

ENTWURFSDOKUMENT

(V. 1.0)

KNOT³

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

1	Einleitung	2
2	Aufbau	3
2.1	Architektur	3
2.2	Klassendiagramm	4
2.3	Verwendete Entwurfsmuster	4
3	Klassenübersicht	5
3.1	Klassen	5
3.1.1	Klasse Angles3	5
3.1.2	Klasse ArrowModel	6
3.1.3	Klasse ArrowModelInfo	6
3.1.4	Klasse AudioSettingsScreen	7
3.1.5	Klasse BooleanOptionInfo	7
3.1.6	Klasse Camera	7
3.1.7	Klasse CelShadingEffect	9
3.1.8	Klasse Challenge	9
3.1.9	Klasse ChallengeFileIO	10
3.1.10	Klasse ChallengeLoadScreen	11
3.1.11	Klasse ChallengeMetaData	11
3.1.12	Klasse ChallengeModeScreen	12
3.1.13	Klasse CheckBoxItem	13
3.1.14	Klasse Circle	14
3.1.15	Klasse ColorPicker	15
3.1.16	Klasse ColorPickItem	15
3.1.17	Klasse ConfigFile	15
3.1.18	Klasse ConfirmDialog	16
3.1.19	Klasse ControlSettingsScreen	16
3.1.20	Klasse CreativeLoadScreen	17
3.1.21	Klasse CreativeModeScreen	17
3.1.22	Klasse CreditsScreen	18
3.1.23	Klasse Dialog	18
3.1.24	Klasse DistinctOptionInfo	19
3.1.25	Klasse DrawableGameScreenComponent	19
3.1.26	Klasse DropDownEntry	20
3.1.27	Klasse DropDownMenuItem	20
3.1.28	Klasse Edge	21
3.1.29	Klasse FadeEffect	21
3.1.30	Klasse FileUtility	22

3.1.31	Klasse	GameModel	23
3.1.32	Klasse	GameModelInfo	24
3.1.33	Klasse	GameObjectInfo	24
3.1.34	Klasse	GameScreen	25
3.1.35	Klasse	GameScreenComponent	26
3.1.36	Klasse	GraphicsSettingsScreen	26
3.1.37	Klasse	Input	27
3.1.38	Klasse	InputItem	28
3.1.39	Klasse	KeyInputItem	28
3.1.40	Klasse	Knot	28
3.1.41	Klasse	Knot3Game	30
3.1.42	Klasse	KnotFileIO	31
3.1.43	Klasse	KnotInputHandler	32
3.1.44	Klasse	KnotMetaData	32
3.1.45	Klasse	KnotRenderer	32
3.1.46	Klasse	KnotStringIO	34
3.1.47	Klasse	Localizer	34
3.1.48	Klasse	Menu	34
3.1.49	Klasse	MenuButton	35
3.1.50	Klasse	MenuItem	36
3.1.51	Klasse	MenuScreen	36
3.1.52	Klasse	ModelFactory	37
3.1.53	Klasse	ModelMouseHandler	37
3.1.54	Klasse	MousePointer	37
3.1.55	Klasse	NodeMap	38
3.1.56	Klasse	NodeModel	38
3.1.57	Klasse	NodeModelInfo	39
3.1.58	Klasse	OptionInfo	39
3.1.59	Klasse	Options	40
3.1.60	Klasse	PauseDialog	40
3.1.61	Klasse	PipeModel	40
3.1.62	Klasse	PipeModelInfo	41
3.1.63	Klasse	PipeMovement	41
3.1.64	Klasse	PrinterIO	42
3.1.65	Klasse	ProfileSettingsScreen	42
3.1.66	Klasse	RenderEffect	43
3.1.67	Klasse	RenderEffectStack	44
3.1.68	Klasse	SettingsScreen	44
3.1.69	Klasse	ShadowGameModel	45
3.1.70	Klasse	ShadowGameObject	45
3.1.71	Klasse	SliderItem	46
3.1.72	Klasse	StandardEffect	46
3.1.73	Klasse	StartScreen	47
3.1.74	Klasse	TextInputDialog	47
3.1.75	Klasse	TutorialChallengeMode	47
3.1.76	Klasse	VerticalMenu	47
3.1.77	Klasse	Widget	48
3.1.78	Klasse	WidgetKeyHandler	48
3.1.79	Klasse	WidgetMouseHandler	49
3.1.80	Klasse	World	49
3.1.81	Klasse	XNA.DrawableGameComponent	50

3.1.82	Klasse XNA.Game	50
3.1.83	Klasse XNA.GameComponent	50
3.2	Schnittstellen	51
3.2.1	Schnittstelle IChallengeIO	51
3.2.2	Schnittstelle ICloneable	51
3.2.3	Schnittstelle IEnumerable	51
3.2.4	Schnittstelle IEnumerable1	52
3.2.5	Schnittstelle IEquatable	52
3.2.6	Schnittstelle IEquatable1	52
3.2.7	Schnittstelle IGameObject	52
3.2.8	Schnittstelle IGameScreenComponent	53
3.2.9	Schnittstelle IJunction	54
3.2.10	Schnittstelle IKeyEventListener	54
3.2.11	Schnittstelle IKnotIO	54
3.2.12	Schnittstelle IMouseEventListener	55
3.2.13	Schnittstelle IRenderEffect	55
3.3	Enumerationen	56
4	Abläufe	57
4.1	Sequenzdiagramme	57
5	Klassenindex	58
6	Anmerkungen	59
7	Glossar	60

Kapitel 1

Einleitung

Das Knobel- und Konstruktionsspiel Knot³, 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 Kameraziel 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.

Angles3
+ X : float + Y : float + Z : float + Zero : Angles3
+ FromDegrees (float X, float Y, float Z) : Angles3 + Angles3 (float X, float Y, float Z) : void + ToDegrees (float X, float Y, float Z) : void

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.

Arrow Model
+ Info : Arrow ModelInfo
+ Draw (GameTime gameTime) : void + Intersects (Ray ray) : GameObjectDistance + Arrow Model (GameScreen screen, Arrow ModelInfo info) : void + Update (GameTime gameTime) : void

3.1.3 Klasse ArrowModelInfo

Beschreibung:

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

Arrow ModelInfo
+ Direction : Vector3
+ Arrow ModelInfo (Vector3 position, Vector3 direction) : void

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.

AudioSettingsScreen
settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void

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.

BooleanOptionInfo
+ Value : bool
+ BooleanOptionInfo (String section, String name, String defaultValue, ConfigFile configFile) : void

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-Target 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 BoundingBox 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.

Camera
+ Position : Vector3 + Target : Vector3 + FoV : float + View Matrix : Matrix + WorldMatrix : Matrix + ProjectionMatrix : Matrix + ArcballTarget : Vector3 + View Frustum : BoundingBox - World : World + Rotation : Angles3
+ TargetDirection () : Vector3 + TargetDistance () : float + Camera (GameScreen screen, World world) : void + Update (GameTime gameTime) : void + GetMouseRay (Vector2 mousePosition) : Ray

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 angegebene 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)

Zeichnet das Rendertarget mithilfe eines Outline-Shaders.

public void DrawModel (**GameTime** gameTime, **GameModel** gameModel)

Zeichnet ein 3D-Modell auf das Rendertarget.

public void RemapModel (**GameModel** gameModel)

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

CelShadingEffect
DrawRenderTarget (GameTime) : void + DrawModel (GameTime, GameModel GameModel) : void + RemapModel (GameModel GameModel) : void + CelShadingEffect (GameScreen screen) : void

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 gegebenen Challenge-Metadaten-Objekt. Erstellt ein Challenge-Objekt aus einer gegebenen 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.

Challenge
+ Start : Knot + Target : Knot - highscore : SortedList<Integer, String> - format : IChallengeIO + Highscore : IEnumerator<KeyValuePair<String, Integer>> + Metadata : ChallengeMetadata + Name : String
+ Challenge (ChallengeMetadata meta, Knot start, Knot target) : void + AddToHighscore (String name, Integer time) : void

Eigenschaften:

private KnotStringIO startParser

Der Parser für den Ausgangsknoten im Challenge-Format.

private KnotStringIO targetParser

Der Parser für den Referenzknoten im Challenge-Format.

ChallengeFileIO
- startParser : KnotStringIO - targetParser : KnotStringIO
+ ChallengeFileIO () : void + Save (Challenge challenge) : void + Load (String filename) : Challenge + LoadMetaData (String filename) : ChallengeMetaData

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 angegebenen 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)

ChallengeLoadScreen
+ Update (GameTime time) : void Entered (GameScreen previousScreen, GameTime gameTime) : void

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 IEnumerable<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, IChallengeIO format)

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

ChallengeMetaData
+ Name : String + Start : KnotMetaData + Target : KnotMetaData + Format : IChallengeIO + Filename : String + Highscore : IEnumerable<KeyValuePair<String, Integer>>
+ ChallengeMetaData (String name, KnotMetaData start, KnotMetaData target, String filename, IChallengeIO format)

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.

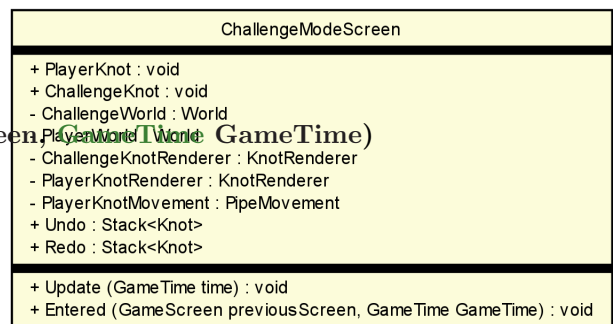
Methoden:

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 verknüpft ist.

Konstruktoren:

public CheckBoxItem (**BooleanOptionInfo** option)

Erstellt einen Menüeintrag für die angegebene Option.

CheckBoxItem
- option : BooleanOptionInfo
+ CheckBoxItem (BooleanOptionInfo option) : void

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.

Circle
+ Content : T + Next : Circle + Previous : Circle
+ Remove () : void + Circle (T content) : void + InsertAfter (T next) : void + InsertBefore (T previous) : void + GetEnumerator () : IEnumerator<T> + GetEnumerator () : IEnumerator

Konstruktoren:

public Circle (**T** content)

Erstellt einen neuen Listeneintrag.

Methoden:

public void Remove ()

Entfernt diesen Listeneintrag und verknüpft 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.

ColorPicker
+ Color : Color
+ OnKeyEvent () : void + Bounds () : Rectangle + OnLeftClick (Vector2 position, ClickState state, GameTime time) : void + OnRightClick (Vector2 position, ClickState state, GameTime time) : void

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.

ColorPickItem
+ Color : Color - picker : ColorPicker

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.

ConfigFile
+ SetOption (String section, String option, String value) : void + GetOption (String section, String option, Boolean defaultValue) : Boolean + GetOption (String section, String option, String defaultValue) : String + SetOption (String section, String option, Boolean _value) : void

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.

ConfirmDialog
- buttons : Menu

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.

ControlSettingsScreen
settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void

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.

CreativeLoadScreen
+ Entered (GameScreen previousScreen, GameTime gameTime) : void + Update (GameTime time) : void

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

CreativeModeScreen
+ Knot : void - World : World - KnotRenderer : KnotRenderer + Undo : Stack<Knot> + Redo : Stack<Knot>
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void

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:

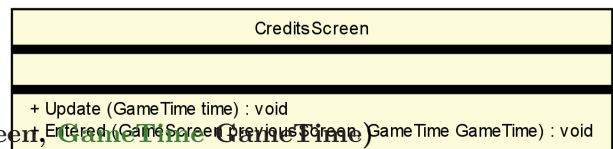
Der Spielzustand, der die Auflistung der Mitwirkenden darstellt.

Methoden:

public void Update (GameTime time)

Wird für jeden Frame aufgerufen.

public void Entered (GameScreen previousScreen, GameTime time)



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.

Dialog
+ Title : String + Text : String
+ OnKeyEvent () : void + Bounds () : Rectangle + OnLeftClick (Vector2 position, ClickState state, GameTime time) : void + OnRightClick (Vector2 position, ClickState state, GameTime time) : void

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.

DistinctOptionInfo
+ ValidValues : HashSet<string> + Value : String
+ DistinctOptionInfo (String section, String name, String defaultValue, IEnumerable<string> validValues)

Konstruktoren:

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

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.

DrawableGameScreenComponent
+ Screen : GameScreen + Index : DisplayLayer
+ SubComponents (GameTime gameTime) : IEnumerable + DrawableGameStateComponent (GameScreen screen, DisplayLayer index) : void

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.

DropDownEntry
+ Text : String

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.

DropDownMenuItem
- dropdown : VerticalMenu
+ AddEntries (DistinctOptionInfo option) : void + AddEntries (DropDownEntry entries) : void

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.

Edge
+ Color : Color + Direction : Direction + Rectangles : List<int>
+ Edge (Direction direction) : void + Get3DDirection () : Vector3

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.

FadeEffect
- IsFinished : Boolean - PreviousRenderTarget : RenderTarget2D
+ FadeEffect (GameScreen newScreen, GameScreen oldScreen) : void # DrawRenderTarget (GameTime) : void

Konstruktoren:

public FadeEffect (GameScreen newScreen, GameScreen oldScreen)

Erstellt einen Überblende-Effekt zwischen den angegebenen Spielzuständen.

Methoden:

protected void DrawRenderTarget (GameTime)

Zeichnet die Überblendung des zuletzt gerenderten Frames des bisherigen Spielzustands in das Rendertarget des aktuellen Spielzustands

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.

FileUtility
+ SettingsDirectory : String + SavegameDirectory : String + ScreenshotDirectory : String
+ ConvertToFileName (String name) : String

Methoden:

public String ConvertToFileName (String name)

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

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 HighlightColor

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.

GameModel
+ Alpha : float + BaseColor : Color + HighlightColor : Color + HighlightIntensity : float + Info : GameModelInfo + Model : XNA.Model + World : World + WorldMatrix : Matrix
+ Center () : Vector3 + Update (GameTime gameTime) : void + Draw (GameTime gameTime) : void + Intersects (Ray ray) : GameObjectDistance + GameModel (GameScreen screen, GameModelInfo info) : void

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.

GameModelInfo
+ Modelname : String + Rotation : Angles3 + Scale : Vector3
+ GameModelInfo (String modelname, Angles3 rotation, Vector3 scale) : void

Konstruktoren:

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

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

3.1.33 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.

GameObjectInfo
+ IsMovable : Boolean + IsSelectable : Boolean + IsVisible : Boolean + Position : Vector3
+ Equals (GameObjectInfo GameObjectInfo) : Boolean + Equals (T other) : Boolean

Methoden:

public Boolean Equals (GameObjectInfo GameObjectInfo)

Vergleicht zwei Informationsobjekte für Spielobjekte.

public Boolean Equals (T other)

Vergleicht zwei Informationsobjekte für Spielobjekte.

3.1.34 Klasse GameScreen

Beschreibung:

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

Eigenschaften:

public Knot3Game Game

Das Spiel, zu dem der Spielzustand gehört.

public Input Input

Der Inpuhandler 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.

GameScreen
+ Game : Knot3Game + Input : Input + PostProcessingEffect : RenderEffect + CurrentRenderEffects : RenderEffectStack
+ Entered (GameScreen previousScreen, GameTime time) : void + BeforeExit (GameScreen nextScreen, GameTime time) : void + Update (GameTime time) : void + GameScreen (Knot3Game game) : void + AddGameComponents (IGameStateComponent[] components) : void + RemoveGameComponents (IGameStateComponent[] components) : void

Konstruktoeren:

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 Inpuhandler 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 Spielkomponenten und deren Unterkomponenten von der Spielkomponentenliste des XNA-Frameworks.

3.1.35 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.

GameScreenComponent
+ Index : DisplayLayer + Screen : GameScreen
+ SubComponents (GameTime gameTime) : IEnumerable + GameStateComponent (GameScreen screen, DisplayLayer index) : void

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.

3.1.36 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.

GraphicsSettingsScreen
settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void

3.1.37 Klasse Input

Beschreibung:

Eigenschaften:

`public ClickState RightMouseButton`

`public ClickState LeftMouseButton`

`public MouseState CurrentMouseState`

`public KeyboardState CurrentKeyboardState`

`public MouseState PreviousMouseState`

`public KeyboardState PreviousKeyboardState`

`public Boolean GrabMouseMovement`

Input
+ RightMouseButton : ClickState + LeftMouseButton : ClickState + CurrentMouseState : MouseState + CurrentKeyboardState : KeyboardState + PreviousMouseState : MouseState + PreviousKeyboardState : KeyboardState + GrabMouseMovement : Boolean
+ Input (GameScreen screen) : void + Update (GameTime time) : void

Konstruktoren:

```
public Input (GameScreen screen)
```

Methoden:

```
public void Update (GameTime time)
```

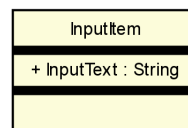
Wird für jeden Frame aufgerufen.

3.1.38 Klasse InputItem

Beschreibung:

Eigenschaften:

```
public String InputText
```

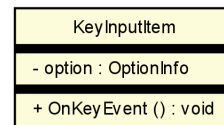


3.1.39 Klasse KeyInputItem

Beschreibung:

Eigenschaften:

```
private OptionInfo option
```



Methoden:

```
public void OnKeyEvent ()
```

3.1.40 Klasse Knot

Beschreibung:

Eigenschaften:

`public String Name`

`private Circle edges`

`public KnotMetaData metaData`

`public Action EdgesChanged`

`public IEnumerable<Edge> SelectedEdges`

Knot
+ Name : String - edges : Circle + metaData : KnotMetaData + EdgesChanged : Action + SelectedEdges : IEnumerable<Edge>
+ Knot () : void + Save () : void + ClearSelection () : void + Knot (KnotMetaData meta, IEnumerable<Edge> edges) : void + IsValidMove (Direction dir, Integer distance) : Boolean + Move (Direction dir, Integer distance) : Boolean + (Knot knotA, Knot knotB) : Boolean + (Knot knotA, Knot knotB) : Boolean + GetEnumerator () : IEnumerator<Edge> + Clone () : Object + AddToSelection (Edge edge) : void + RemoveFromSelection (Edge edge) : void + AddRangeToSelection (Edge edge) : void + IsSelected (Edge edge) : Boolean + GetEnumerator () : IEnumerator + Save (, IKnotIO format) : void

Konstruktoren:

`public Knot ()`

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

Methoden:

`public void Save ()`

`public void ClearSelection ()`

`public Boolean IsValidMove (Direction dir, Integer distance)`

`public Boolean Move (Direction dir, Integer distance)`

`public Boolean (Knot knotA, Knot knotB)`

`public Boolean (Knot knotA, Knot knotB)`


```
public IEnumerator<Edge> GetEnumerator ()
```

```
public Object Clone ()
```

```
public void AddToSelection (Edge edge)
```

```
public void RemoveFromSelection (Edge edge)
```

```
public void AddRangeToSelection (Edge edge)
```

```
public Boolean IsSelected (Edge edge)
```

```
public IEnumerator GetEnumerator ()
```

```
public void Save (, IKnotIO format)
```

3.1.41 Klasse Knot3Game

Beschreibung:

Eigenschaften:

```
public Boolean IsFullScreen
```

```
public Stack<GameScreen> Screens
```

```
public Boolean VSync
```

```
public GraphicsDeviceManager Graphics
```

Knot3Game
+ IsFullScreen : Boolean + Screens : Stack<GameScreen> + VSync : Boolean + Graphics : GraphicsDeviceManager
+ Game () : void + Initialize () : void + LoadContent () : void + UnloadContent () : void + Draw (GameTime) : void + Update (GameTime) : void

Methoden:

```
public void Game ()
```

```
public void Initialize ()
```

```
public void LoadContent ()
```

```
public void UnloadContent ()
```

```
public void Draw (GameTime)
```

```
public void Update (GameTime)
```

3.1.42 Klasse KnotFileIO

Beschreibung:

Eigenschaften:

```
private KnotStringIO parser
```

Konstruktoren:

```
public KnotFileIO ()
```

Methoden:

```
public void Save (Knot knot)
```

```
public Knot Load (String filename)
```

```
public KnotMetaData LoadMetaData (String filename)
```

KnotFileIO
- parser : KnotStringIO
+ KnotFileIO () : void + Save (Knot knot) : void + Load (String filename) : Knot + LoadMetaData (String filename) : KnotMetaData

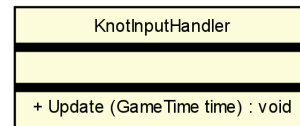
3.1.43 Klasse KnotInputHandler

Beschreibung:

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.



3.1.44 Klasse KnotMetaData

Beschreibung:

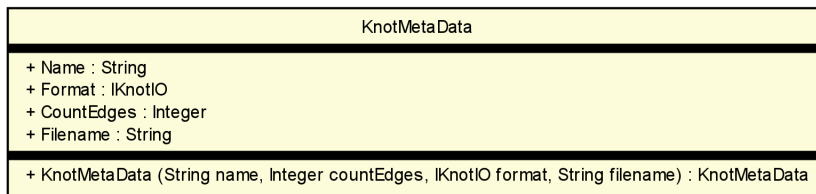
Eigenschaften:

public String Name

public IKnotIO Format

public Integer CountEdges

public String Filename



Konstruktoren:

public KnotMetaData (**String** name, **Integer** countEdges, **IKnotIO** format, **String** filename)

3.1.45 Klasse KnotRenderer

Beschreibung:

Eigenschaften:

public GameObjectInfo Info

public World World

private List<ArrowModel> arrows

private List<NodeModel> nodes

private List<PipeModel> pipes

public Knot Knot

private ModelFactory pipeFactory

private ModelFactory nodeFactory

private ModelFactory arrowFactory

Methoden:

public Vector3 Center ()

public GameObjectDistance Intersects (Ray Ray)

public void OnEdgesChanged ()

public void ModelRenderer (GameScreen screen, GameObjectInfo info)

public void Update (GameTime gameTime)

KnotRenderer
+ Info : GameObjectInfo + World : World - arrows : List<Arrow Model> - nodes : List<NodeModel> - pipes : List<PipeModel> + Knot : Knot - pipeFactory : ModelFactory - nodeFactory : ModelFactory - arrow Factory : ModelFactory
+ Center () : Vector3 + Intersects (Ray Ray) : GameObjectDistance + OnEdgesChanged () : void + ModelRenderer (GameScreen screen, GameObjectInfo info) : void + Update (GameTime gameTime) : void + Draw (GameTime gameTime) : void + GetEnumerator () : IEnumerator

Wird für jeden Frame aufgerufen.

```
public void Draw (GameTime gameTime)
```

```
public IEnumerator GetEnumerator ()
```

3.1.46 Klasse KnotStringIO

Beschreibung:

Konstruktoeren:

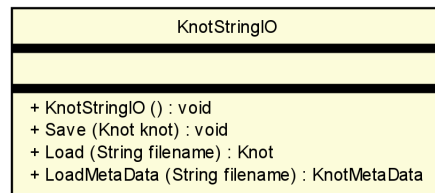
```
public KnotStringIO ()
```

Methoden:

```
public void Save (Knot knot)
```

```
public Knot Load (String filename)
```

```
public KnotMetaData LoadMetaData (String filename)
```

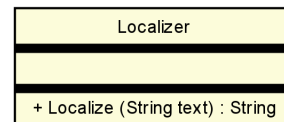


3.1.47 Klasse Localizer

Beschreibung:

Methoden:

```
public String Localize (String text)
```



3.1.48 Klasse Menu

Beschreibung:

Eigenschaften:

`public String Name`

`public Func<int, Vector2> RelativeItemSize`

`public Func<int, Vector2> RelativeItemPosition`

`public Func<ItemState, Vector2> ItemForegroundColor`

`public Func<ItemState, Vector2> ItemBackgroundColor`

`public HorizontalAlignment ItemAlignX`

`public VerticalAlignment ItemAlignY`

Methoden:

`public void Add (MenuItem item)`

`public void Delete (MenuItem item)`

`public MenuItem GetItem (Integer i)`

`public Integer Size ()`

`public IEnumerable GetEnumerator ()`

Menu
+ Name : String + RelativeItemSize : Func<int, Vector2> + RelativeItemPosition : Func<int, Vector2> + ItemForegroundColor : Func<ItemState, Vector2> + ItemBackgroundColor : Func<ItemState, Vector2> + ItemAlignX : HorizontalAlignment + ItemAlignY : VerticalAlignment
+ Add (MenuItem item) : void + Delete (MenuItem item) : void + GetItem (Integer i) : MenuItem + Size () : Integer + GetEnumerator () : IEnumerable

3.1.49 Klasse MenuButton

Beschreibung:

Eigenschaften:

`public String Name`

MenuButton
+ Name : String
+ MenuButton (String name) : void

Konstruktoren:

`public MenuButton (String name)`

3.1.50 Klasse MenuItem

Beschreibung:

Eigenschaften:

`public ItemState ItemState`

`public Integer ItemOrder`

`public String Text`

MenuItem
+ ItemState : ItemState + ItemOrder : Integer + Text : String
+ OnKeyEvent () : void + OnLeftClick (Vector2 position, ClickState state, gameTime time) : void + OnRightClick (Vector2 position, ClickState state, gameTime time) : void + Bounds () : Rectangle

Methoden:

`public void OnKeyEvent ()`

`public void OnLeftClick (Vector2 position, ClickState state, gameTime time)`

`public void OnRightClick (Vector2 position, ClickState state, gameTime time)`

`public Rectangle Bounds ()`

3.1.51 Klasse MenuScreen

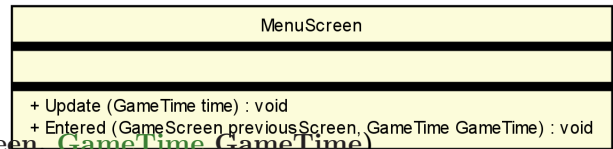
Beschreibung:

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.

public void Entered (**GameScreen** previousScreen, **GameTime** gameTime)



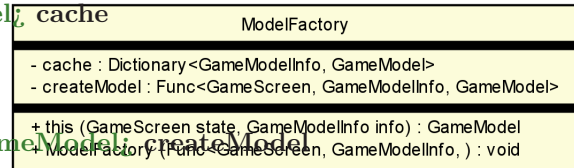
3.1.52 Klasse ModelFactory

Beschreibung:

Eigenschaften:

private Dictionary<GameModelInfo, GameModel> cache

private Func<GameScreen, GameModelInfo, GameModel> createModel



Konstruktoren:

public ModelFactory (**Func<GameScreen, GameModelInfo, >**)

Methoden:

public GameModel this (**GameScreen** state, **GameModelInfo** info)

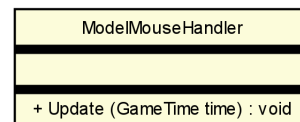
3.1.53 Klasse ModelMouseHandler

Beschreibung:

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.

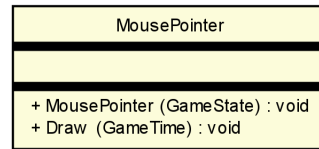


3.1.54 Klasse MousePointer

Beschreibung:

Konstruktoren:

`public` `MousePointer` (`GameState`)



Methoden:

`public void` `Draw` (`GameTime`)

3.1.55 Klasse NodeMap

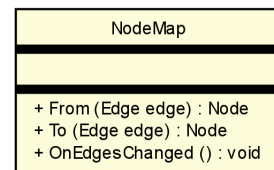
Beschreibung:

Methoden:

`public` `Node` `From` (`Edge` `edge`)

`public` `Node` `To` (`Edge` `edge`)

`public void` `OnEdgesChanged` ()



3.1.56 Klasse NodeModel

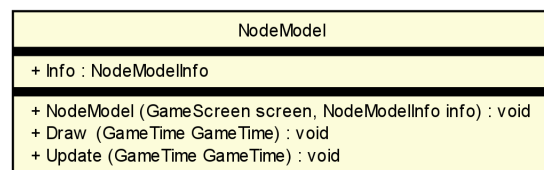
Beschreibung:

Eigenschaften:

`public` `NodeModelInfo` `Info`

Konstruktoren:

`public` `NodeModel` (`GameScreen` `screen`, `NodeModelInfo` `info`)



Methoden:

`public void` `Draw` (`GameTime` `GameTime`)

`public void` `Update` (`GameTime` `GameTime`)

Wird für jeden Frame aufgerufen.

3.1.57 Klasse NodeModelInfo

Beschreibung:

Eigenschaften:

`public Edge EdgeFrom`

`public Edge EdgeTo`

`public Knot Knot`

`public Vector3 Position`

NodeModelInfo
+ EdgeFrom : Edge + EdgeTo : Edge + Knot : Knot + Position : Vector3
+ NodeModelInfo (Knot knot, Edge from, Edge to) : void

Konstruktoren:

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

3.1.58 Klasse OptionInfo

Beschreibung:

Eigenschaften:

`private ConfigFile configFile`

`public String Section`

`public String Name`

`public String DefaultValue`

`public String Value`

OptionInfo
- configFile : ConfigFile + Section : String + Name : String + DefaultValue : String + Value : String
+ OptionInfo (String section, String name, String defaultValue, ConfigFile configFile) : void

Konstruktoren:

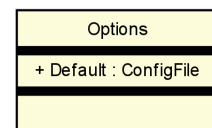
`public OptionInfo (String section, String name, String defaultValue, ConfigFile configFile)`

3.1.59 Klasse Options

Beschreibung:

Eigenschaften:

`public ConfigFile Default`

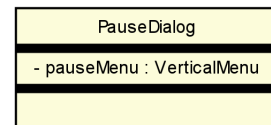


3.1.60 Klasse PauseDialog

Beschreibung:

Eigenschaften:

`private VerticalMenu pauseMenu`

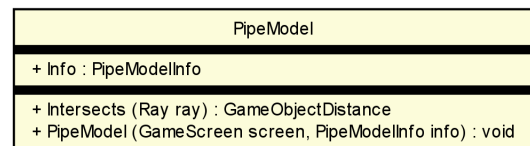


3.1.61 Klasse PipeModel

Beschreibung:

Eigenschaften:

`public PipeModelInfo Info`



Konstruktoren:

`public PipeModel (GameScreen screen, PipeModelInfo info)`

Methoden:

`public GameObjectDistance Intersects (Ray ray)`

3.1.62 Klasse PipeModelInfo

Beschreibung:

Eigenschaften:

public **Edge** Edge

public **Knot** Knot

public **Vector3** PositionFrom

public **Vector3** PositionTo

PipeModelInfo
+ Edge : Edge + Knot : Knot + PositionFrom : Vector3 + PositionTo : Vector3
+ PipeModelInfo (Knot knot, Edge edge) : void

Konstruktoren:

public PipeModelInfo (**Knot** knot, **Edge** edge)

3.1.63 Klasse PipeMovement

Beschreibung:

Eigenschaften:

public **GameObjectInfo** Info

public **Knot** Knot

public **World** World

PipeMovement
+ Info : GameObjectInfo + Knot : Knot + World : World
+ Center () : Vector3 + Intersects (Ray Ray) : GameObjectDistance + Update (GameTime GameTime) : void + PipeMovement (GameState, World, GameObjectInfo) : void + GetEnumerator () : IEnumerator + Draw (GameTime GameTime) : void

Konstruktoren:

public PipeMovement (**GameState**, **World**, **GameObjectInfo**)

Methoden:

```
public Vector3 Center ()
```

```
public GameObjectDistance Intersects (Ray Ray)
```

```
public void Update (GameTime gameTime)
```

Wird für jeden Frame aufgerufen.

```
public IEnumerator GetEnumerator ()
```

```
public void Draw (GameTime gameTime)
```

3.1.64 Klasse PrinterIO

Beschreibung:

Konstruktoeren:

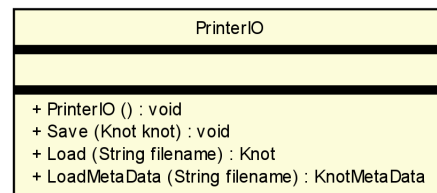
```
public PrinterIO ()
```

Methoden:

```
public void Save (Knot knot)
```

```
public Knot Load (String filename)
```

```
public KnotMetaData LoadMetaData (String filename)
```



3.1.65 Klasse ProfileSettingsScreen

Beschreibung:

Eigenschaften:

`protected void settingsMenu`

ProfileSettingsScreen
settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime GameTime) : void

Methoden:

`public void Update (GameTime time)`

Wird für jeden Frame aufgerufen.

`public void Entered (GameScreen previousScreen, GameTime GameTime)`

3.1.66 Klasse RenderEffect

Beschreibung:

Eigenschaften:

`public RenderTarget2D RenderTarget`

`protected GameScreen screen`

`protected SpriteBatch spriteBatch`

RenderEffect
+ RenderTarget : RenderTarget2D # screen : GameScreen # spriteBatch : SpriteBatch
+ Begin (GameTime) : void + End (GameTime) : void + Draw Model (GameTime, GameModel GameModel) : void + RemapModel (GameModel GameModel) : void # Draw RenderTarget (GameTime) : void

Methoden:

`public void Begin (GameTime)`

`public void End (GameTime)`

`public void DrawModel (GameTime, GameModel GameModel)`

`public void RemapModel (GameModel GameModel)`

`protected void DrawRenderTarget (GameTime)`

3.1.67 Klasse RenderEffectStack

Beschreibung:

Eigenschaften:

public **IRenderEffect** CurrentEffect

private **IRenderEffect** DefaultEffect

RenderEffectStack
+ CurrentEffect : IRenderEffect - DefaultEffect : IRenderEffect
+ Pop () : IRenderEffect + Push (IRenderEffect effect) : void + RenderEffectStack (IRenderEffect defaultEffect) : void

Konstruktoeren:

public RenderEffectStack (**IRenderEffect** defaultEffect)

Methoden:

public **IRenderEffect** Pop ()

public void Push (**IRenderEffect** effect)

3.1.68 Klasse SettingsScreen

Beschreibung:

Eigenschaften:

protected void navigation

SettingsScreen
navigation : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime time) : void

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.

public void Entered (**GameScreen** previousScreen, **GameTime** time)

3.1.69 Klasse ShadowGameModel

Beschreibung:

Eigenschaften:

`public Color ShadowColor`

`public float ShadowAlpha`

Shadow GameModel
+ Shadow Color : Color + Shadow Alpha : float
+ Shadow GameModel (GameScreen screen, GameModel decoratedModel) : void + Draw (GameTime GameTime) : void

Konstruktoren:

`public ShadowGameModel (GameScreen screen, GameModel decoratedModel)`

Methoden:

`public void Draw (GameTime GameTime)`

3.1.70 Klasse ShadowGameObject

Beschreibung:

Eigenschaften:

`public GameObjectInfo Info`

`public World World`

`public Vector3 ShadowPosition`

`public Vector3 OriginalPosition`

Shadow GameObject
+ Info : GameObjectInfo + World : World + Shadow Position : Vector3 + OriginalPosition : Vector3
+ Center () : Vector3 + Update (GameTime GameTime) : void + Draw (GameTime GameTime) : void + Intersects (Ray Ray) : GameObjectDistance + Shadow GameObject (GameScreen screen, IGameObject decoratedObj) : void

Konstruktoren:

`public ShadowGameObject (GameScreen screen, IGameObject decoratedObj)`

Methoden:

```
public Vector3 Center ()
```

```
public void Update (GameTime gameTime)
```

Wird für jeden Frame aufgerufen.

```
public void Draw (GameTime gameTime)
```

```
public GameObjectDistance Intersects (Ray ray)
```

3.1.71 Klasse SliderItem

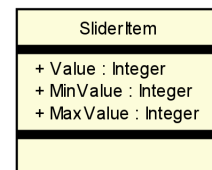
Beschreibung:

Eigenschaften:

```
public Integer Value
```

```
public Integer MinValue
```

```
public Integer MaxValue
```

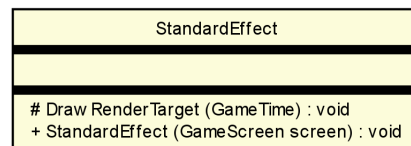


3.1.72 Klasse StandardEffect

Beschreibung:

Konstruktoren:

```
public StandardEffect (GameScreen screen)
```



Methoden:

```
protected void DrawRenderTarget (GameTime)
```

3.1.73 Klasse StartScreen

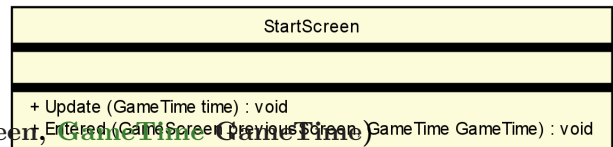
Beschreibung:

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.

public void Entered (**GameScreen** previousScreen, **GameTime** gameTime)

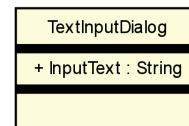


3.1.74 Klasse TextInputDialog

Beschreibung:

Eigenschaften:

public String InputText

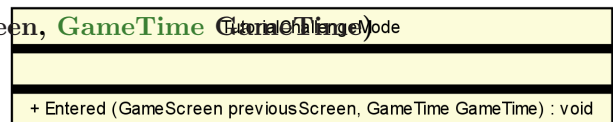


3.1.75 Klasse TutorialChallengeMode

Beschreibung:

Methoden:

public void Entered (**GameScreen** previousScreen, **GameTime** gameTime)

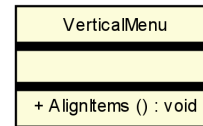


3.1.76 Klasse VerticalMenu

Beschreibung:

Methoden:

`public void AlignItems ()`



3.1.77 Klasse Widget

Beschreibung:

Eigenschaften:

`public Vector2 RelativeSize`

`public Vector2 RelativePosition`

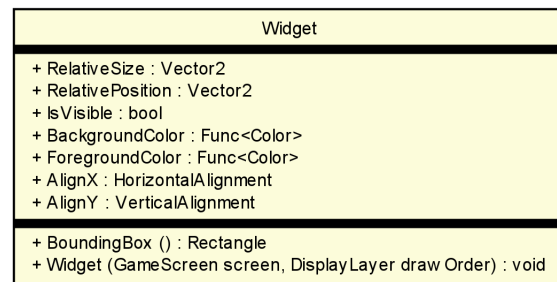
`public bool IsVisible`

`public Func<Color> BackgroundColor`

`public Func<Color> ForegroundColor`

`public HorizontalAlignment AlignX`

`public VerticalAlignment AlignY`



Konstruktoren:

`public Widget (GameScreen screen, DisplayLayer drawOrder)`

Methoden:

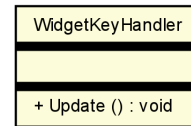
`public Rectangle BoundingBox ()`

3.1.78 Klasse WidgetKeyHandler

Beschreibung:

Methoden:

```
public void Update ()
```

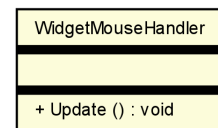


3.1.79 Klasse WidgetMouseHandler

Beschreibung:

Methoden:

```
public void Update ()
```



3.1.80 Klasse World

Beschreibung:

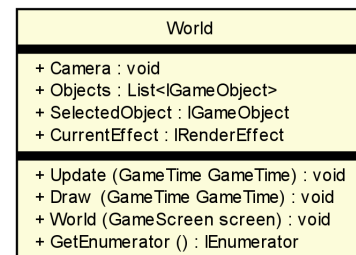
Eigenschaften:

```
public void Camera
```

```
public List<IGameObject> Objects
```

```
public IGameObject SelectedObject
```

```
public IRenderEffect CurrentEffect
```



Konstruktoren:

```
public World (GameScreen screen)
```

Methoden:

```
public void Update (GameTime gameTime)
```

Wird für jeden Frame aufgerufen.

```
public void Draw (GameTime gameTime)
```

```
public IEnumerator GetEnumerator ()
```

3.1.81 Klasse XNA.DrawableGameComponent

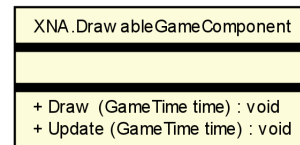
Beschreibung:

Methoden:

`public void Draw (GameTime time)`

`public void Update (GameTime time)`

Wird für jeden Frame aufgerufen.



3.1.82 Klasse XNA.Game

Beschreibung:

Methoden:

`public void Game ()`

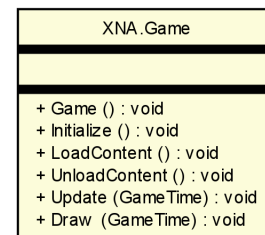
`public void Initialize ()`

`public void LoadContent ()`

`public void UnloadContent ()`

`public void Update (GameTime)`

`public void Draw (GameTime)`



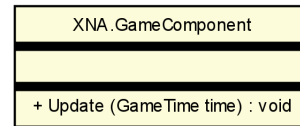
3.1.83 Klasse XNA.GameComponent

Beschreibung:

Methoden:

public void Update (**GameTime** time)

Wird für jeden Frame aufgerufen.



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)

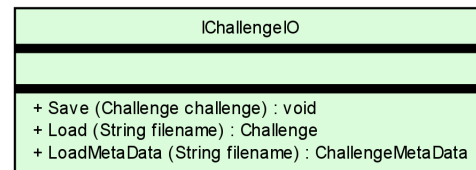
Speichert eine Challenge.

public Challenge Load (**String** filename)

Lädt eine Challenge.

public ChallengeMetaData LoadMetaData (**String** filename)

Lädt die Metadaten einer Challenge.

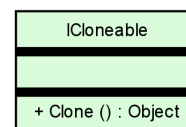


3.2.2 Schnittstelle ICloneable

Beschreibung:

Methoden:

public Object Clone ()

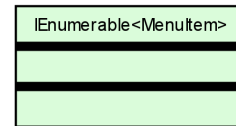


3.2.3 Schnittstelle IEnumerable

Beschreibung:

Methoden:

`public IEnumerator GetEnumerator ()`

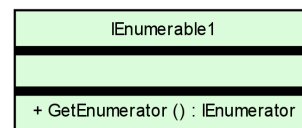


3.2.4 Schnittstelle IEnumerable1

Beschreibung:

Methoden:

`public IEnumerator GetEnumerator ()`

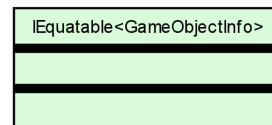


3.2.5 Schnittstelle IEquatable

Beschreibung:

Methoden:

`public Boolean Equals (T other)`

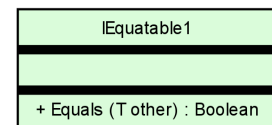


3.2.6 Schnittstelle IEquatable1

Beschreibung:

Methoden:

`public Boolean Equals (T other)`



3.2.7 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.

IGameObject
+ Info : GameObjectInfo + World : World
+ Center () : Vector3 + Update (GameTime time) : void + Draw (GameTime time) : void + Intersects (Ray ray) : GameObjectDistance

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.

3.2.8 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.

IGameScreenComponent
+ Index : DisplayLayer + Screen : GameScreen
+ SubComponents (GameTime time) : IEnumerable

Methoden:

public IEnumerable SubComponents (GameTime time)

Gibt Spielkomponenten zurück, die in dieser Spielkomponente enthalten sind.

3.2.9 Schnittstelle IJunction

Beschreibung:

Repräsentiert einen Übergang zwischen zwei Kanten.

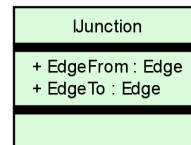
Eigenschaften:

public Edge EdgeFrom

Die Kante vor dem Übergang.

public Edge EdgeTo

Die Kante nach dem Übergang.



3.2.10 Schnittstelle IKeyEventListe- ner

Beschreibung:

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

Eigenschaften:

public DisplayLayer Index

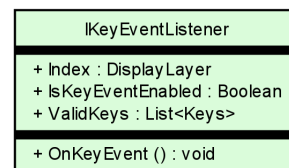
Die Eingabepriorität.

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.

3.2.11 Schnittstelle IKnotIO

Beschreibung:

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

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.

IKnotIO
+ Save (Knot knot) : void + Load (String filename) : Knot + LoadMetaData (String filename) : KnotMetaData

3.2.12 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.

IMouseEventListener
+ Index : DisplayLayer + IsMouseEventEnabled : Boolean
+ Bounds () : Rectangle + OnLeftClick (Vector2 position, ClickState state, GameTime time) : void + OnRightClick (Vector2 position, ClickState state, GameTime time) : void

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.

3.2.13 Schnittstelle IRenderEffect

Beschreibung:

Eigenschaften:

`public RenderTarget2D RenderTarget`

Methoden:

`public void Begin (GameTime)`

`public void End (GameTime)`

`public void DrawModel (GameTime, GameModel model)`

`public void RemapModel (GameModel model)`

IRenderEffect
+ RenderTarget : RenderTarget2D
+ Begin (GameTime) : void + End (GameTime) : void + Draw Model (GameTime, GameModel model) : void + RemapModel (GameModel model) : void

3.3 Enumerationen

Kapitel 4

Abläufe

4.1 Sequenzdiagramme

Kapitel 5

Klassenindex

Kapitel 6

Anmerkungen

Kapitel 7

Glossar

Test	Test
------	------