

# **ENTWURFSDOKUMENT**

**(V. 1.0)**

**KNOT<sup>3</sup>**  
**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

19. Dezember 2013

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
<b>2</b>	<b>Aufbau</b>	<b>6</b>
2.1	Architektur . . . . .	6
2.2	Klassendiagramm . . . . .	7
2.3	Verwendete Entwurfsmuster . . . . .	7
<b>3</b>	<b>Klassenübersicht</b>	<b>8</b>
3.1	Klassen . . . . .	8
3.1.1	Klasse Angles3 . . . . .	8
3.1.2	Klasse ArrowModel . . . . .	9
3.1.3	Klasse ArrowModelInfo . . . . .	9
3.1.4	Klasse AudioSettingsScreen . . . . .	10
3.1.5	Klasse BooleanOptionInfo . . . . .	11
3.1.6	Klasse Camera . . . . .	11
3.1.7	Klasse CelShadingEffect . . . . .	13
3.1.8	Klasse Challenge . . . . .	14
3.1.9	Klasse ChallengeFileIO . . . . .	15
3.1.10	Klasse ChallengeLoadScreen . . . . .	15
3.1.11	Klasse ChallengeMetaData . . . . .	16
3.1.12	Klasse ChallengeModeScreen . . . . .	17
3.1.13	Klasse CheckBoxItem . . . . .	18
3.1.14	Klasse Circle . . . . .	19
3.1.15	Klasse ColorPicker . . . . .	20
3.1.16	Klasse ColorPickItem . . . . .	20
3.1.17	Klasse ConfigFile . . . . .	21
3.1.18	Klasse ConfirmDialog . . . . .	22
3.1.19	Klasse ControlSettingsScreen . . . . .	22
3.1.20	Klasse CreativeLoadScreen . . . . .	23
3.1.21	Klasse CreativeModeScreen . . . . .	23
3.1.22	Klasse CreditsScreen . . . . .	24
3.1.23	Klasse Dialog . . . . .	25
3.1.24	Klasse DistinctOptionInfo . . . . .	26
3.1.25	Klasse DrawableGameScreenComponent . . . . .	26
3.1.26	Klasse DropDownEntry . . . . .	27
3.1.27	Klasse DropDownMenuItem . . . . .	27
3.1.28	Klasse Edge . . . . .	28
3.1.29	Klasse FadeEffect . . . . .	29
3.1.30	Klasse FileUtility . . . . .	29
3.1.31	Klasse GameModel . . . . .	30
3.1.32	Klasse GameModelInfo . . . . .	32

3.1.33	Klasse	GameObjectDistance . . . . .	32
3.1.34	Klasse	GameObjectInfo . . . . .	33
3.1.35	Klasse	GameScreen . . . . .	33
3.1.36	Klasse	GameScreenComponent . . . . .	35
3.1.37	Klasse	GraphicsSettingsScreen . . . . .	35
3.1.38	Klasse	Input . . . . .	36
3.1.39	Klasse	InputItem . . . . .	37
3.1.40	Klasse	KeyInputItem . . . . .	38
3.1.41	Klasse	Knot . . . . .	38
3.1.42	Klasse	Knot3Game . . . . .	41
3.1.43	Klasse	KnotFileIO . . . . .	42
3.1.44	Klasse	KnotInputHandler . . . . .	43
3.1.45	Klasse	KnotMetaData . . . . .	43
3.1.46	Klasse	KnotRenderer . . . . .	45
3.1.47	Klasse	KnotStringIO . . . . .	47
3.1.48	Klasse	Localizer . . . . .	47
3.1.49	Klasse	Menu . . . . .	48
3.1.50	Klasse	MenuButton . . . . .	49
3.1.51	Klasse	MenuItem . . . . .	50
3.1.52	Klasse	MenuScreen . . . . .	51
3.1.53	Klasse	ModelFactory . . . . .	51
3.1.54	Klasse	ModelMouseHandler . . . . .	52
3.1.55	Klasse	MousePointer . . . . .	52
3.1.56	Klasse	NodeMap . . . . .	53
3.1.57	Klasse	NodeModel . . . . .	53
3.1.58	Klasse	NodeModelInfo . . . . .	54
3.1.59	Klasse	OptionInfo . . . . .	55
3.1.60	Klasse	Options . . . . .	55
3.1.61	Klasse	PauseDialog . . . . .	56
3.1.62	Klasse	PipeModel . . . . .	56
3.1.63	Klasse	PipeModelInfo . . . . .	57
3.1.64	Klasse	PipeMovement . . . . .	57
3.1.65	Klasse	PrinterIO . . . . .	58
3.1.66	Klasse	ProfileSettingsScreen . . . . .	59
3.1.67	Klasse	RenderEffect . . . . .	60
3.1.68	Klasse	RenderEffectStack . . . . .	61
3.1.69	Klasse	SettingsScreen . . . . .	62
3.1.70	Klasse	ShadowGameModel . . . . .	62
3.1.71	Klasse	ShadowGameObject . . . . .	63
3.1.72	Klasse	SliderItem . . . . .	64
3.1.73	Klasse	StandardEffect . . . . .	65
3.1.74	Klasse	StartScreen . . . . .	65
3.1.75	Klasse	TextInputDialog . . . . .	66
3.1.76	Klasse	TutorialChallengeModeScreen . . . . .	67
3.1.77	Klasse	VerticalMenu . . . . .	67
3.1.78	Klasse	Widget . . . . .	67
3.1.79	Klasse	WidgetKeyHandler . . . . .	68
3.1.80	Klasse	WidgetMouseHandler . . . . .	69
3.1.81	Klasse	World . . . . .	69

3.2	Schnittstellen . . . . .	70
3.2.1	Schnittstelle IChallengeIO . . . . .	70
3.2.2	Schnittstelle IGameObject . . . . .	71
3.2.3	Schnittstelle IGameScreenComponent . . . . .	71
3.2.4	Schnittstelle IJunction . . . . .	72
3.2.5	Schnittstelle IKeyEventListener . . . . .	72
3.2.6	Schnittstelle IKnotIO . . . . .	73
3.2.7	Schnittstelle IMouseEventListener . . . . .	74
3.2.8	Schnittstelle IRenderEffect . . . . .	74
3.3	Enumerationen . . . . .	75
3.3.1	Enumeration ClickState . . . . .	75
3.3.2	Enumeration Direction . . . . .	76
3.3.3	Enumeration DisplayLayer . . . . .	77
3.3.4	Enumeration HorizontalAlignment . . . . .	78
3.3.5	Enumeration ItemState . . . . .	78
3.3.6	Enumeration VerticalAlignment . . . . .	79
<b>4</b>	<b>Abläufe</b>	<b>80</b>
4.1	Sequenzdiagramme . . . . .	80
<b>5</b>	<b>Klassenindex</b>	<b>81</b>
<b>6</b>	<b>Anmerkungen</b>	<b>82</b>
<b>7</b>	<b>Glossar</b>	<b>83</b>

# 1 Einleitung

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

## 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 implementiert 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

## 3 Klassenübersicht

### 3.1 Klassen

#### 3.1.1 Klasse `Angles3`

##### Beschreibung:

Diese Klasse repräsentiert die Rotationswinkel der drei Achsen X, Y und Z. Sie bietet Möglichkeit vordefinierte Winkelwerte zu verwenden, z.B. stellt Zero den Nullvektor dar. Die Umwandlung zwischen verschiedenen Winkelmaßen wie Grad- und Bogenmaß unterstützt sie durch entsprechende Methoden.

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

##### 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 Eigenschaft 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 ArrowModel repräsentiert ein 3D-Modell für einen Pfeil, zum Einblenden an selektierten Kanten (s. Edge).

ArrowModel
+ Info :ArrowModelInfo
+ Draw (GameTime gameTime) : void + Intersects (Ray ray) : GameObjectDistance + ArrowModel (GameScreen screen, ArrowModelInfo info) : void + Update (GameTime gameTime) : void

#### Eigenschaften:

**public** **ArrowModelInfo** Info

Das Info-Objekt, das die Position und Richtung des ArrowModel's 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 der Klasse ArrowModelInfo hält alle Informationen, die zur Erstellung eines Pfeil-3D-Modelles (s. ArrowModel) notwendig sind.

ArrowModelInfo
+ Direction : Vector3
+ ArrowModelInfo (Vector3 position, Vector3 direction) : void

### Eigenschaften:

**public** Vector3 Direction

Gibt die Richtung, in die der Pfeil zeigen soll an.

### Konstruktoren:

**public** ArrowModelInfo (Vector3 position, Vector3 direction)

Erstellt ein neues ArrowModelInfo-Objekt an einer bestimmten Position position im 3D-Raum. Dieses zeigt in eine durch direction bestimmte Richtung.

## 3.1.4 Klasse AudioSettingsScreen

### Beschreibung:

Die Klasse AudioSettingsScreen steht für den Spielzustand, der die Audio-Einstellungen repräsentiert.

AudioSettingsScreen
# settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + AudioSettingsScreen (Knot3Game game) : void

### Eigenschaften:

**protected** void settingsMenu

Das Menü, das die Einstellungen enthält.

### Konstruktoren:

**public** AudioSettingsScreen (Knot3Game game)

Erzeugt ein neues AudioSettingsScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

### 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.

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

#### 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.

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

### 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 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)

!!!

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

Zeichnet ein 3D-Modell auf das Rendertarget.

CelShadingEffect
<pre># DrawRenderTarget (GameTime gameTime) : void + DrawModel (GameTime, GameModel gameModel) : void + RemapModel (GameModel gameModel) : void + CelShadingEffect (GameScreen screen) : void</pre>

**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:

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

**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.

**Konstruktoren:**

ChallengeFileIO
+ ChallengeFileIO () : void + Save (Challenge challenge) : void + Load (String filename) : Challenge + LoadMetaData (String filename) : ChallengeMetaData

**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.

ChallengeLoadScreen
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + ChallengeModeScreen (Knot3Game game) : void

## 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.

**public void** ChallengeModeScreen (**Knot3Game** game)

Erzeugt ein neues ChallengeLoadScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

### 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.

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



**public** **IEnumerator**<**KeyValuePair**<**String**, **Integer**>> **Highscore**

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

#### Konstruktooren:

**public** **ChallengeMetaData** (**String** name, **KnotMetaData** start, **KnotMetaDa-**  
**ta** 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

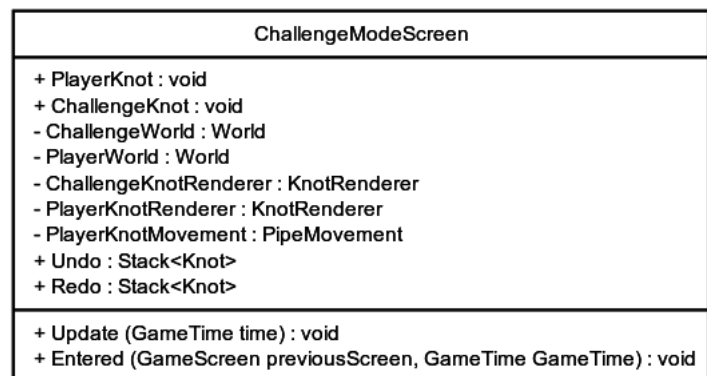
#### Beschreibung:

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

#### Eigenschaften:

**public**  
**id**  
**Knot**

**vo-**  
**Player-**



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 Inpuhandler, 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 Inpuhandler 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 (GameScreen screen, DisplayLayer drawOrder, BooleanOptionInfo option)**

Erzeugt ein neues CheckBoxItem-Objekt und initialisiert dieses mit dem zugehörigen

CheckBoxItem
- option : BooleanOptionInfo
+ CheckBoxItem (GameScreen screen, DisplayLayer drawOrder, BooleanOptionInfo option) : void

GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge und der Auswahloption Pflicht.

### 3.1.14 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

### 3.1.15 Klasse ColorPickItem

#### Beschreibung:

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

ColorPickItem
+ Color : Color - picker : ColorPicker
+ ColorPickItem (GameScreen screen, DisplayLayer drawOrder, Color color) : void

### Eigenschaften:

**public** **Color** Color

Die aktuelle Farbe.

**private** **ColorPicker** picker

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

### Konstruktoren:

**public** ColorPickItem (**GameScreen** screen, **DisplayLayer** drawOrder, **Color** color)

Erzeugt ein neues ColorPickItem-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge und der Auswahloption Pflicht.

## 3.1.16 Klasse ConfigFile

### Beschreibung:

Repräsentiert eine Einstellungsdatei.

### Methoden:

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** **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.

### 3.1.17 Klasse ConfirmDialog

#### Beschreibung:

Ein Dialog, der Schaltflächen zum Bestätigen einer Aktion anzeigt.

ConfirmDialog
- buttons : Menu
+ ConfirmDialog (GameScreen screen, DisplayLayer drawOrder, String title, String text) : void

#### Eigenschaften:

`private Menu buttons`

Das Menü, das Schaltflächen enthält.

#### Konstruktoren:

`public ConfirmDialog (GameScreen screen, DisplayLayer drawOrder, String title, String text)`

Erzeugt ein neues ConfirmDialog-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge, einer Zeichenkette für den Titel und für den eingeblendeten Text Pflicht.

### 3.1.18 Klasse ControlSettingsScreen

#### Beschreibung:

Der Spielzustand, der die Steuerungseinstellungen darstellt.

ControlSettingsScreen
# settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + ControlSettingsScreen (Knot3Game game) : void

#### Eigenschaften:

`protected void settingsMenu`

Das Menü, das die Einstellungen enthält.

#### Konstruktoren:

`public ControlSettingsScreen (Knot3Game game)`

Erzeugt ein neues ControlSettingsScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

#### 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.

### 3.1.19 Klasse CreativeLoadScreen

#### Beschreibung:

Der Spielzustand, der den Ladebildschirm für Knoten darstellt.

#### Konstruktoren:

CreativeLoadScreen
+ Entered (GameScreen previousScreen, GameTime gameTime) : void + Update (GameTime time) : void + CreativeLoadScreen (Knot3Game game) : void

**public** CreativeLoadScreen (**Knot3Game** game)

Erzeugt ein neues CreativeLoadScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

#### 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.

### 3.1.20 Klasse CreativeModeScreen

#### Beschreibung:

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

#### Eigenschaften:

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

**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.

### 3.1.21 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.

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

### 3.1.22 Klasse Dialog

#### Beschreibung:

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

#### Eigenschaften:

**public**                **String**                **Tit-**  
**le**

Der Fenstertitel.

**public** **String** **Text**

Der angezeigte Text.

#### Konstruktoren:

**public** **Dialog** (**GameScreen** screen, **DisplayLayer** drawOrder, **String** title, **String** text)

Erzeugt ein neues Dialog-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge, einer Zeichenkette für den Titel und für den eingeblendeten Text Pflicht.

#### 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)

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 + Dialog (GameScreen screen, DisplayLayer drawOrder, String title, String text) : void



Bei einem Rechtsklick geschieht nichts.

### 3.1.23 Klasse DistinctOptionInfo

#### Beschreibung:

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

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

#### Eigenschaften:

**public** HashSet<string> ValidValues

Eine Menge von Texten, welche die für die Option gültigen Werte beschreiben.

**public** String Value

Ein Property, das den aktuell abgespeicherten Wert zurück gibt.

#### Konstruktoren:

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

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

### 3.1.24 Klasse DrawableGameScreenComponent

#### Beschreibung:

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

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

#### 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.

### 3.1.25 Klasse DropDownEntry

#### Beschreibung:

Repräsentiert einen Eintrag in einem Dropdown-Menü.

DropDownEntry
+ Text : String

#### Eigenschaften:

**public String** Text

Der Text des Eintrags.

### 3.1.26 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.

DropDownMenuItem
- dropdown : VerticalMenu
+ AddEntries (DistinctOptionInfo option) : void + AddEntries (DropDownEntry enties) : void + DropDownMenuItem (GameScreen screen, DisplayLayer drawOrder) : void

#### Eigenschaften:

**private VerticalMenu** dropdown

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

#### Konstruktoren:

**public** DropDownMenuItem (**GameScreen** screen, **DisplayLayer** drawOrder)

Erzeugt ein neues `ConfirmDialog`-Objekt und initialisiert dieses mit dem zugehörigen `GameScreen`-Objekt. Zudem ist die Angabe der Zeichenreihenfolge Pflicht.

#### 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.

### 3.1.27 Klasse Edge

#### Beschreibung:

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

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

#### 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.

### 3.1.28 Klasse FadeEffect

#### Beschreibung:

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

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

#### 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)**

!!!

### 3.1.29 Klasse FileUtility

#### Beschreibung:

Eine Hilfsklasse für Dateioperationen.

#### Eigenschaften:

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

**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 ungültiger Zeichen.

**public String** GetHash (**String** filename)

Liefert einen Hash-Wert zu der durch filename spezifizierten Datei.

### 3.1.30 Klasse GameModel

#### Beschreibung:

Repräsentiert ein 3D-Modell in einer Spielwelt.

#### Eigenschaften:

**public float** Alpha

Die Transparenz des Modells.

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 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.

**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.

### 3.1.31 Klasse GameModelInfo

#### Beschreibung:

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

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

#### 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 angegebenen Informationen zu Dateiname, Rotation und Skalierung.

### 3.1.32 Klasse GameObjectDistance

#### Beschreibung:

!!!

GameObjectDistance
+ Object : IGameObject + Distance : float

#### Eigenschaften:

**public IGameObject** Object

**public float** Distance

### 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.

#### Methoden:

**public Boolean** Equals (**C** other)

Vergleicht zwei Informationsobjekte für Spielobjekte.

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

### 3.1.34 Klasse GameScreen

#### Beschreibung:

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

#### Eigenschaften:

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 (IGameScreenComponent[] components) : void + RemoveGameComponents (IGameScreenComponent[] components) : void



**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)

Erzeugt ein neues GameScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

#### **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 zusätzlich 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** (**IGameScreenComponent**[] components)

**public** **void** **RemoveGameComponents** (**IGameScreenComponent**[] components)

### 3.1.35 Klasse GameScreenComponent

#### Beschreibung:

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

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

#### 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.

### 3.1.36 Klasse GraphicsSettingsScreen

#### Beschreibung:

Der Spielzustand, der die Grafikeinstellungen darstellt.

GraphicsSettingsScreen
# settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + GraphicsSettingsScreen (Knot3Game game) : void

#### Eigenschaften:

**protected void** settingsMenu

Das Menü, das die Einstellungen enthält.

### Konstruktoren:

**public** GraphicsSettingsScreen (**Knot3Game** game)

Erzeugt ein neues GraphicsSettingsScreen-Objekt und initialisiert dieses mit einem Knot3Game-Objekt.

### 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.

## 3.1.37 Klasse Input

### Beschreibung:

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

### Eigenschaften:

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

**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 zum aktuellen Frames.

**public KeyboardState** CurrentKeyboardState

Enthält den Tastaturzustand von XNA zum aktuellen Frames.

**public MouseState** PreviousMouseState

Enthält den Mauszustand von XNA zum vorherigen Frames.

**public** **KeyboardState** PreviousKeyboardState

Enthält den Tastaturzustand von XNA zum vorherigen Frames.

**public** **Boolean** GrabMouseMovement

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

#### Konstruktoeren:

**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.

### 3.1.38 Klasse InputItem

#### Beschreibung:

Ein Menüeintrag, der Texteingaben vom Spieler annimmt.

InputItem
+ InputText : String
+ InputItem (GameScreen screen, DisplayLayer drawOrder, String text) : void

#### Eigenschaften:

**public** **String** InputText

Beinhaltet den vom Spieler eingegebenen Text.

#### Konstruktoeren:

**public** **InputItem** (**GameScreen** screen, **DisplayLayer** drawOrder, **String** text)

Erzeugt ein neues InputItem-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge und für evtl. bereits vor-eingetragenen Text Pflicht.

### 3.1.39 Klasse KeyInputItem

### Beschreibung:

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

KeyInputItem
- option : OptionInfo
+ OnKeyEvent () : void + KeyInputItem (GameScreen screen, DisplayLayer drawOrder, OptionInfo option) : void

### Eigenschaften:

**private** OptionInfo option

Die Option in einer Einstellungsdatei.

### Konstruktoren:

**public** KeyInputItem (GameScreen screen, DisplayLayer drawOrder, OptionInfo option)

Erzeugt ein neues CheckBoxItem-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angaben zur Zeichenreihenfolge und der Eingabeoption Pflicht.

### Methoden:

**public void** OnKeyEvent ()

Speichert die aktuell gedrückte Taste in der Option.

## 3.1.40 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 Nutzer in diesem Fall zwingend einen nichtleeren Namen wählen.

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

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 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 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 Object Clone ()**

Erstellt eine vollständige Kopie des Knotens, inklusive der Kanten-Datenstruktur 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 ClearSelection ()**

Hebt die aktuelle Kantenauswahl auf.

**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 angegebenen Dateiformat.

**public Boolean Equals (T other)**

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

### 3.1.41 Klasse Knot3Game

#### Beschreibung:

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

#### Eigenschaften:

**public**  
**Boolean**  
**IsFullScreen**

**Boolean**  
**IsFullScreen**

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

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 denselben 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 Einstellungsdatei gespeicherte Auflösung oder falls nicht vorhanden auf die aktuelle Bildschirmauflösung und wechselt in den Vollbildmodus.

#### Methoden:

**public void Draw (GameTime time)**



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

```
public void Initialize ()
```

```
public void UnloadContent ()
```

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

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

### 3.1.42 Klasse KnotFileIO

#### Beschreibung:

Implementiert das Speicherformat für Knoten.

#### Eigenschaften:

```
public IEnumerable<string>  
FileExtensions
```

Die für eine Knoten-Datei gültigen Dateierendungen.

#### 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 einen Knoten aus einer angegebenen Datei.

KnotFileIO
+ FileExtensions : IEnumerable<string>
+ KnotFileIO () : void + Save (Knot knot) : void + Load (String filename) : Knot + LoadMetaData (String filename) : KnotMetaData

**public KnotMetaData LoadMetaData (String filename)**

Lädt die Metadaten eines Knotens aus einer angegebenen Datei.

### 3.1.43 Klasse KnotInputHandler

#### Beschreibung:

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

KnotInputHandler
- world : World - screen : GameScreen
+ Update (GameTime time) : void + KnotInputHandler (GameScreen screen, World world) : void

#### 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.

### 3.1.44 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

KnotMetaData
+ Name : String + Format : IKnotIO + CountEdges : Func<Integer> + Filename : String
+ KnotMetaData (String name, Func<Integer> countEdges, IKnotIO format, String filename) : KnotMetaData + KnotMetaData (String name, Func<Integer> countEdges) : KnotMetaData

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)`

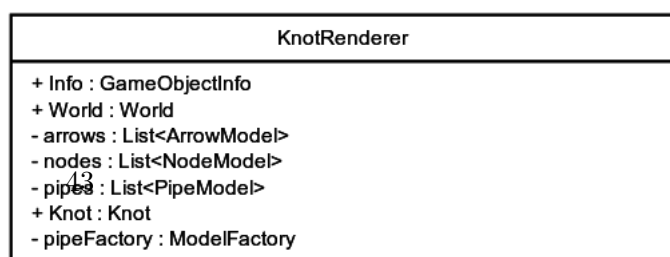
Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knotennamen und einer angegebenen Funktion, welche eine Kantenanzahl zurück gibt. Zusätzlich wird der Dateiname oder das Speicherformat angegeben, aus dem die Metadaten gelesen wurden.

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

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

### 3.1.45 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. Ist außerdem ein IGameObject und ein Container für die erstellten Spielobjekte.

**Eigenschaften:**

**public** **GameObjectInfo** **Game-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 verknüpft.

**public** **void** Update (**GameTime** gameTime)

Ruft die Update()-Methoden der Kanten, Übergänge und Pfeile auf.

**public** **void** Draw (**GameTime** gameTime)

Ruft die Draw()-Methoden der Kanten, Übergänge und Pfeile auf.

**public** **IEnumerator** GetEnumerator ()

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

### 3.1.46 Klasse KnotStringIO

#### Beschreibung:

Diese Klasse repräsentiert einen Parser für das Knoten-Austauschformat und enthält die eingelesenen Informationen

KnotStringIO
+ Name : String + Edges : IEnumerable<Edge> + CountEdges : Integer + Content : String
+ KnotStringIO (String content) : void + KnotStringIO (Knot knot) : void

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 eine 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.

### 3.1.47 Klasse Localizer

#### Beschreibung:

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

Localizer
- localization : ConfigFile
+ Localize (String text) : String

#### Eigenschaften:

### private ConfigFile localization

Die Datei, welche Informationen für die Lokalisierung enthält.

#### 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.

## 3.1.48 Klasse Menu

#### Beschreibung:

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

#### Eigenschaften:

Menu
+ RelativeItemSize : Func<int, Vector2> + RelativeItemPosition : Func<int, Vector2> + ItemForegroundColor : Func<ItemState, Color> + ItemBackgroundColor : Func<ItemState, Color> + ItemAlignX : HorizontalAlignment + ItemAlignY : VerticalAlignment
+ Add (MenuItem item) : void + Delete (MenuItem item) : void + GetItem (Integer i) : MenuItem + Size () : Integer + GetEnumerator () : IEnumerator + Menu (GameScreen screen, DisplayLayer drawOrder) : void

### 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.

#### Konstruktoren:

**public Menu (GameScreen screen, DisplayLayer drawOrder)**

Erzeugt ein neues Menu-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem ist die Angabe der Zeichenreihenfolge Pflicht.

#### 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.

### 3.1.49 Klasse MenuButton

#### Beschreibung:

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

MenuButton
+ OnClick : Action
+ MenuButton (GameScreen screen, DisplayLayer drawOrder, String name, Action onClick) : void

#### Eigenschaften:

**public Action OnClick**



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

#### Konstruktoren:

```
public MenuButton (GameScreen screen, DisplayLayer drawOrder, String name, Action onClick)
```

Erzeugt ein neues MenuButton-Objekt und initialisiert dieses mit dem zugehörigen GameScreen-Objekt. Zudem sind Angabe der Zeichenreihenfolge, einer Zeichenkette für den Namen der Schaltfläche und der Aktion, welche bei einem Klick ausgeführt wird Pflicht.

### 3.1.50 Klasse MenuItem

#### Beschreibung:

Ein abstrakte Klasse für Menüeinträge, die

#### Eigenschaften:

```
public ItemState Item-
```

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 OnLeftClick (Vector2 position, ClickState state, GameTime time)
```

Reaktionen auf einen Linksklick.

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

Reaktionen auf einen Rechtsklick.

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

**public void OnKeyEvent ()**

Reaktionen auf Tasteneingaben.

**public Rectangle Bounds ()**

Gibt die Ausmaße des Eintrags zurück.

### 3.1.51 Klasse MenuScreen

#### Beschreibung:

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

MenuScreen
+ 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)**

Wird aufgerufen, wenn in diesen Spielzustand gewechselt wird.

### 3.1.52 Klasse ModelFactory

#### Beschreibung:

Ein Zwischenspeicher für 3D-Modelle.

#### Eigenschaften:

ModelFactory
- cache : Dictionary<GameModelInfo, GameModel> - createModel : Func<GameScreen, GameModelInfo, GameModel>
+ this (GameScreen state, GameModelInfo info) : GameModel + ModelFactory (GameModelInfo, GameModel>, Func<GameScreen createModel) : void

**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 create-Model)
```

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.

## 3.1.53 Klasse ModelMouseHandler

### Beschreibung:

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

ModelMouseHandler
+ Update (GameTime time) : void

### Methoden:

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

Wird für jeden Frame aufgerufen.

## 3.1.54 Klasse MousePointer

### Beschreibung:

Repräsentiert einen Mauszeiger.

MousePointer
+ MousePointer (GameScreen screen) : void + Draw (GameTime time) : void

### Konstruktoren:

```
public MousePointer (GameScreen screen)
```

Erstellt einen neuen Mauszeiger für den angegebenen Spielzustand.

### Methoden:

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

Zeichnet den Mauszeiger.

### 3.1.55 Klasse Node

Beschreibung:

Eigenschaften:

public Integer X

public Integer Y

public Integer Z

private Integer scale

Konstruktoren:

public Node (Integer x, Integer y, Integer z)

Methoden:

public Vector3 ToVector ()

Node
+ X : Integer + Y : Integer + Z : Integer - scale : Integer
+ ToVector () : Vector3 + Node (Integer x, Integer y, Integer z) : void

### 3.1.56 Klasse NodeMap

Beschreibung:

Eine Zuordnung zwischen Kanten und Kantenübergänge.

NodeMap
+ Scale : Integer
+ From (Edge edge) : Node + To (Edge edge) : Node + OnEdgesChanged () : void

#### Eigenschaften:

**public Integer** Scale

#### 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:

NodeModel
+ Info : NodeModelInfo
+ NodeModel (GameScreen screen, NodeModelInfo info) : void + Draw (GameTime GameTime) : void + Update (GameTime GameTime) : void

**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:

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

**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.

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

#### 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 übergebenen Werten.

### 3.1.60 Klasse Options

#### Beschreibung:

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

Options
+ Default : ConfigFile

#### 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 Spielkomponenten angezeigt.

PauseDialog
- pauseMenu : VerticalMenu

### Eigenschaften:

**private** VerticalMenu pauseMenu

Das Menü, das verschiedene Schaltflächen enthält.

## 3.1.62 Klasse PipeModel

### Beschreibung:

Ein 3D-Modell, das eine Kante darstellt.

PipeModel
+ Info : PipeModelInfo
+ Intersects (Ray ray) : GameObjectDistance + PipeModel (GameScreen screen, PipeModelInfo info) : void

### 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.

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



### 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**  
**meObject-**  
**Info**  
**fo**

**Ga-**  
**In-**

Enthält Informationen über die Position des Knotens.

**public** **Knot** Knot

Der Knoten, dessen Kanten verschoben werden können.

**public** **World** World

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

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 Dateierweiterungen für das 3D-Drucker-Format.

PrinterIO
+ FileExtensions : IEnumerable<string>
+ PrinterIO () : void + Save (Knot knot) : void + Load (String filename) : Knot + LoadMetaData (String filename) : KnotMetaData

#### 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:

ProfileSettingsScreen
# settingsMenu : void
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + ProfileSettingsScreen (Knot3Game game) : void

**protected void** settingsMenu

!!!

#### Konstruktoren:

**public** ProfileSettingsScreen (**Knot3Game** game)

Erzeugt eine neue Instanz eines ProfileSettingsScreen-Objekts und initialisiert dieses mit einem Knot3Game-Objekt.

#### 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:

RenderEffect
<b>+ RenderTarget : RenderTarget2D</b> <b># screen : GameScreen</b> <b># spriteBatch : SpriteBatch</b>
<b>+ Begin (gameTime) : void</b> <b>+ End (gameTime) : void</b> <b>+ DrawModel (gameTime, GameModel GameModel) : void</b> <b>+ RemapModel (GameModel GameModel) : void</b> <b># DrawRenderTarget (gameTime time) : void</b>

**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 übergeordnete 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.

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

#### 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:

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

`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:

ShadowGameModel
+ ShadowColor : Color + ShadowAlpha : float
+ ShadowGameModel (GameScreen sreen, GameModel decoratedModel) : void + Draw (GameTime GameTime) : void

`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**

**GameObjectInfo**

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:

ShadowGameObject
+ Info : GameObjectInfo + World : World + ShadowPosition : Vector3 + OriginalPosition : Vector3
+ Center () : Vector3 + Update (GameTime gameTime) : void + Draw (GameTime gameTime) : void + Intersects (Ray Ray) : GameObjectDistance + ShadowGameObject (GameScreen screen, IGameObject decoratedObj) : void

**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.

SliderItem
+ Value : Integer + MinValue : Integer + MaxValue : Integer + Step : Integer
+ SliderItem (GameScreen screen, DisplayLayer drawOrder, Integer max, Integer min, Integer step, Integer value) : void

#### Eigenschaften:

**public Integer** Value

Der aktuelle Wert.

**public Integer** MinValue

Der minimale Wert.

**public Integer** MaxValue

Der maximale Wert.

**public Integer** Step

Schrittweite zwischen zwei einstellbaren Werten.

#### Konstruktoren:

**public** SliderItem (**GameScreen** screen, **DisplayLayer** drawOrder, **Integer** max,



Integer min, Integer step, Integer value)

### 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.

StandardEffect
# DrawRenderTarget (GameTime gameTime) : void + StandardEffect (GameScreen screen) : void

#### Konstruktoren:

**public** StandardEffect (GameScreen screen)

Erstellt einen neuen Standardeffekt.

#### Methoden:

**protected void** DrawRenderTarget (GameTime gameTime)

!!!

### 3.1.74 Klasse StartScreen

#### Beschreibung:

Der Startbildschirm.

#### Eigenschaften:

StartScreen
- buttons : Menu
+ Update (GameTime time) : void + Entered (GameScreen previousScreen, GameTime gameTime) : void + StartScreen (Knot3Game game) : void

**private** Menu buttons

Die Schaltflächen des Startbildschirms.

#### Konstruktoren:

**public** StartScreen (Knot3Game game)

Erzeugt eine neue Instanz eines StartScreen-Objekts und initialisiert diese mit einem Knot3Game-Objekt.

#### 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.

TextInputDialog
+ InputText : String
+ TextInputDialog (GameScreen screen, DisplayLayer drawOrder, String title, String text, String inputText) : void

#### Eigenschaften:

**public String** InputText

Der Text, der durch den Spieler eingegeben wurde.

#### Konstruktoren:

**public** TextInputDialog (**GameScreen** screen, **DisplayLayer** drawOrder, **String** title, **String** text, **String** inputText)

### 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.

TutorialChallengeModeScreen
+ Entered (GameScreen previousScreen, GameTime gameTime) : void

#### Methoden:

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

!!!

### 3.1.77 Klasse VerticalMenu

### Beschreibung:

Ein Menü, das alle Einträge vertikal anordnet.

VerticalMenu
+ AlignItems () : void + VerticalMenu (GameScreen screen, DisplayLayer drawOrder) : void

### Konstruktoren:

**public** VerticalMenu (GameScreen screen, DisplayLayer drawOrder)

Erzeugt eine neue Instanz eines VerticalMenu-Objekts und initialisiert diese mit dem zugehörigen GameScreen-Objekt. Zudem ist die Angaben der Zeichenreihenfolge Pflicht.

### 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

Widget
+ RelativeSize : Vector2 + RelativePosition : Vector2 + IsVisible : bool + BackgroundColor : Func<Color> + ForegroundColor : Func<Color> + AlignX : HorizontalAlignment + AlignY : VerticalAlignment
+ BoundingBox () : Rectangle + Widget (GameScreen screen, DisplayLayer drawOrder) : void

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.

WidgetKeyHandler
+ Update () : void

#### **Methoden:**

**public** **void** **Update** ()

!!!

### **3.1.80 Klasse WidgetMouseHandler**

#### **Beschreibung:**

Ein Inputhandler, der Mauseingaben auf Widgets verarbeitet.

WidgetMouseHandler
+ Update () : void

#### 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)
```

World
+ Camera : Camera + Objects : List<IGameObject> + SelectedObject : IGameObject + CurrentEffect : IRenderEffect
+ Update (GameTime gameTime) : void + Draw (GameTime gameTime) : void + World (GameScreen screen) : void + GetEnumerator () : IEnumerator

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.

IChallengeIO
+ Save (Challenge challenge) : void + Load (String filename) : Challenge + LoadMetaData (String filename) : ChallengeMetaData

#### 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 dieses Game befindet, sowie Informationen zu dem Game.

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

#### 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.

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

#### 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:

IJunction
+ EdgeFrom : Edge + EdgeTo : Edge

**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 implementiert wird, welche auf Tastatureingaben reagieren.

#### Eigenschaften:

IKeyEventListener
+ Index : DisplayLayer + IsKeyEventEnabled : Boolean + ValidKeys : List<Keys>
+ OnKeyEvent () : void

**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.

IKnotIO
+ FileExtensions : IEnumerable<string>
+ Save (Knot knot) : void + Load (String filename) : Knot + LoadMetaData (String filename) : KnotMetaData

#### Eigenschaften:

**public** **IEnumerable<string>** FileExtensions

Aufzählung der Dateierweiterungen.

#### 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 Mausklicks reagieren.

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

#### 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:

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

**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 Stapel gesichert und das Rendertarget des Effekts wird als aktuelles Rendertarget gesetzt.

**public void** End (**GameTime**)

Das auf dem Stapel gesicherte, vorher genutzte Rendertarget wird wiederhergestellt und das Rendertarget dieses Rendereffekts wird, unter Umständen in Unterklassen verändert, auf dieses übergeordnete 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

### 3.3.1 Enumeration ClickState

#### Beschreibung:

Eine Wertesammlung der möglichen Klickzustände einer Maustaste.

#### Eigenschaften:

**None** = 0

Kein gültiger oder ein undefinierter Zustand. Keine Reihenfolge.

**SingleClick** = 1

Ein Einzelklick.

**DoubleClick** = 2

Ein Doppelklick.

### 3.3.2 Enumeration Direction

#### Beschreibung:

Eine Wertesammlung der möglichen Richtungen in einem dreidimensionalen Raum. Wird benutzt, damit keine ungültigen Kantenrichtungen angegeben werden können.

#### Eigenschaften:

**Left** = 1

Links.

**Right** = 2

Rechts.

**Up** = 3

Hoch.

**Down** = 4

Runter.

**Forward** = 5

Vorwärts.

**Backward** = 6

Rückwärts.

**Zero** = 0

Keine Richtung.

### 3.3.3 Enumeration DisplayLayer

#### **Beschreibung:**

Die Zeichenreihenfolge der Elemente der grafischen Benutzeroberfläche.

#### **Eigenschaften:**

**None** = 0

Kein gültiger oder ein undefinierter Zustand. Keine Reihenfolge.

**Background** = 1

!!!

**GameWorld** = 2

**Dialog** = 3

**Menu** = 4

**MenuItem** = 5

**SubMenu** = 6

**SubMenuItem** = 7

**Overlay** = 8

**Cursor** = 9

### 3.3.4 Enumeration HorizontalAlignment

#### **Beschreibung:**

Eine horizontale Ausrichtung.

#### **Eigenschaften:**

**Left** = 0

Links.

**Center** = 1

Mittig.

**Right** = 2

Rechts.

### 3.3.5 Enumeration ItemState

#### Beschreibung:

Der Zustand eines Menüeintrags.

#### Eigenschaften:

**Selected** = 1

Ausgewählt.

**Hovered** = 2

Die Maus wurde direkt über den Menüeintrag navigiert und verweilt dort.

**None** = 0

Kein gültiger oder ein undefinierter Zustand. Keine Reihenfolge.

### 3.3.6 Enumeration VerticalAlignment

#### Beschreibung:

Die vertikale Ausrichtung.

#### Eigenschaften:

**Top** = 1

Oben.

**Center** = 0

Mittig.

**Bottom** = 2

Unten.



## **4 Abläufe**

### **4.1 Sequenzdiagramme**



## 5 Klassenindex

## **6 Anmerkungen**

## 7 Glossar

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