt wird.

public Stack<GameScreen> Screens

Enthält als oberste Element den aktuellen Spielzustand und darunter die zuvor aktiven Spielzustände.

public Boolean VSync

Dieses Attribut dient sowohl zum Setzen des Aktivierungszustandes der vertikalen Synchronisation, als auch zum Auslesen dieses Zustandes.

public GraphicsDeviceManager Graphics

Der aktuelle Grafikgeräteverwalter des XNA-Frameworks.

Konstruktoren:

public Knot3Game ()

Erstellt ein neues zentrales Spielobjekt und setzt die Auflösung des BackBuffers auf die in der Einstelungsdatei gespeicherte Auflösung oder falls nicht vorhanden auf die aktuelle Bildschirmauflösung und wechselt in den Vollbildmodus.

Methoden:

public void LoadContent ()

Wird einmal beim Spielstart aufgerufen und lädt die Spielzustände.

public void UnloadContent ()

Macht nichts. Das Freigeben aller Objekte wird von der automatischen Speicherbereinigung übernommen.

public void Draw (GameTime time)

Ruft die Draw()-Methode des aktuellen Spielzustands auf.

public void Update (GameTime GameTime)

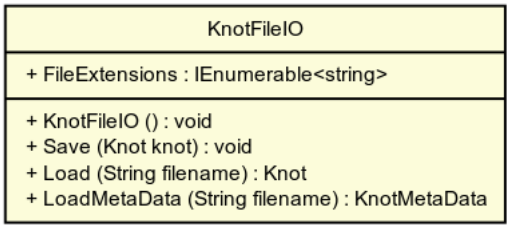
Ruft die Update()-Methode des aktuellen Spielzustands auf und wechselt den Spielzustand bei Bedarf.

3.1.43 Klasse KnotFileIO

Beschreibung:

Implementiert das Speicherformat für Knoten.

Eigenschaften:



public IEnumerable<string> FileExtensions

Die für eine Knoten-Datei gültigen Dateiendungen.

Konstruktoren:

public KnotFileIO ()

Erstellt ein KnotFileIO-Objekt.

Methoden:

public void Save (Knot knot)

Speichert einen Knoten in dem Dateinamen, der in dem Knot-Objekt enthalten ist.

public Knot Load (String filename)

Lädt eines Knotens aus einer angebenenen Datei.

public KnotMetaData LoadMetaData (String filename)

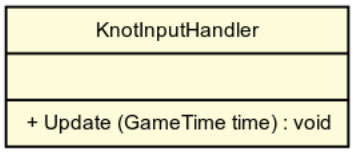
Lädt die Metadaten eines Knotens aus einer angegebenen Datei.

3.1.44 Klasse KnotInputHandler

Beschreibung:

Verarbeitet die Maus- und Tastatureingaben des Spielers und modifiziert die Kamera-Position und das Kamera-Ziel.

Eigenschaften:



private World world

Die Spielwelt.

private GameScreen screen

Der Spielzustand.

Konstruktoren:

public KnotInputHandler (GameScreen screen, World world)

Erstellt einen neuen KnotInputHandler für den angegebenen Spielzustand und die angegebene Spielwelt.

Methoden:

public void Update (GameTime time)

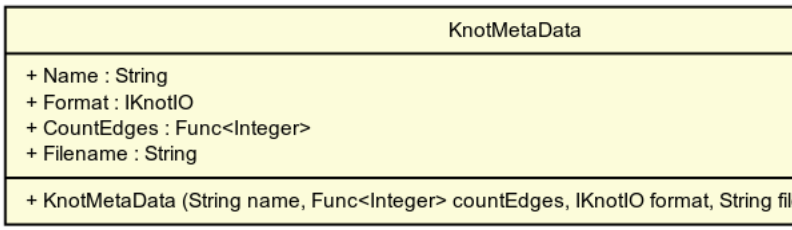
Wird für jeden Frame aufgerufen.

3.1.45 Klasse KnotMetaData

Beschreibung:

Enthält Metadaten eines Knotens, die aus einer Spielstand-Datei schneller eingelesen werden können, als der vollständige Knoten. Dieses Objekt enthält keine Datenstruktur zur Repräsentation der Kanten, sondern nur Informationen über den Namen des Knoten und die Anzahl seiner Kanten. Es kann ohne ein dazugehöriges Knoten-Objekt existieren, aber jedes Knoten-Objekt enthält genau ein Knoten-Metadaten-Objekt.

Eigenschaften:



public String Name

Der Anzeigename des Knotens, welcher auch leer sein kann. Beim Speichern muss der Spieler in diesem Fall zwingend einen nichtleeren Namen wählen. Wird ein neuer Anzeigename festgelegt, dann wird der Dateiname ebenfalls auf einen neuen Wert gesetzt, unabhängig davon ob er bereits einen Wert enthält oder „null “ist. Diese Eigenschaft kann öffentlich gelesen und gesetzt werden.

public IKnotIO Format

Das Format, aus dem die Metadaten geladen wurden. Es ist genau dann „null “, wenn die Metadaten nicht aus einer Datei gelesen wurden. Nur lesbar.

public Func<Integer> CountEdges

Ein Delegate, das die Anzahl der Kanten zurückliefert. Falls dieses Metadaten-Objekt Teil eines Knotens ist, gibt es dynamisch die Anzahl der Kanten des Knoten-Objektes zurück. Anderenfalls gibt es eine statische Zahl zurück, die beim Einlesen der Metadaten vor dem Erstellen dieses Objektes gelesen wurde. Nur lesbar.

public String Filename

Falls die Metadaten aus einer Datei eingelesen wurden, enthält dieses Attribut den Dateinamen, sonst „null “.

Konstruktoren:

public KnotMetaData (String name, Func<Integer> countEdges, IKnotIO format, String filename)

Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knotennamen und einer angebenen Funktion, welche eine Kantenanzahl zurück gibt. Zusätzlich wird der Dateiname oder das Speicherformat angegeben, aus dem die Metadaten gelesen wurden.

public KnotMetaData (String name, Func<Integer> countEdges)

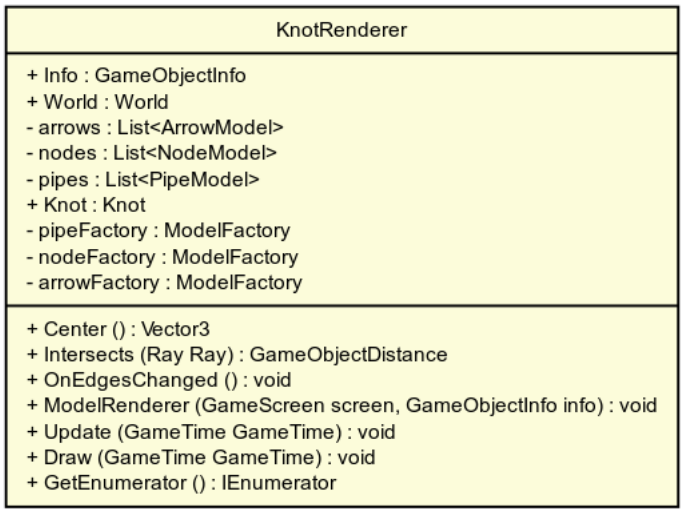
Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knotennamen und einer angebenen Funktion, welche eine Kantenanzahl zurück gibt.

3.1.46 Klasse KnotRenderer

Beschreibung:

Erstellt aus einem Knoten-Objekt die zu dem Knoten gehörenden 3D-Modelle sowie die 3D-Modelle der Pfeile, die nach einer Auswahl von Kanten durch den Spieler angezeigt werden.

Eigenschaften:



public GameObjectInfo Info

Enthält Informationen über die Position des Knotens.

public World World

Die Spielwelt, in der die 3D-Modelle erstellt werden sollen.

private List<ArrowModel> arrows

Die Liste der 3D-Modelle der Pfeile, die nach einer Auswahl von Kanten durch den Spieler angezeigt werden.

private List<NodeModel> nodes

Die Liste der 3D-Modelle der Kantenübergänge.

private List<PipeModel> pipes

Die Liste der 3D-Modelle der Kanten.

public Knot Knot

Der Knoten, für den 3D-Modelle erstellt werden sollen.

private ModelFactory pipeFactory

Der Zwischenspeicher für die 3D-Modelle der Kanten. Hier wird das Fabrik-Entwurfsmuster verwendet.

private ModelFactory nodeFactory

Der Zwischenspeicher für die 3D-Modelle der Kantenübergänge. Hier wird das Fabrik-Entwurfsmuster verwendet.

private ModelFactory arrowFactory

Der Zwischenspeicher für die 3D-Modelle der Pfeile. Hier wird das Fabrik-Entwurfsmuster verwendet.

Konstruktoren:

public KnotRenderer (GameScreen screen, GameObjectInfo info)

Erstellt ein neues KnotRenderer-Objekt für den angegebenen Spielzustand mit den angegebenen Spielobjekt-Informationen, die unter Anderem die Position des Knotenursprungs enthalten.

Methoden:

public Vector3 Center ()

Gibt den Ursprung des Knotens zurück.

public GameObjectDistance Intersects (Ray Ray)

Gibt immer „null “zurück.

public void OnEdgesChanged ()

Wird mit dem EdgesChanged-Event des Knotens verknüft.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

public void Draw (GameTime GameTime)

Zeichnet nichts.

public IEnumerator GetEnumerator ()

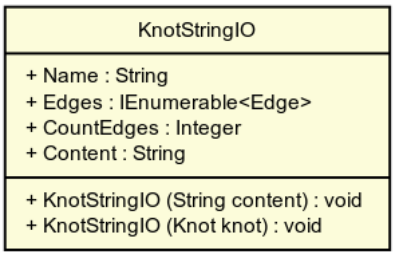
Gibt einen Enumerator der aktuell vorhandenen 3D-Modelle zurück.

3.1.47 Klasse KnotStringIO

Beschreibung:

Diese Klasse repräsentiert einen Parser für das Knoten-Austauschformat und enthält die eingelesenen Informationen wie den Namen des Knotens und die Kantenliste als Eigenschaften.

Eigenschaften:



public String Name

Der Name der eingelesenen Knotendatei oder des zugewiesenen Knotenobjektes.

public IEnumerable<Edge> Edges

Die Kanten der eingelesenen Knotendatei oder des zugewiesenen Knotenobjektes.

public Integer CountEdges

Die Anzahl der Kanten der eingelesenen Knotendatei oder des zugewiesenen Knotenobjektes.

public String Content

Erstellt aus den „Name“ - und „Edges“ -Eigenschaften einen neue Zeichenkette, die als Dateiinhalt in einer Datei eines Spielstandes einen gültigen Knoten repräsentiert.

Konstruktoren:

public KnotStringIO (String content)

Liest das in der angegebenen Zeichenkette enthaltene Dateiformat ein. Enthält es einen gültigen Knoten, so werden die „Name“ - und „Edges“ -Eigenschaften auf die eingelesenen Werte gesetzt. Enthält es einen ungültigen Knoten, so wird eine IOException geworfen und das Objekt wird nicht erstellt.

public KnotStringIO (Knot knot)

Erstellt ein neues Objekt und setzt die „Name“ - und „Edge“ -Eigenschaften auf die im angegebenen Knoten enthaltenen Werte.

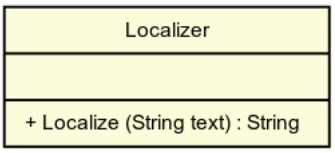
3.1.48 Klasse Localizer

Beschreibung:

Eine statische Klasse, die Bezeichner in lokalisierten Text umsetzen kann.

Eigenschaften:

private ConfigFile localization



Methoden:

public String Localize (String text)

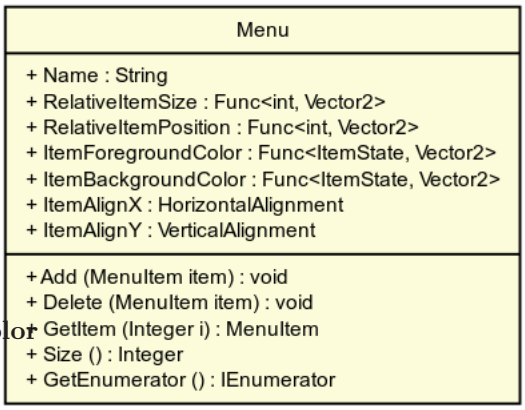
Liefert zu dem übergebenen Bezeichner den zugehörigen Text aus der Lokalisierungsdatei der aktuellen Sprache zurück, die dabei aus der Einstellungsdatei des Spiels gelesen wird.

3.1.49 Klasse Menu

Beschreibung:

Ein Menü enthält Bedienelemente zur Benutzerinteraktion. Diese Klasse bietet Standardwerte für Positionen, Größen, Farben und Ausrichtungen der Menueinträge. Sie werden gesetzt, wenn die Werte der Menüeinträge „null“ sind.

Eigenschaften:



public Func<int, Vector2> RelativeItemSize

Die von der Auflösung unabhängige Größe in Prozent.

public Func<int, Vector2> RelativeItemPosition

Die von der Auflösung unabhängige Position in Prozent.

public Func<ItemState, Color> ItemForegroundColor

Die vom Zustand des Menüeintrags abhängige Vordergrundfarbe des Menüeintrags.

public Func<ItemState, Color> ItemBackgroundColor

Die vom Zustand des Menüeintrags abhängige Hintergrundfarbe des Menüeintrags.

public HorizontalAlignment ItemAlignX

Die horizontale Ausrichtung der Menüeinträge.

public VerticalAlignment ItemAlignY

Die vertikale Ausrichtung der Menüeinträge.

Methoden:

public void Add (MenuItem item)

Fügt einen Eintrag in das Menü ein. Falls der Menüeintrag „null“ oder leere Werte für Position, Größe, Farbe oder Ausrichtung hat, werden die Werte mit denen des Menüs überschrieben.

public void Delete (MenuItem item)

Entfernt einen Eintrag aus dem Menü.

public MenuItem GetItem (Integer i)

Gibt einen Eintrag des Menüs zurück.

public Integer Size ()

Gibt die Anzahl der Einträge des Menüs zurück.

public IEnumerator GetEnumerator ()

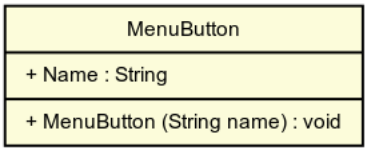
Gibt einen Enumerator über die Einträge des Menüs zurück.

3.1.50 Klasse MenuButton

Beschreibung:

Eine Schaltfläche, der eine Zeichenkette anzeigt und auf einen Linksklick reagiert.

Eigenschaften:



public Action OnClick

Die Aktion, die ausgeführt wird, wenn der Spieler auf die Schaltfläche klickt.

Konstruktoren:

public MenuButton (String name, Action onClick)

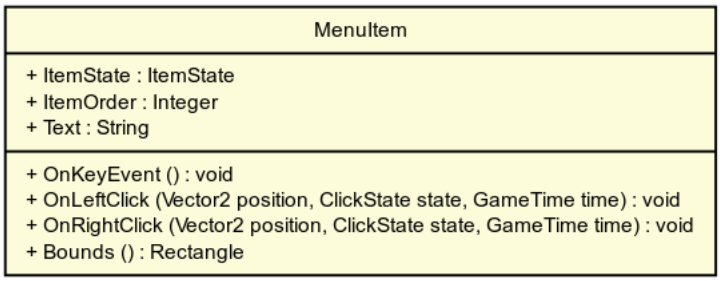
Erstellt eine neue Schaltfläche mit dem angegebenen Namen und der angegebenen Aktion.

3.1.51 Klasse MenuItem

Beschreibung:

Ein abstrakte Klasse für Menüeinträge, die

Eigenschaften:



public ItemState ItemState

Gibt an, ob die Maus sich über dem Eintrag befindet, ohne ihn anzuklicken, ob er ausgewählt ist oder nichts von beidem.

public Integer ItemOrder

Die Zeichenreihenfolge.

public String Text

Der Anzeigetext der Schaltfläche.

Methoden:

public void OnKeyEvent ()

Reaktionen auf Tasteneíngaben.

public void OnLeftClick (Vector2 position, ClickState state, GameTime time)

Reaktionen auf einen Linksklick.

public void OnRightClick (Vector2 position, ClickState state, GameTime time)

Reaktionen auf einen Rechtsklick.

public Rectangle Bounds ()

Gibt die Ausmaße des Eintrags zurück.

3.1.52 Klasse MenuScreen

Beschreibung:

Eine abstrakte Klasse, von der alle Spielzutsände erben, die Menüs darstellen.

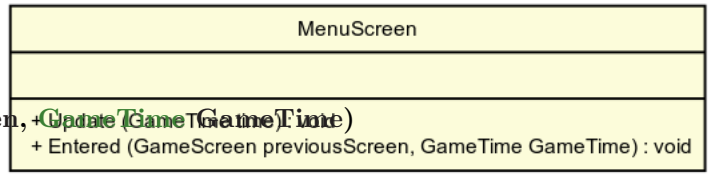
Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Wird aufgerufen, wenn in diesen Spielzustand gewechselt wird.



3.1.53 Klasse ModelFactory

Beschreibung:

Ein Zwischenspeicher für 3D-Modelle.

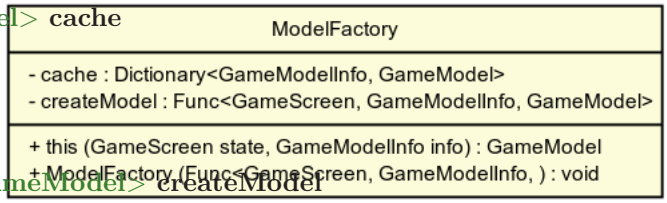
Eigenschaften:

private Dictionary<GameModelInfo, GameModel> cache

Die Zuordnung zwischen den Modellinformationen zu den 3D-Modellen.

private Func<GameScreen, GameModelInfo, GameModel> createModel

Ein Delegate, das beim Erstellen eines Zwischenspeichers zugewiesen wird und aus den angegebenen Modellinformationen und dem angegebenen Spielzustand ein 3D-Modell erstellt.



Konstruktoren:

public ModelFactory (GameModelInfo, GameModel>, Func<GameScreen createModel)

Erstellt einen neuen Zwischenspeicher.

Methoden:

public GameModel this (GameScreen state, GameModelInfo info)

Falls das 3D-Modell zwischengespeichert ist, wird es zurückgegeben, sonst mit createModel() erstellt.

3.1.54 Klasse ModelMouseHandler

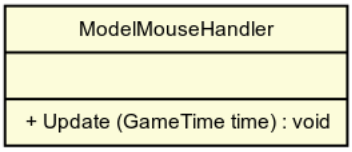
Beschreibung:

Ein Inputhandler, der Mauseingaben auf 3D-Modellen verarbeitet.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.



3.1.55 Klasse MousePointer

Beschreibung:

Repräsentiert einen Mauszeiger.

Konstruktoren:

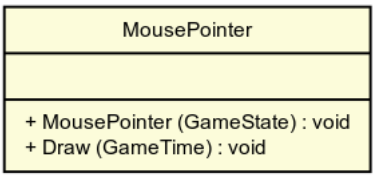
public MousePointer (GameScreen screen)

Erstellt einen neuen Mauszeiger für den angegebenen Spielzustand.

Methoden:

public void Draw (GameTime time)

Zeichnet den Mauszeiger.



3.1.56 Klasse NodeMap

Beschreibung:

Eine Zuordnung zwischen Kanten und Kantenübergänge.

Methoden:

public Node From (Edge edge)

Gibt den Übergang am Anfang der Kante zurück.

public Node To (Edge edge)

Gibt den Übergang am Ende der Kante zurück.

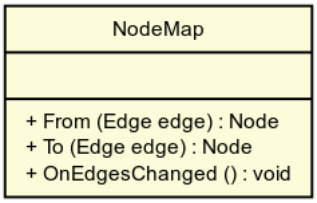
public void OnEdgesChanged ()

Aktualisiert die Zuordnung, wenn sich die Kanten geändert haben.

3.1.57 Klasse NodeModel

Beschreibung:

Ein 3D-Modell, das einen Kantenübergang darstellt.



Eigenschaften:

public NodeModelInfo Info

Enthält Informationen über den darzustellende 3D-Modell des Kantenübergangs.

Konstruktoren:

public NodeModel (GameScreen screen, NodeModelInfo info)

Erstellt ein neues 3D-Modell mit dem angegebenen Spielzustand und dem angegebenen Informationsobjekt.

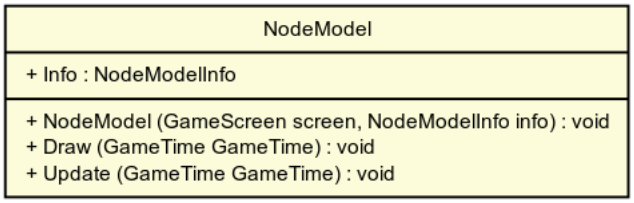
Methoden:

public void Draw (GameTime GameTime)

Zeichnet das 3D-Modell mit dem aktuellen Rendereffekt.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

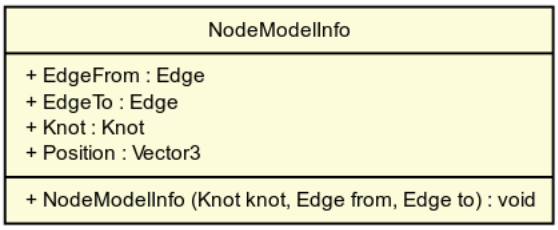


3.1.58 Klasse NodeModelInfo

Beschreibung:

Enthält Informationen über ein 3D-Modell, das einen Kantenübergang darstellt.

Eigenschaften:



public Edge EdgeFrom

Die Kante vor dem Übergang.

public Edge EdgeTo

Die Kante nach dem Übergang.

public Knot Knot

Der Knoten, der die Kanten enthält.

public Vector3 Position

Die Position des Übergangs.

Konstruktoren:

public NodeModelInfo (Knot knot, Edge from, Edge to)

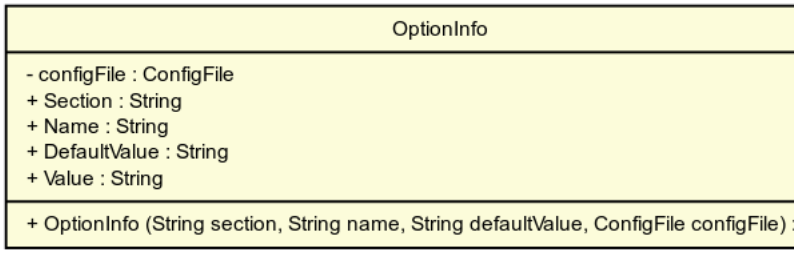
Erstellt ein neues Informationsobjekt für ein 3D-Modell, das einen Kantenübergang darstellt.

3.1.59 Klasse OptionInfo

Beschreibung:

Enthält Informationen über einen Eintrag in einer Einstellungsdatei.

Eigenschaften:



private ConfigFile configFile

Die Einstellungsdatei.

public String Section

Der Abschnitt der Einstellungsdatei.

public String Name

Der Name der Option.

public String DefaultValue

Der Standardwert der Option.

public String Value

Der Wert der Option.

Konstruktoren:

public OptionInfo (String section, String name, String defaultValue, ConfigFile configFile)

Erstellt ein neues OptionsInfo-Objekt aus den übergegebenen Werten.

3.1.60 Klasse Options

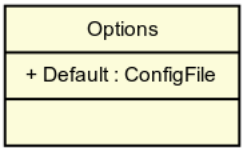
Beschreibung:

Eine statische Klasse, die eine Referenz auf die zentrale Einstellungsdatei des Spiels enthält.

Eigenschaften:

public ConfigFile Default

Die zentrale Einstellungsdatei des Spiels.



3.1.61 Klasse PauseDialog

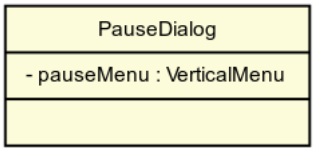
Beschreibung:

Pausiert ein Spieler im Creative- oder Challenge-Modus das Spiel, wird dieser Dialog über anderen Spielkonmponenten angezeigt.

Eigenschaften:

private VerticalMenu pauseMenu

Das Menü, das verschiedene Schaltflächen enthält.



3.1.62 Klasse PipeModel

Beschreibung:

Ein 3D-Modell, das eine Kante darstellt.

Eigenschaften:

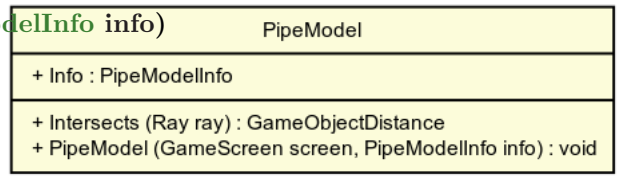
public PipeModelInfo Info

Enthält Informationen über die darzustellende Kante.

Konstruktoren:

public PipeModel (GameScreen screen, PipeModelInfo info)

Erstellt ein neues 3D-Modell mit dem angegebenen Spielzustand und den angegebenen Spielinformationen.



Methoden:

public GameObjectDistance Intersects (Ray ray)

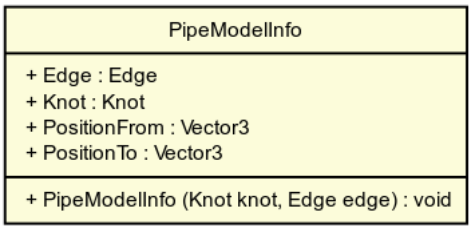
Prüft, ob der angegebene Mausstrahl das 3D-Modell schneidet.

3.1.63 Klasse PipeModelInfo

Beschreibung:

Enthält Informationen über ein 3D-Modell, das eine Kante darstellt.

Eigenschaften:



public Edge Edge

Die Kante, die durch das 3D-Modell dargestellt wird.

public Knot Knot

Der Knoten, der die Kante enthält.

public Vector3 PositionFrom

Die Position, an der die Kante beginnt.

public Vector3 PositionTo

Die Position, an der die Kante endet.

Konstruktoren:

public PipeModelInfo (Knot knot, Edge edge)

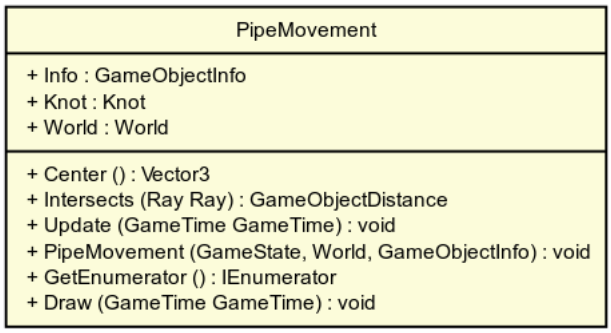
Erstellt ein neues Informationsobjekt für ein 3D-Modell, das eine Kante darstellt.

3.1.64 Klasse PipeMovement

Beschreibung:

Ein Inputhandler, der für das Verschieben der Kanten zuständig ist.

Eigenschaften:



public GameObjectInfo Info

Enthält Informationen über die Position des Knotens.

public Knot Knot

Der Knoten, dessen Kanten verschoben werden können.

public World World

Die Spielwelt, in der sich die 3D-Modelle der Kanten befinden.

Konstruktoren:

public PipeMovement (GameScreen screen, World world, GameObjectInfo info)

Methoden:

public Vector3 Center ()

Gibt den Ursprung des Knotens zurück.

public GameObjectDistance Intersects (Ray Ray)

Gibt immer „null “zurück.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

public IEnumerator GetEnumerator ()

Gibt einen Enumerator über die während einer Verschiebeaktion dynamisch erstellten 3D-Modelle zurück.

public void Draw (GameTime GameTime)

Zeichnet die während einer Verschiebeaktion dynamisch erstellten 3D-Modelle.

3.1.65 Klasse PrinterIO

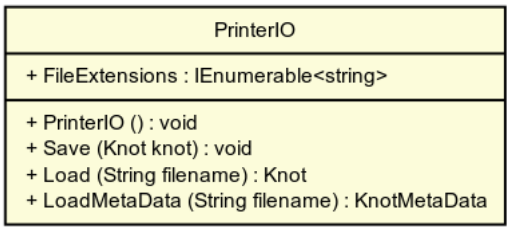
Beschreibung:

Ein Exportformat für 3D-Drucker.

Eigenschaften:

public IEnumerable<string> FileExtensions

Die gültigen Dateiendungen für das 3D-Drucker-Format.



Konstruktoren:

public PrinterIO ()

Erstellt ein neues PrinterIO-Objekt.

Methoden:

public void Save (Knot knot)

Exportiert den Knoten in einem gültigen 3D-Drucker-Format.

public Knot Load (String filename)

Gibt eine IOException zurück.

public KnotMetaData LoadMetaData (String filename)

Gibt eine IOException zurück.

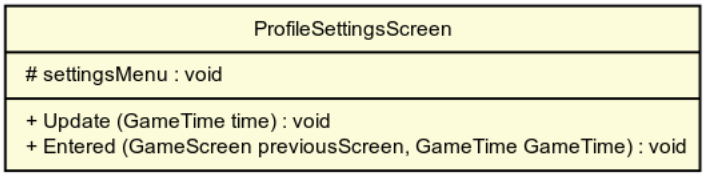
3.1.66 Klasse ProfileSettingsScreen

Beschreibung:

Der Spielzustand, der die Profil-Einstellungen darstellt.

Eigenschaften:

protected void settingsMenu



Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

Fügt das Menü mit den Einstellungen in die Spielkomponentenliste ein.

3.1.67 Klasse RenderEffect

Beschreibung:

Eine abstrakte Klasse, die eine Implementierung von IRenderEffect darstellt.

Eigenschaften:

public RenderTarget2D RenderTarget

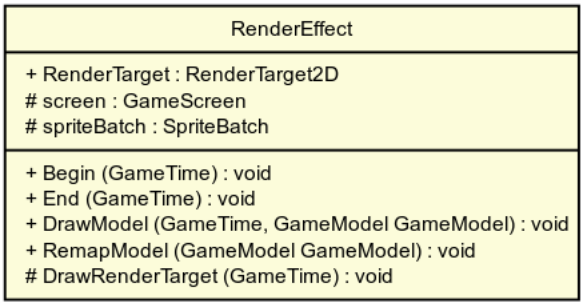
Das Rendertarget, in das zwischen dem Aufruf der Begin()- und der End()-Methode gezeichnet wird, weil es in Begin() als primäres Rendertarget des XNA-Frameworks gesetzt wird.

protected GameScreen screen

Der Spielzustand, in dem der Effekt verwendet wird.

protected SpriteBatch spriteBatch

Ein Spritestapel, der verwendet wird, um das Rendertarget dieses Rendereffekts auf das übergeordnete Rendertarget zu zeichnen.



Methoden:

public void Begin (GameTime)

In der Methode Begin() wird das aktuell von XNA genutzte Rendertarget auf einem Stack gesichert und das Rendertarget des Effekts wird als aktuelles Rendertarget gesetzt.

public void End (GameTime)

Das auf dem Stack gesicherte, vorher genutzte Rendertarget wird wiederhergestellt und das Rendertarget dieses Rendereffekts wird, unter Umständen in Unterklassen verändert, auf dieses ubergeordnete Rendertarget gezeichnet.

public void DrawModel (GameTime, GameModel GameModel)

Zeichnet das übergebene 3D-Modell auf das Rendertarget.

public void RemapModel (GameModel GameModel)

Beim Laden des Modells wird von der XNA-Content-Pipeline jedem ModelMeshPart ein Shader der Klasse BasicEffect zugewiesen. Für die Nutzung des Modells in diesem Rendereffekt kann jedem ModelMeshPart ein anderer Shader zugewiesen werden.

protected void DrawRenderTarget (GameTime time)

3.1.68 Klasse RenderEffectStack

Beschreibung:

Ein Stapel, der während der Draw-Aufrufe die Hierarchie der aktuell verwendeten Rendereffekte verwaltet und automatisch das aktuell von XNA verwendete Rendertarget auf das Rendertarget des obersten Rendereffekts setzt.

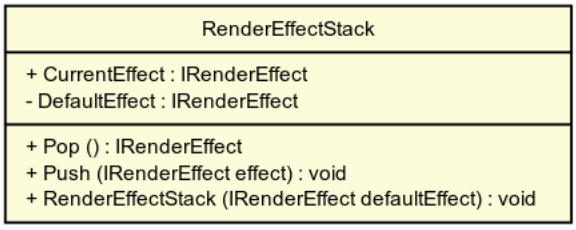
Eigenschaften:

public IRenderEffect CurrentEffect

Der oberste Rendereffekt.

private IRenderEffect DefaultEffect

Der Standard-Rendereffekt, der verwendet wird, wenn der Stapel leer ist.



Konstruktoren:

public RenderEffectStack (IRenderEffect defaultEffect)

Erstellt einen neuen Rendereffekt-Stapel.

Methoden:

public IRenderEffect Pop ()

Entfernt den obersten Rendereffekt vom Stapel.

public void Push (IRenderEffect effect)

Legt einen Rendereffekt auf den Stapel.

3.1.69 Klasse SettingsScreen

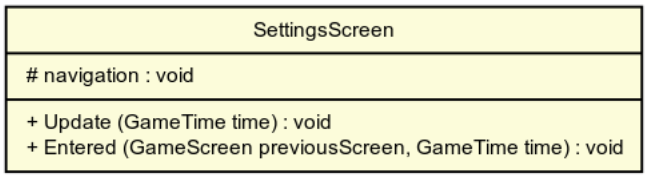
Beschreibung:

Ein Spielzustand, der das Haupt-Einstellungsmenü zeichnet.

Eigenschaften:

protected void navigation

Das Haupt-Einstellungsmenü.



Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime time)

Fügt das Haupt-Einstellungsmenü in die Spielkomponentenliste ein.

3.1.70 Klasse ShadowGameModel

Beschreibung:

Die 3D-Modelle, die während einer Verschiebung von Kanten die Vorschaumodelle repräsentieren.

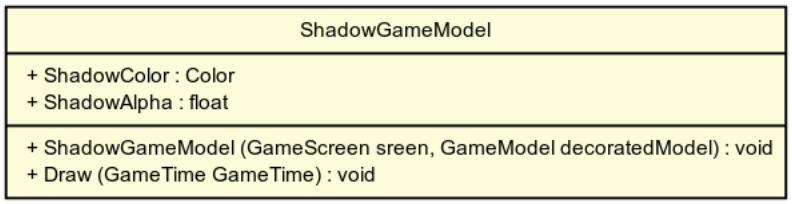
Eigenschaften:

public Color ShadowColor

Die Farbe der Vorschaumodelle.

public float ShadowAlpha

Die Transparenz der Vorschaumodelle.



Konstruktoren:

public ShadowGameModel (GameScreen sreen, GameModel decoratedModel)

Erstellt ein neues Vorschaumodell in dem angegebenen Spielzustand für das angegebene zu dekorierende Modell.

Methoden:

public void Draw (GameTime GameTime)

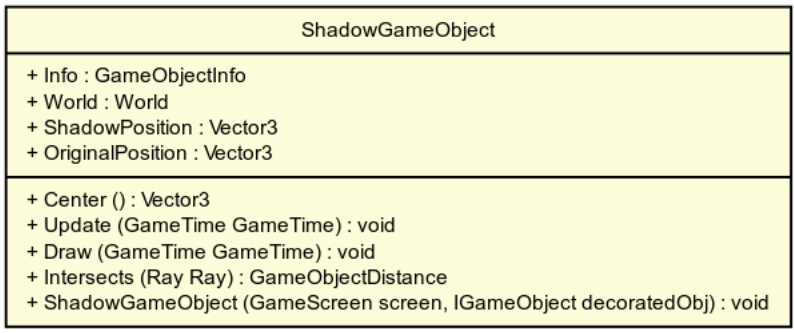
Zeichnet das Vorschaumodell.

3.1.71 Klasse ShadowGameObject

Beschreibung:

Eine abstrakte Klasse, die ein Vorschau-Spielobjekt darstellt.

Eigenschaften:



public GameObjectInfo Info

Enthält Informationen über das Vorschau-Spielobjekt.

public World World

Eine Referenz auf die Spielwelt, in der sich das Spielobjekt befindet.

public Vector3 ShadowPosition

Die Position, an der das Vorschau-Spielobjekt gezeichnet werden soll.

public Vector3 OriginalPosition

Die Position, an der sich das zu dekorierende Objekt befindet.

Konstruktoren:

public ShadowGameObject (GameScreen screen, IGameObject decoratedObj)

Erstellt ein neues Vorschauobjekt in dem angegebenen Spielzustand für das angegebene zu dekorierende Objekt.

Methoden:

public Vector3 Center ()

Die Position, an der das Vorschau-Spielobjekt gezeichnet werden soll.

public void Update (GameTime GameTime)

Wird für jeden Frame aufgerufen.

public void Draw (GameTime GameTime)

Zeichnet das Vorschau-Spielobjekt.

public GameObjectDistance Intersects (Ray Ray)

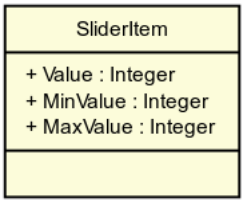
Prüft, ob der angegebene Mausstrahl das Vorschau-Spielobjekt schneidet.

3.1.72 Klasse SliderItem

Beschreibung:

Ein Menüeintrag, der einen Schieberegler bereitstellt, mit dem man einen Wert zwischen einem minimalen und einem maximalen Wert über Verschiebung einstellen kann.

Eigenschaften:



public Integer Value

Der aktuelle Wert.

public Integer MinValue

Der minimale Wert.

public Integer MaxValue

Der maximale Wert.

public Integer Step

3.1.73 Klasse StandardEffect

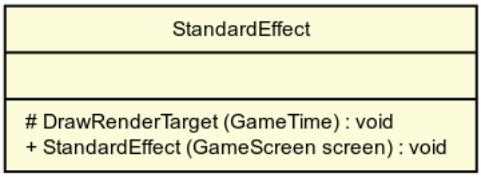
Beschreibung:

Ein Rendereffekt, der 3D-Modelle mit dem von der XNA-Content-Pipeline standardmäßig zugewiesenen BasicEffect-Shader zeichnet und keinen Post-Processing-Effekt anwendet.

Konstruktoren:

public StandardEffect (GameScreen screen)

Erstellt einen neuen Standardeffekt.



Methoden:

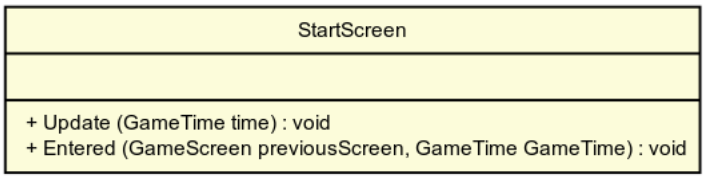
protected void DrawRenderTarget (GameTime GameTime)

3.1.74 Klasse StartScreen

Beschreibung:

Der Startbildschirm.

Eigenschaften:



private Menu buttons

Die Schaltflächen des Startbildschirms.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime GameTime)

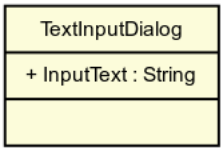
Fügt die das Menü in die Spielkomponentenliste ein.

3.1.75 Klasse TextInputDialog

Beschreibung:

Ein Dialog, der eine Texteingabe des Spielers entgegennimmt.

Eigenschaften:



public String InputText

Der Text, der durch den Spieler eingegeben wurde.

3.1.76 Klasse TutorialChallengeModeScreen

Beschreibung:

Eine Einführung in das Spielen von Challenges.
Der Spieler wird dabei durch Anweisungen an das Lösen von Challenges herangeführt.

Methoden:

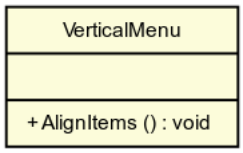
public void Entered (GameScreen previousScreen, GameTime GameTime)

3.1.77 Klasse VerticalMenu

Beschreibung:

Ein Menü, das alle Einträge vertikal anordnet.

Methoden:



public void AlignItems ()

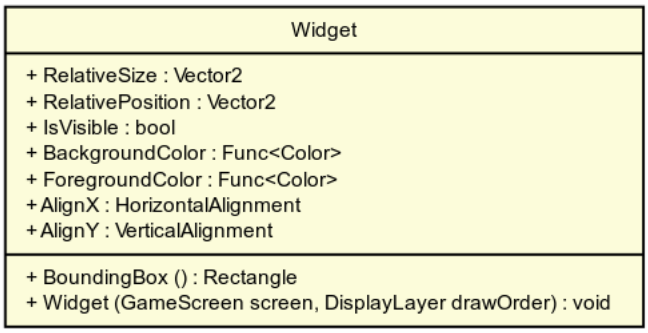
Ordnet die Einträge vertikal an.

3.1.78 Klasse Widget

Beschreibung:

Eine abstrakte Klasse, von der alle Element der grafischen Benutzeroberfläche erben.

Eigenschaften:



public Vector2 RelativeSize

Die von der Auflösung unabhängige Größe in Prozent.

public Vector2 RelativePosition

Die von der Auflösung unabhängige Position in Prozent.

public bool IsVisible

Gibt an, ob das grafische Element sichtbar ist.

public Func<Color> BackgroundColor

Die Hintergrundfarbe.

public Func<Color> ForegroundColor

Die Vordergrundfarbe.

public HorizontalAlignment AlignX

Die horizontale Ausrichtung.

public VerticalAlignment AlignY

Die vertikale Ausrichtung.

Konstruktoren:

public Widget (GameScreen screen, DisplayLayer drawOrder)

Erstellt ein neues grafisches Benutzerschnittstellenelement in dem angegebenen Spielzustand mit der angegebenen Zeichenreihenfolge.

Methoden:

public Rectangle BoundingBox ()

Die Ausmaße des grafischen Elements

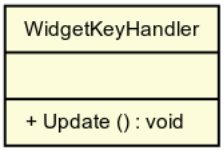
3.1.79 Klasse WidgetKeyHandler

Beschreibung:

Ein Inputhandler, der Tastatureingaben auf Widgets verarbeitet.

Methoden:

public void Update ()



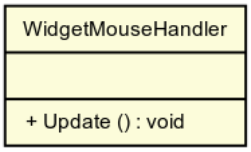
3.1.80 Klasse WidgetMouseHandler

Beschreibung:

Ein Inputhandler, der Mauseingaben auf Widgets verarbeitet.

Methoden:

public void Update ()



3.1.81 Klasse World

Beschreibung:

Repräsentiert eine Spielwelt, in der sich 3D-Modelle befinden und gezeichnet werden können.

Eigenschaften:

public Camera Camera

Die Kamera dieser Spielwelt.

public List<IGameObject> Objects

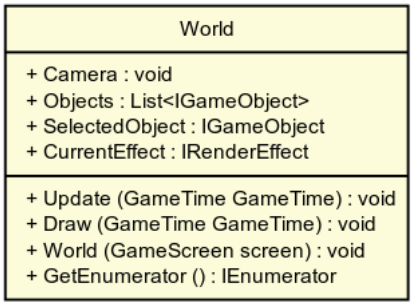
Die Liste von Spielobjekten.

public IGameObject SelectedObject

Das aktuell ausgewählte Spielobjekt.

public IRenderEffect CurrentEffect

Der aktuell angewendete Rendereffekt.



Konstruktoren:

public World (GameScreen screen)

Erstellt eine neue Spielwelt im angegebenen Spielzustand.

Methoden:

public void Update (GameTime GameTime)

Ruft auf allen Spielobjekten die Update()-Methode auf.

public void Draw (GameTime GameTime)

Ruft auf allen Spielobjekten die Draw()-Methode auf.

public IEnumerator GetEnumerator ()

Liefert einen Enumerator über die Spielobjekte dieser Spielwelt.

3.2 Schnittstellen

3.2.1 Schnittstelle IChallengeIO

Beschreibung:

Diese Schnittstelle enthält Methoden, die von Speicherformaten für Challenges implementiert werden müssen.

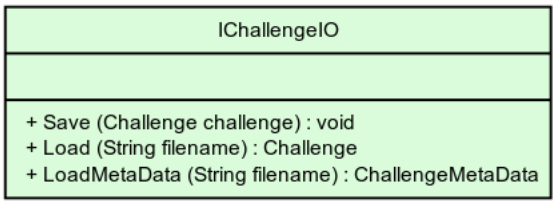
Methoden:

public void Save (Challenge challenge)

Speichert eine Challenge.

public Challenge Load (String filename)

Lädt eine Challenge.



public ChallengeMetaData LoadMetaData (String filename)

Lädt die Metadaten einer Challenge.

3.2.2 Schnittstelle IGameObject

Beschreibung:

Diese Schnittstelle repräsentiert ein Spielobjekt und enthält eine Referenz auf die Spielwelt, in der sich das Spielobjekt befindet, sowie Informationen zu dem Spielobjekt.

Eigenschaften:

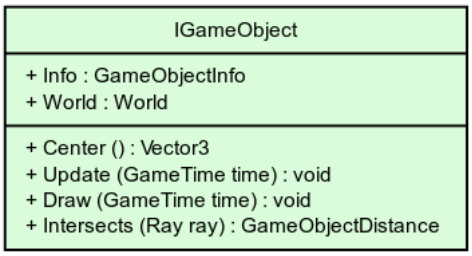
public GameObjectInfo Info

Informationen über das Spielobjekt, wie z.B. die Position.

public World World

Eine Referenz auf die Spielwelt, in der sich das Spielobjekt befindet.

Methoden:



public Vector3 Center ()

Die Mitte des Spielobjektes im 3D-Raum.

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Draw (GameTime time)

Zeichnet das Spielobjekt.

public GameObjectDistance Intersects (Ray ray)

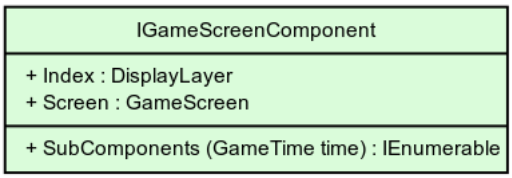
Überprüft, ob der Mausstrahl das Spielobjekt schneidet.

3.2.3 Schnittstelle IGameScreenComponent

Beschreibung:

Eine Schnittstelle für eine Spielkomponente, die in einem angegebenen Spielzustand verwendet wird und eine bestimmte Priorität hat.

Eigenschaften:



public DisplayLayer Index

Die Zeichen- und Eingabepriorität.

public GameScreen Screen

Der zugewiesene Spielzustand.

Methoden:

public IEnumerable SubComponents (GameTime time)

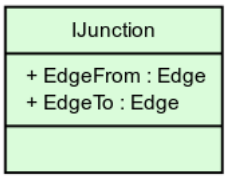
Gibt Spielkomponenten zurück, die in dieser Spielkomponente enthalten sind.

3.2.4 Schnittstelle IJunction

Beschreibung:

Repräsentiert einen Übergang zwischen zwei Kanten.

Eigenschaften:



public Edge EdgeFrom

Die Kante vor dem Übergang.

public Edge EdgeTo

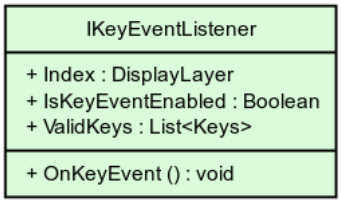
Die Kante nach dem Übergang.

3.2.5 Schnittstelle IKeyEventListener

Beschreibung:

Eine Schnittstelle, die von Klassen inmplementiert wird, welche auf Tastatureingaben reagieren.

Eigenschaften:



public DisplayLayer Index

Die Eingabepriorität.

public Boolean IsKeyEventEnabled

Zeigt an, ob die Klasse zur Zeit auf Tastatureingaben reagiert.

public List<Keys> ValidKeys

Die Tasten, auf die die Klasse reagiert.

Methoden:

public void OnKeyEvent ()

Die Reaktion auf eine Tasteneingabe.

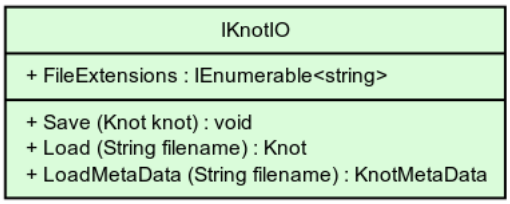
3.2.6 Schnittstelle IKnotIO

Beschreibung:

Diese Schnittstelle enthält Methoden, die von Speicherformaten für Knoten implementiert werden müssen.

Eigenschaften:

public IEnumerable<string> FileExtensions



Methoden:

public void Save (Knot knot)

Speichert einen Knoten.

public Knot Load (String filename)

Lädt einen Knoten.

public KnotMetaData LoadMetaData (String filename)

Lädt die Metadaten eines Knotens.

3.2.7 Schnittstelle IMouseEventListener

Beschreibung:

Eine Schnittstelle, die von Klassen implementiert wird, die auf Maus-Klicks reagieren.

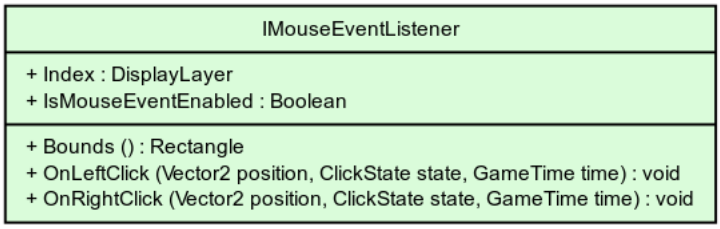
Eigenschaften:

public DisplayLayer Index

Die Eingabepriorität.

public Boolean IsMouseEventEnabled

Ob die Klasse zur Zeit auf Mausklicks reagiert.



Methoden:

public Rectangle Bounds ()

Die Ausmaße des von der Klasse repräsentierten Objektes.

public void OnLeftClick (Vector2 position, ClickState state, GameTime time)

Die Reaktion auf einen Linksklick.

public void OnRightClick (Vector2 position, ClickState state, GameTime time)

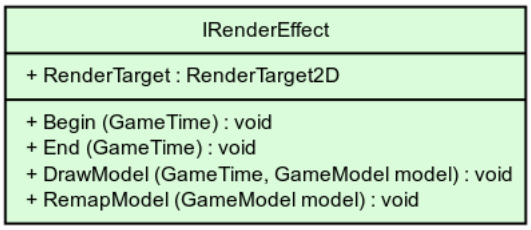
Die Reaktion auf einen Rechtsklick.

3.2.8 Schnittstelle IRenderEffect

Beschreibung:

Stellt eine Schnittstelle für Klassen bereit, die Rendereffekte ermöglichen.

Eigenschaften:



public RenderTarget2D RenderTarget

Das Rendertarget, in das zwischen dem Aufruf der Begin()- und der End()-Methode gezeichnet wird, weil es in Begin() als primäres Rendertarget des XNA-Frameworks gesetzt wird.

Methoden:

public void Begin (GameTime)

In der Methode Begin() wird das aktuell von XNA genutzte Rendertarget auf einem Stack gesichert und das Rendertarget des Effekts wird als aktuelles Rendertarget gesetzt.

public void End (GameTime)

Das auf dem Stack gesicherte, vorher genutzte Rendertarget wird wiederhergestellt und das Rendertarget dieses Rendereffekts wird, unter Umständen in Unterklassen verändert, auf dieses ubergeordnete Rendertarget gezeichnet.

public void DrawModel (GameTime, GameModel model)

Zeichnet das übergebene 3D-Modell auf das Rendertarget.

public void RemapModel (GameModel model)

Beim Laden des Modells wird von der XNA-Content-Pipeline jedem ModelMeshPart ein Shader der Klasse BasicEffect zugewiesen. Für die Nutzung des Modells in diesem Rendereffekt kann jedem ModelMeshPart ein anderer Shader zugewiesen werden.

3.3 Enumerationen

Kapitel 4

Abläufe

4.1 Sequenzdiagramme

Kapitel 5

Klassenindex

Kapitel 6

Anmerkungen

Kapitel 7

Glossar

Test	Test