

ENTWURFSDOKUMENT

(V. 1.0)

KNOT³

PSE WS 2013/14

Auftraggeber:

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

Betreuer:

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

Auftragnehmer:

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

15. Dezember 2013

Inhaltsverzeichnis

1	Einleitung	4
2	Aufbau	5
2.1	Architektur	5
2.2	Klassendiagramm	6
2.3	Verwendete Entwurfsmuster	6
3	Klassenübersicht	7
3.1	Klassen	7
3.1.1	Klasse Angles3	7
3.1.2	Klasse ArrowModel	8
3.1.3	Klasse ArrowModelInfo	8
3.1.4	Klasse AudioSettingsScreen	9
3.1.5	Klasse BooleanOptionInfo	9
3.1.6	Klasse Camera	9
3.1.7	Klasse CelShadingEffect	11
3.1.8	Klasse Challenge	11
3.1.9	Klasse ChallengeFileIO	12
3.1.10	Klasse ChallengeLoadScreen	13
3.1.11	Klasse ChallengeMetaData	14
3.1.12	Klasse ChallengeMode	14
3.1.13	Klasse CheckBoxItem	15
3.1.14	Klasse Circle	15
3.1.15	Klasse ColorPicker	16
3.1.16	Klasse ColorPickItem	16
3.1.17	Klasse ConfigFile	17
3.1.18	Klasse ConfirmDialog	17
3.1.19	Klasse ControlSettingsScreen	17
3.1.20	Klasse CreativeLoadScreen	18
3.1.21	Klasse CreativeMode	18
3.1.22	Klasse CreditsScreen	19
3.1.23	Klasse Dialog	19
3.1.24	Klasse DistinctOptionInfo	20
3.1.25	Klasse DrawableGameScreenComponent	20
3.1.26	Klasse DropDownEntry	20
3.1.27	Klasse DropDownMenuItem	21
3.1.28	Klasse Edge	21
3.1.29	Klasse FadeEffect	22
3.1.30	Klasse FileIO	22

3.1.31	Klasse	GameModel	23
3.1.32	Klasse	GameModelInfo	24
3.1.33	Klasse	GameObjectInfo	24
3.1.34	Klasse	GameScreen	25
3.1.35	Klasse	GameScreenComponent	25
3.1.36	Klasse	GraphicsSettingsScreen	26
3.1.37	Klasse	Input	26
3.1.38	Klasse	InputItem	27
3.1.39	Klasse	KeyInputItem	28
3.1.40	Klasse	Knot	28
3.1.41	Klasse	Knot3Game	30
3.1.42	Klasse	KnotFileIO	30
3.1.43	Klasse	KnotMetaData	31
3.1.44	Klasse	KnotStringIO	31
3.1.45	Klasse	Localizer	32
3.1.46	Klasse	Menu	32
3.1.47	Klasse	MenuButton	33
3.1.48	Klasse	MenuItem	34
3.1.49	Klasse	MenuScreen	34
3.1.50	Klasse	ModelFactory	35
3.1.51	Klasse	ModelkeyHandler	35
3.1.52	Klasse	ModelMouseHandler	35
3.1.53	Klasse	ModelRenderer	36
3.1.54	Klasse	MousePointer	37
3.1.55	Klasse	NodeMap	37
3.1.56	Klasse	NodeModel	38
3.1.57	Klasse	NodeModelInfo	38
3.1.58	Klasse	OptionInfo	39
3.1.59	Klasse	Options	40
3.1.60	Klasse	PauseDialog	40
3.1.61	Klasse	PipeModel	40
3.1.62	Klasse	PipeModelInfo	40
3.1.63	Klasse	PipeMovement	41
3.1.64	Klasse	PrinterIO	42
3.1.65	Klasse	ProfileSettingsScreen	42
3.1.66	Klasse	RenderEffect	43
3.1.67	Klasse	RenderEffectStack	44
3.1.68	Klasse	SettingsScreen	44
3.1.69	Klasse	ShadowGameModel	45
3.1.70	Klasse	ShadowGameObject	45
3.1.71	Klasse	SliderItem	46
3.1.72	Klasse	StandardEffect	46
3.1.73	Klasse	StartScreen	47
3.1.74	Klasse	TextInputDialog	47
3.1.75	Klasse	TutorialChallengeMode	47
3.1.76	Klasse	VerticalMenu	47
3.1.77	Klasse	Widget	48
3.1.78	Klasse	WidgetKeyHandler	48
3.1.79	Klasse	WidgetMouseHandler	49
3.1.80	Klasse	World	49
3.1.81	Klasse	XNA.DrawableGameComponent	50

3.1.82	Klasse XNA.Game	50
3.1.83	Klasse XNA.GameComponent	50
3.2	Schnittstellen	51
3.2.1	Schnittstelle IChallengeIO	51
3.2.2	Schnittstelle ICloneable	51
3.2.3	Schnittstelle IEnumerable	52
3.2.4	Schnittstelle IEnumerable1	52
3.2.5	Schnittstelle IEquatable	52
3.2.6	Schnittstelle IEquatable1	52
3.2.7	Schnittstelle IGameObject	53
3.2.8	Schnittstelle IGameStateComponent	53
3.2.9	Schnittstelle IJunction	54
3.2.10	Schnittstelle IKeyEventListener	54
3.2.11	Schnittstelle IKnotIO	54
3.2.12	Schnittstelle IMouseEventListener	55
3.2.13	Schnittstelle IRenderEffect	55
3.3	Enumerationen	56
4	Abläufe	57
4.1	Sequenzdiagramme	57
5	Klassenindex	58
6	Anmerkungen	59
7	Glossar	60
7.1	Fachausdrücke	60
7.2	Abkürzungen	61

Kapitel 1

Einleitung

Das Knobel- und Konstruktionsspiel Knot³, welches im Auftrag des IBDS Dachsbacher ausgearbeitet und wie im Pflichtenheft spezifiziert angefertigt wird.

Kapitel 2

Aufbau

2.1 Architektur

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

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

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

Kapitel 3

Klassenübersicht

3.1 Klassen

3.1.1 Klasse Angles3

Beschreibung:

Eigenschaften:

`public float X`

`public float Y`

`public float Z`

`public Angles3 Zero`

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

Konstruktoren:

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

Methoden:

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

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

3.1.2 Klasse ArrowModel

Beschreibung:

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

Eigenschaften:

public ArrowModelInfo Info

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

Konstruktoren:

public ArrowModel (GameScreen screen, ArrowModelInfo info)

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

Methoden:

public void Draw (GameTime gameTime)

Zeichnet den Pfeil.

public GameObjectDistance Intersects (Ray ray)

public void Update (GameTime gameTime)

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

3.1.3 Klasse ArrowModelInfo

Beschreibung:

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

Eigenschaften:

public Vector3 Direction

Die Richtung, die der Pfeil zeigen soll.

Konstruktoren:

public ArrowModelInfo (Vector3 position, Vector3 direction)

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

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

3.1.4 Klasse AudioSettingsScreen

Beschreibung:

Eigenschaften:

`protected void settingsMenu`

Methoden:

`public void Update (GameTime time)`

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

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

3.1.5 Klasse BooleanOptionInfo

Beschreibung:

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

Eigenschaften:

`public bool Value`

Ein Property, das den aktuell abgespeicherten Wert zurück gibt.

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

Konstruktoren:

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

3.1.6 Klasse Camera

Beschreibung:

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

Eigenschaften:

public Vector3 Position

Die Position der Kamera.

public Vector3 Target

Das Ziel der Kamera.

public float FoV

Das Field of View.

public Matrix ViewMatrix

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

public Matrix WorldMatrix

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

public Matrix ProjectionMatrix

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

public Vector3 ArcballTarget

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

public BoundingBox ViewFrustum

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

private World World

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

public Angles3 Rotation

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

Konstruktoren:

public Camera (GameScreen screen, World world)

Methoden:

```
public Vector3 TargetDirection ()
```

```
public float TargetDistance ()
```

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

```
public Ray GetMouseRay (Vector2 mousePosition)
```

3.1.7 Klasse CelShadingEffect

Beschreibung:

Konstruktoren:

```
public CelShadingEffect (GameScreen screen)
```

Methoden:

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

```
public void DrawModel (GameTime, GameModel gameModel)
```

```
public void RemapModel (GameModel gameModel)
```

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

3.1.8 Klasse Challenge

Beschreibung:

Eigenschaften:

`public Knot Start`

`public Knot Target`

`private SortedList<Integer, String> highscore`

`public String Name`

`private IChallengeIO file`

`public IEnumerable<KeyValuePair<String, Integer>> Highscore`

`public ChallengeMetaData Info`

Konstruktoren:

`public Challenge (ChallengeInfo info)`

`public Challenge (IChallengeIO file)`

Methoden:

`public Boolean CreateChallenge (Knot start, Knot target, String name, IChallengeIO file)`

`public void AddToHighscore (String name, Integer time)`

Challenge
+ Start : Knot + Target : Knot - highscore : SortedList<Integer, String> + Name : String - file : IChallengeIO + Highscore : IEnumerable<KeyValuePair<String, Integer>> + Info : ChallengeMetaData
+ Challenge (ChallengeInfo info) : Challenge + Challenge (IChallengeIO file) : Challenge + CreateChallenge (Knot start, Knot target, String name, IChallengeIO file) : Boolean + AddToHighscore (String name, Integer time) : void

3.1.9 Klasse ChallengeFileIO

Beschreibung:

Eigenschaften:

```
public IEnumerable<KeyValuePair<String, Integer>>
```

```
public IEnumerable<KeyValuePair<String, Integer>> GetValues()
```

public String Name

public Knot StartKnot

```
public Knot TargetKnot
```

```
private KnotStringIO startParser
```

```
private KnotStringIO targetParser
```

```
public ChallengeMetaData Meta
```

Konstrukturen:

```
public ChallengeFileIO (String path)
```

Methoden:

```
public void Save (Challenge challenge)
```

3.1.10 Klasse ChallengeLoadScreen

Beschreibung:

```

ChallengeFileIO

+ Highscore : Enumerable<KeyValuePair<String, Integer>>
+ Name : String
+ StartKnot : Knot
+ TargetKnot : Knot
- startParser : KnotStringIO
- targetParser : KnotStringIO
+ Meta : ChallengeMetaData

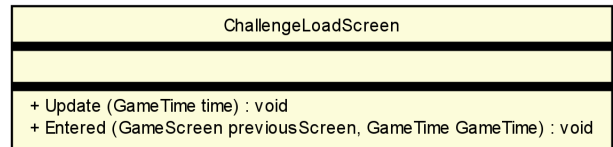
+ ChallengeFileIO (String path) : void
+ Save (Challenge challenge) : void

```

Methoden:

`public void Update (GameTime time)`

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



3.1.11 Klasse ChallengeMetaData

Beschreibung:

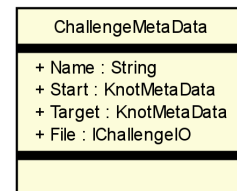
Eigenschaften:

`public String Name`

`public KnotMetaData Start`

`public KnotMetaData Target`

`public IChallengeIO File`



3.1.12 Klasse ChallengeMode

Beschreibung:

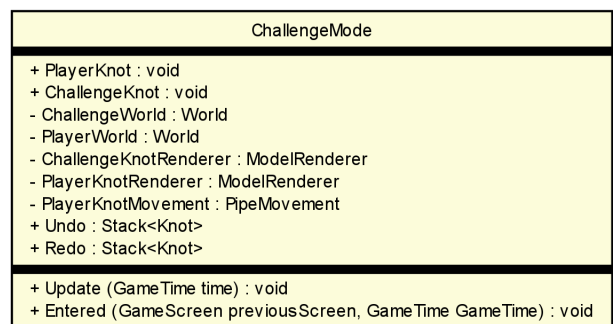
Eigenschaften:

`public void PlayerKnot`

`public void ChallengeKnot`

`private World ChallengeWorld`

`private World PlayerWorld`



```
private ModelRenderer ChallengeKnotRenderer
```

```
private ModelRenderer PlayerKnotRenderer
```

```
private PipeMovement PlayerKnotMovement
```

```
public Stack<Knot> Undo
```

```
public Stack<Knot> Redo
```

Methoden:

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

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

3.1.13 Klasse CheckBoxItem

Beschreibung:

Eigenschaften:

```
private BooleanOptionInfo option
```

Konstruktoren:

```
public CheckBoxItem (BooleanOptionInfo option)
```

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

3.1.14 Klasse Circle

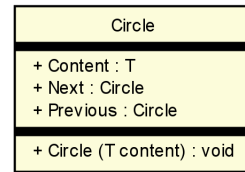
Beschreibung:

Eigenschaften:

`public T Content`

`public Circle Next`

`public Circle Previous`



Konstruktoren:

`public Circle (T content)`

3.1.15 Klasse ColorPicker

Beschreibung:

Eigenschaften:

`public Color Color`

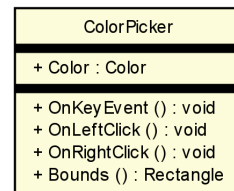
Methoden:

`public void OnKeyEvent ()`

`public void OnLeftClick ()`

`public void OnRightClick ()`

`public Rectangle Bounds ()`

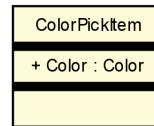


3.1.16 Klasse ColorPickItem

Beschreibung:

Eigenschaften:

`public Color Color`



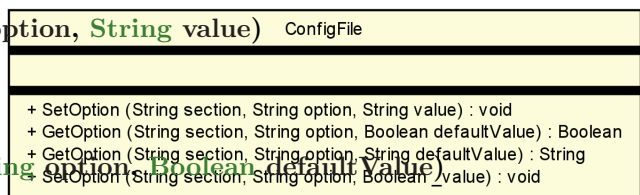
3.1.17 Klasse ConfigFile

Beschreibung:

Methoden:

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

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



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

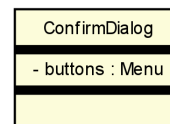
`public void SetOption (String section, String option, Boolean value)`

3.1.18 Klasse ConfirmDialog

Beschreibung:

Eigenschaften:

`private Menu buttons`



3.1.19 Klasse ControlSettingsScreen

Beschreibung:

Eigenschaften:

`protected void settingsMenu`

Methoden:

`public void Update (GameTime time)`

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

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

3.1.20 Klasse CreativeLoadScreen

Beschreibung:

Methoden:

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

`public void Update (GameTime time)`

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

3.1.21 Klasse CreativeMode

Beschreibung:

Eigenschaften:

`public void Knot`

`private World World`

`private ModelRenderer KnotRenderer`

`public Stack<Knot> Undo`

`public Stack<Knot> Redo`

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

Methoden:

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

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

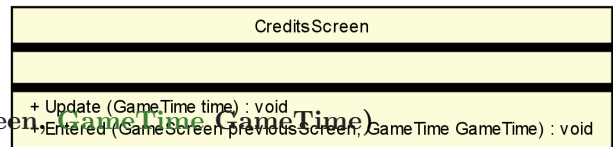
3.1.22 Klasse CreditsScreen

Beschreibung:

Methoden:

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

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



3.1.23 Klasse Dialog

Beschreibung:

Eigenschaften:

```
public String Name
```

```
public String Text
```

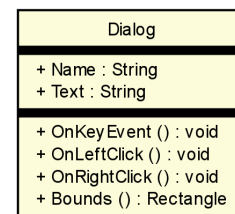
Methoden:

```
public void OnKeyEvent ()
```

```
public void OnLeftClick ()
```

```
public void OnRightClick ()
```

```
public Rectangle Bounds ()
```



3.1.24 Klasse DistinctOptionInfo

Beschreibung:

Eigenschaften:

`public HashSet<string> ValidValues`

`public String Value`

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

Konstruktoren:

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

3.1.25 Klasse DrawableGameScreenComponent

Beschreibung:

Eigenschaften:

`public GameScreen Screen`

`public DisplayLayer Index`

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

Methoden:

`public IEnumerable SubComponents (GameTime gameTime)`

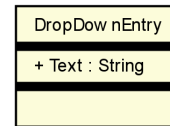
`public void DrawGameScreenStateComponent (GameScreen screen, DisplayLayer index)`

3.1.26 Klasse DropDownEntry

Beschreibung:

Eigenschaften:

`public String Text`



3.1.27 Klasse DropDownMenuItem

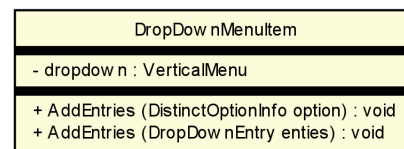
Beschreibung:

Eigenschaften:

`private VerticalMenu dropdown`

Methoden:

`public void AddEntries (DistinctOptionInfo option)`



`public void AddEntries (DropDownEntry enties)`

3.1.28 Klasse Edge

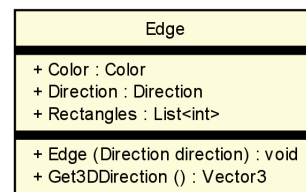
Beschreibung:

Eigenschaften:

`public Color Color`

`public Direction Direction`

`public List<int> Rectangles`



Konstruktoren:

`public Edge (Direction direction)`

Methoden:

`public Vector3 Get3DDirection ()`

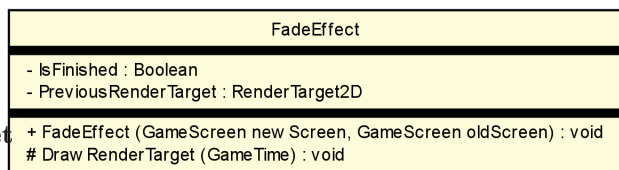
3.1.29 Klasse FadeEffect

Beschreibung:

Eigenschaften:

`private Boolean IsFinished`

`private RenderTarget2D PreviousRenderTarget`



Konstruktoren:

`public FadeEffect (GameScreen newScreen, GameScreen oldScreen)`

Methoden:

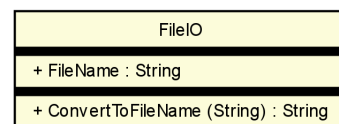
`protected void DrawRenderTarget (GameTime)`

3.1.30 Klasse FileIO

Beschreibung:

Eigenschaften:

`public String FileName`



Methoden:

`public String ConvertToFileName (String)`

3.1.31 Klasse GameModel

Beschreibung:

Eigenschaften:

`public float Alpha`

`public Color BaseColor`

`public Color HighlightColor`

`public float HighlightIntensity`

`public GameModelInfo Info`

`public XNA.Model Model`

`public World World`

`public Matrix WorldMatrix`

Konstruktoeren:

`public GameModel (GameScreen, GameModelInfo)`

Methoden:

`public Vector3 Center ()`

`public void Update (GameTime gameTime)`

`public void Draw (GameTime gameTime)`

`public GameObjectDistance Intersects (Ray ray)`

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, GameModelInfo) : void

3.1.32 Klasse GameModelInfo

Beschreibung:

Eigenschaften:

`public String Modelname`

`public Angles3 Rotation`

`public Vector3 Scale`

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

Konstruktoren:

`public GameModelInfo (String modelname, Angles3 rotation)`

3.1.33 Klasse GameObjectInfo

Beschreibung:

Eigenschaften:

`public Boolean IsMovable`

`public Boolean IsSelectable`

`public Boolean IsVisible`

`public Vector3 Position`

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

Methoden:

`public Boolean Equals (GameObjectInfo GameObjectInfo)`

`public Boolean Equals (T other)`

3.1.34 Klasse GameScreen

Beschreibung:

Eigenschaften:

`public Knot3Game Game`

`public Input Input`

`public RenderEffect PostProcessingEffect`

`public RenderEffectStack CurrentRenderEffects`

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

Konstruktoren:

`public GameScreen (Knot3Game game)`

Methoden:

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

`public void BeforeExit (GameScreen nextScreen, GameTime time)`

`public void Update (GameTime time)`

`public void AddGameComponents (IGameStateComponent[] components)`

`public void RemoveGameComponents (IGameStateComponent[] components)`

3.1.35 Klasse GameScreenComponent

Beschreibung:

Eigenschaften:

`public DisplayLayer Index`

`public GameScreen Screen`

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

Methoden:

`public IEnumerable SubComponents (GameTime gameTime)`

`public void GameStateComponent (GameScreen screen, DisplayLayer index)`

3.1.36 Klasse GraphicsSettingsScreen

Beschreibung:

Eigenschaften:

`protected void settingsMenu`

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

Methoden:

`public void Update (GameTime gameTime)`

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

3.1.37 Klasse Input

Beschreibung:

Eigenschaften:

`public ClickState RightMouseButton`

`public ClickState LeftMouseButton`

`public MouseState CurrentMouseState`

`public KeyboardState CurrentKeyboardState`

`public MouseState PreviousMouseState`

`public KeyboardState PreviousKeyboardState`

`public Boolean GrabMouseMovement`

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

Konstruktoren:

`public Input (GameScreen screen)`

Methoden:

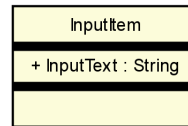
`public void Update (GameTime time)`

3.1.38 Klasse InputItem

Beschreibung:

Eigenschaften:

`public String InputText`

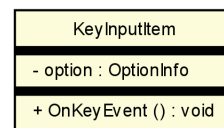


3.1.39 Klasse KeyInputItem

Beschreibung:

Eigenschaften:

`private OptionInfo option`



Methoden:

`public void OnKeyEvent ()`

3.1.40 Klasse Knot

Beschreibung:

Eigenschaften:

`public String Name`

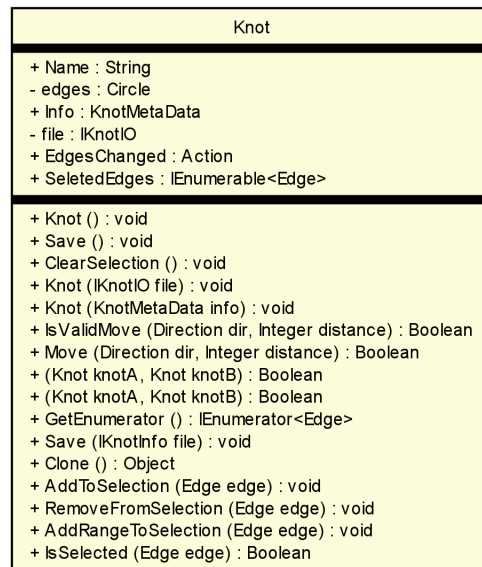
`private Circle edges`

`public KnotMetaData Info`

`private IKnotIO file`

`public Action EdgesChanged`

`public IEnumerable<Edge> SeletedEdges`



Konstruktoren:

```
public Knot ()
```

```
public Knot (IKnotIO file)
```

```
public Knot (KnotMetaData info)
```

Methoden:

```
public void Save ()
```

```
public void ClearSelection ()
```

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

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

```
public Boolean (Knot knotA, Knot knotB)
```

```
public Boolean (Knot knotA, Knot knotB)
```

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

```
public void Save (IKnotInfo file)
```

```
public Object Clone ()
```

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

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

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

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

3.1.41 Klasse Knot3Game

Beschreibung:

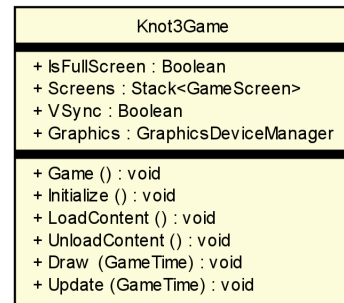
Eigenschaften:

`public Boolean IsFullScreen`

`public Stack<GameScreen> Screens`

`public Boolean VSync`

`public GraphicsDeviceManager Graphics`



Methoden:

`public void Game ()`

`public void Initialize ()`

`public void LoadContent ()`

`public void UnloadContent ()`

`public void Draw (GameTime)`

`public void Update (GameTime)`

3.1.42 Klasse KnotFileIO

Beschreibung:

Eigenschaften:

`public IEnumerable<Edge> Edges`

`public String Name`

`private KnotStringIO parser`

`public KnotMetaData Meta`

KnotFileIO
+ Edges : IEnumerable<Edge> + Name : String - parser : KnotStringIO + Meta : KnotMetaData
+ KnotFileIO (String path) : void + Save (Knot knot) : void

Konstruktoren:

`public KnotFileIO (String path)`

Methoden:

`public void Save (Knot knot)`

3.1.43 Klasse KnotMetaData

Beschreibung:

Eigenschaften:

`public String Name`

`public IKnotIO File`

`public Integer CountEdges`

KnotMetaData
+ Name : String + File : IKnotIO + CountEdges : Integer
KnotInfo (String name, Integer countEdges, IKnotIO file) : KnotMetaData

Methoden:

`protected KnotMetaData KnotInfo (String name, Integer countEdges, IKnotIO file)`

3.1.44 Klasse KnotStringIO

Beschreibung:

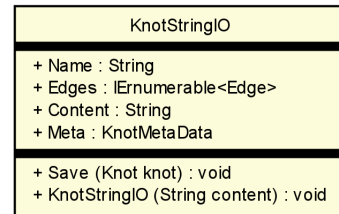
Eigenschaften:

`public String Name`

`public IEnumerable<Edge> Edges`

`public String Content`

`public KnotMetaData Meta`



Konstruktoren:

`public KnotStringIO (String content)`

Methoden:

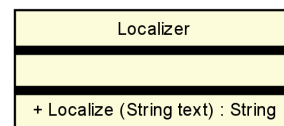
`public void Save (Knot knot)`

3.1.45 Klasse Localizer

Beschreibung:

Methoden:

`public String Localize (String text)`



3.1.46 Klasse Menu

Beschreibung:

Eigenschaften:

`public String Name`

`public Func<int, Vector2> RelativeItemSize`

`public Func<int, Vector2> RelativeItemPosition`

`public Func<ItemState, Vector2> ItemForegroundColor`

`public Func<ItemState, Vector2> ItemBackgroundColor`

`public HorizontalAlignment ItemAlignX`

`public VerticalAlignment ItemAlignY`

Methoden:

`public void Add (MenuItem item)`

`public void Delete (MenuItem item)`

`public MenuItem GetItem (Integer i)`

`public Integer Size ()`

`public IEnumerable GetEnumerator ()`

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

3.1.47 Klasse MenuButton

Beschreibung:

Eigenschaften:

`public String Name`

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

Konstruktoren:

`public MenuButton (String name)`

3.1.48 Klasse MenuItem

Beschreibung:

Eigenschaften:

`public ItemState ItemState`

`public Integer ItemOrder`

`public String Text`

MenuItem
+ ItemState : ItemState + ItemOrder : Integer + Text : String
+ OnLeftClick () : void + OnRightClick () : void + OnKeyEvent () : void + Bounds () : Rectangle

Methoden:

`public void OnLeftClick ()`

`public void OnRightClick ()`

`public void OnKeyEvent ()`

`public Rectangle Bounds ()`

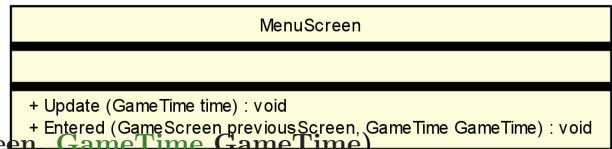
3.1.49 Klasse MenuScreen

Beschreibung:

Methoden:

`public void Update (GameTime time)`

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



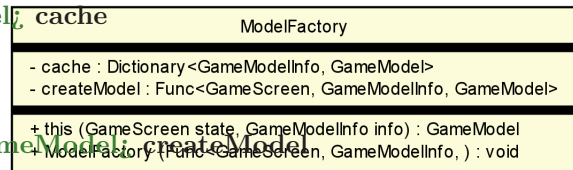
3.1.50 Klasse ModelFactory

Beschreibung:

Eigenschaften:

`private Dictionary<GameModelInfo, GameModel> cache`

`private Func<GameScreen, GameModelInfo, GameModel> createModel`



Konstruktoren:

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

Methoden:

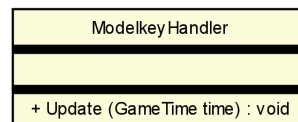
`public GameModel this (GameScreen state, GameModelInfo info)`

3.1.51 Klasse ModelkeyHandler

Beschreibung:

Methoden:

`public void Update (GameTime time)`

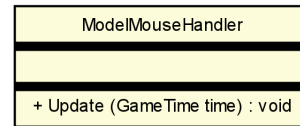


3.1.52 Klasse ModelMouseHandler

Beschreibung:

Methoden:

`public void Update (GameTime time)`



3.1.53 Klasse ModelRenderer

Beschreibung:

Eigenschaften:

`public GameObjectInfo Info`

`public World World`

`private List<ArrowModel> arrows`

`private List<NodeModel> nodes`

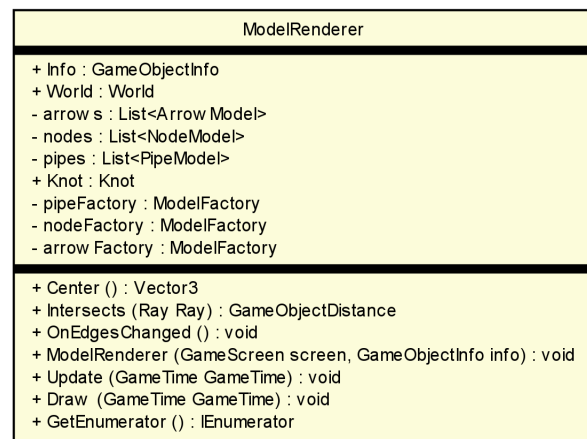
`private List<PipeModel> pipes`

`public Knot Knot`

`private ModelFactory pipeFactory`

`private ModelFactory nodeFactory`

`private ModelFactory arrowFactory`



Konstruktoren:

`public ModelRenderer (GameScreen screen, GameObjectInfo info)`

Methoden:

```
public Vector3 Center ()
```

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

```
public void OnEdgesChanged ()
```

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

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

```
public IEnumerator GetEnumerator ()
```

3.1.54 Klasse MousePointer

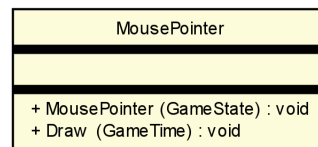
Beschreibung:

Konstruktoeren:

```
public MousePointer (GameState)
```

Methoden:

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



3.1.55 Klasse NodeMap

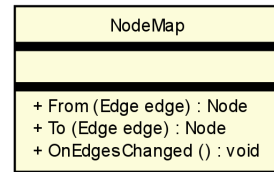
Beschreibung:

Methoden:

public **Node** From (**Edge** edge)

public **Node** To (**Edge** edge)

public **void** OnEdgesChanged ()



3.1.56 Klasse NodeModel

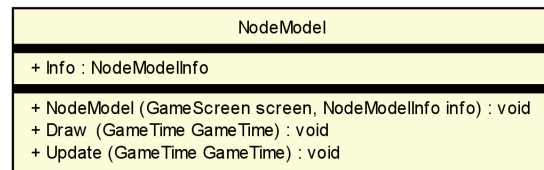
Beschreibung:

Eigenschaften:

public **NodeModelInfo** Info

Konstruktoren:

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



Methoden:

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

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

3.1.57 Klasse NodeModelInfo

Beschreibung:

Eigenschaften:

`public Edge EdgeFrom`

`public Edge EdgeTo`

`public Knot Knot`

`public Vector3 Position`

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

Konstruktoren:

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

3.1.58 Klasse OptionInfo

Beschreibung:

Eigenschaften:

`private ConfigFile configFile`

`public String Section`

`public String Name`

`public String DefaultValue`

`public String Value`

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

Konstruktoren:

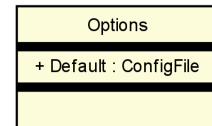
`public OptionInfo (String section, String name, String defaultValue, ConfigFile configFile)`

3.1.59 Klasse Options

Beschreibung:

Eigenschaften:

`public ConfigFile Default`

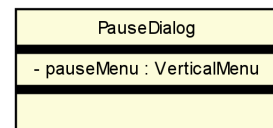


3.1.60 Klasse PauseDialog

Beschreibung:

Eigenschaften:

`private VerticalMenu pauseMenu`



3.1.61 Klasse PipeModel

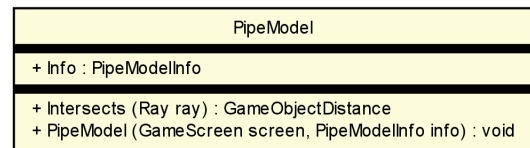
Beschreibung:

Eigenschaften:

`public PipeModelInfo Info`

Konstruktoren:

`public PipeModel (GameScreen screen, PipeModelInfo info)`



Methoden:

`public GameObjectDistance Intersects (Ray ray)`

3.1.62 Klasse PipeModelInfo

Beschreibung:

Eigenschaften:

public Edge Edge

public Knot Knot

public Vector3 PositionFrom

public Vector3 PositionTo

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

Konstruktoren:

public PipeModelInfo (Knot knot, Edge edge)

3.1.63 Klasse PipeMovement

Beschreibung:

Eigenschaften:

public GameObjectInfo Info

public Knot Knot

public World World

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

Konstruktoren:

public PipeMovement (GameState, World, GameObjectInfo)

Methoden:

```
public Vector3 Center ()
```

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

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

```
public IEnumerator GetEnumerator ()
```

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

3.1.64 Klasse PrinterIO

Beschreibung:

Eigenschaften:

```
public IEnumerable<Edge> Edges
```

```
public String Name
```

```
public KnotMetaData Meta
```

Konstruktoren:

```
public PrinterIO (string path)
```

Methoden:

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

3.1.65 Klasse ProfileSettingsScreen

Beschreibung:

PrinterIO
+ Edges : IEnumerable<Edge> + Name : String + Meta : KnotMetaData
+ Save (Knot knot) : void + PrinterIO (string path) : void

Eigenschaften:

`protected void settingsMenu`

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

Methoden:

`public void Update (GameTime time)`

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

3.1.66 Klasse RenderEffect

Beschreibung:

Eigenschaften:

`public RenderTarget2D RenderTarget`

`protected GameScreen screen`

`protected SpriteBatch spriteBatch`

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

Methoden:

`public void Begin (GameTime)`

`public void End (GameTime)`

`public void DrawModel (GameTime, GameModel GameModel)`

`public void RemapModel (GameModel GameModel)`

`protected void DrawRenderTarget (GameTime)`

3.1.67 Klasse RenderEffectStack

Beschreibung:

Eigenschaften:

public **IRenderEffect** CurrentEffect

private **IRenderEffect** DefaultEffect

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

Konstruktoeren:

public RenderEffectStack (**IRenderEffect** defaultEffect)

Methoden:

public **IRenderEffect** Pop ()

public void Push (**IRenderEffect** effect)

3.1.68 Klasse SettingsScreen

Beschreibung:

Eigenschaften:

protected void navigation

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

Methoden:

public void Update (**GameTime** time)

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

3.1.69 Klasse ShadowGameModel

Beschreibung:

Eigenschaften:

`public Color ShadowColor`

`public float ShadowAlpha`

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

Konstruktoren:

`public ShadowGameModel (GameScreen screen, GameModel decoratedModel)`

Methoden:

`public void Draw (GameTime GameTime)`

3.1.70 Klasse ShadowGameObject

Beschreibung:

Eigenschaften:

`public GameObjectInfo Info`

`public World World`

`public Vector3 ShadowPosition`

`public Vector3 OriginalPosition`

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

Konstruktoren:

`public ShadowGameObject (GameScreen screen, IGameObject decoratedObj)`

Methoden:

```
public Vector3 Center ()
```

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

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

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

3.1.71 Klasse SliderItem

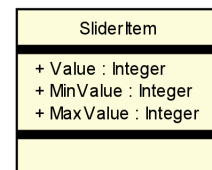
Beschreibung:

Eigenschaften:

```
public Integer Value
```

```
public Integer MinValue
```

```
public Integer MaxValue
```



3.1.72 Klasse StandardEffect

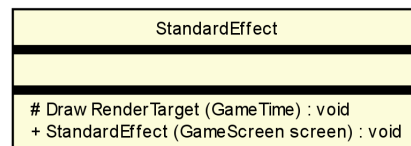
Beschreibung:

Konstruktoren:

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

Methoden:

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



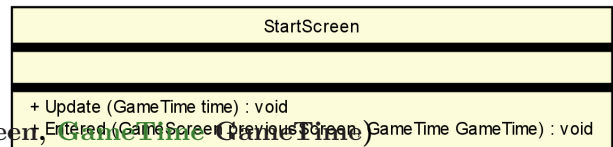
3.1.73 Klasse StartScreen

Beschreibung:

Methoden:

`public void Update (GameTime time)`

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

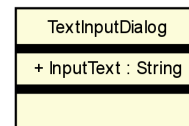


3.1.74 Klasse TextInputDialog

Beschreibung:

Eigenschaften:

`public String InputText`

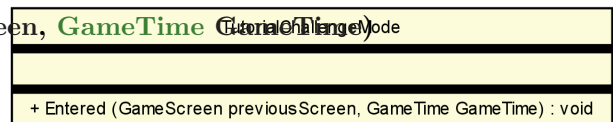


3.1.75 Klasse TutorialChallengeMode

Beschreibung:

Methoden:

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

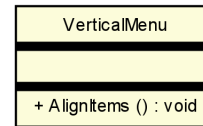


3.1.76 Klasse VerticalMenu

Beschreibung:

Methoden:

`public void AlignItems ()`



3.1.77 Klasse Widget

Beschreibung:

Eigenschaften:

`public Vector2 RelativeSize`

`public Vector2 RelativePosition`

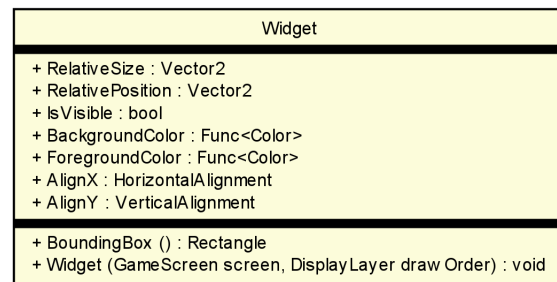
`public bool IsVisible`

`public Func<Color> BackgroundColor`

`public Func<Color> ForegroundColor`

`public HorizontalAlignment AlignX`

`public VerticalAlignment AlignY`



Konstruktoren:

`public Widget (GameScreen screen, DisplayLayer drawOrder)`

Methoden:

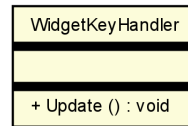
`public Rectangle BoundingBox ()`

3.1.78 Klasse WidgetKeyHandler

Beschreibung:

Methoden:

```
public void Update ()
```

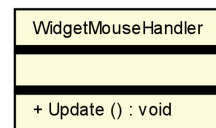


3.1.79 Klasse WidgetMouseHandler

Beschreibung:

Methoden:

```
public void Update ()
```



3.1.80 Klasse World

Beschreibung:

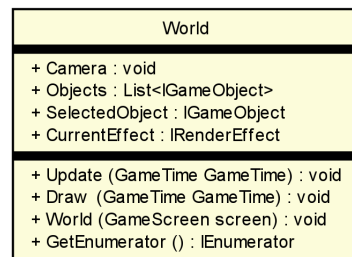
Eigenschaften:

```
public void Camera
```

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

```
public IGameObject SelectedObject
```

```
public IRenderEffect CurrentEffect
```



Konstruktoren:

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

Methoden:

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

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

```
public IEnumerator GetEnumerator ()
```

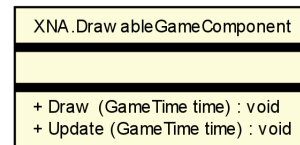
3.1.81 Klasse XNA.DrawableGameComponent

Beschreibung:

Methoden:

`public void Draw (GameTime time)`

`public void Update (GameTime time)`



3.1.82 Klasse XNA.Game

Beschreibung:

Methoden:

`public void Game ()`

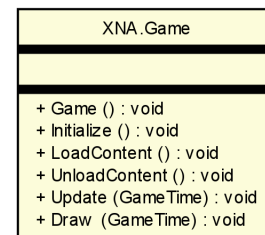
`public void Initialize ()`

`public void LoadContent ()`

`public void UnloadContent ()`

`public void Update (GameTime)`

`public void Draw (GameTime)`

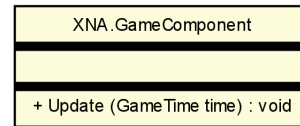


3.1.83 Klasse XNA.GameComponent

Beschreibung:

Methoden:

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



3.2 Schnittstellen

3.2.1 Schnittstelle IChallengeIO

Beschreibung:

Eigenschaften:

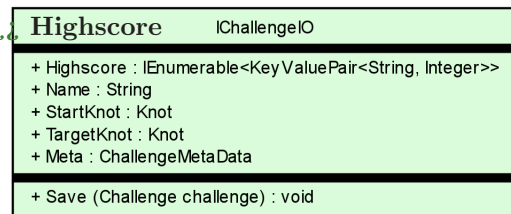
```
public IEnumerable<KeyValuePair<String, Integer>>
```

```
public String Name
```

```
public Knot StartKnot
```

```
public Knot TargetKnot
```

```
public ChallengeMetaData Meta
```



Methoden:

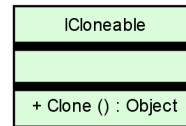
```
public void Save (Challenge challenge)
```

3.2.2 Schnittstelle ICloneable

Beschreibung:

Methoden:

`public Object Clone ()`

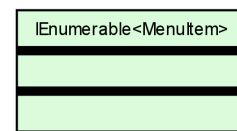


3.2.3 Schnittstelle IEnumerable

Beschreibung:

Methoden:

`public IEnumerator GetEnumerator ()`

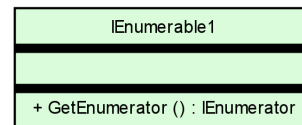


3.2.4 Schnittstelle IEnumerable1

Beschreibung:

Methoden:

`public IEnumerator GetEnumerator ()`

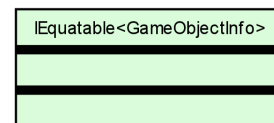


3.2.5 Schnittstelle IEquatable

Beschreibung:

Methoden:

`public Boolean Equals (T other)`

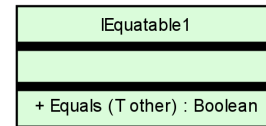


3.2.6 Schnittstelle IEquatable1

Beschreibung:

Methoden:

`public Boolean Equals (T other)`



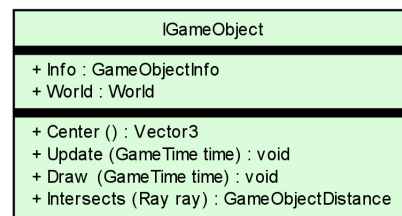
3.2.7 Schnittstelle IGameObject

Beschreibung:

Eigenschaften:

`public GameObjectInfo Info`

`public World World`



Methoden:

`public Vector3 Center ()`

`public void Update (GameTime time)`

`public void Draw (GameTime time)`

`public GameObjectDistance Intersects (Ray ray)`

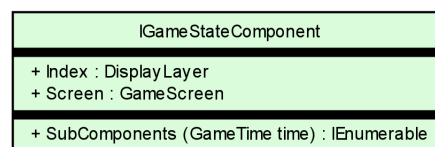
3.2.8 Schnittstelle IGameStateComponent

Beschreibung:

Eigenschaften:

`public DisplayLayer Index`

`public GameScreen Screen`



Methoden:

`public IEnumerable SubComponents (GameTime time)`

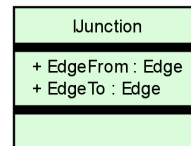
3.2.9 Schnittstelle IJunction

Beschreibung:

Eigenschaften:

`public Edge EdgeFrom`

`public Edge EdgeTo`



3.2.10 Schnittstelle IKeyEventListe- ner

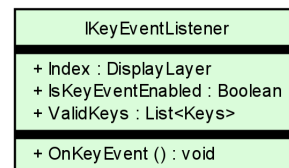
Beschreibung:

Eigenschaften:

`public DisplayLayer Index`

`public Boolean IsKeyEventEnabled`

`public List<Keys> ValidKeys`



Methoden:

`public void OnKeyEvent ()`

3.2.11 Schnittstelle IKnotIO

Beschreibung:

Eigenschaften:

`public IEnumerable<Edge> Edges`

`public String Name`

`public KnotMetaData Meta`

Methoden:

`public void Save (Knot knot)`

IKnotIO
+ Edges : IEnumerable<Edge> + Name : String + Meta : KnotMetaData
+ Save (Knot knot) : void

3.2.12 Schnittstelle IMouseEventListener

Beschreibung:

Eigenschaften:

`public DisplayLayer Index`

`public Boolean IsMouseEventEnabled`

Methoden:

`public void OnLeftClick ()`

`public void OnRightClick ()`

`public Rectangle Bounds ()`

IMouseEventListener
+ Index : DisplayLayer + IsMouseEventEnabled : Boolean
+ OnLeftClick () : void + OnRightClick () : void + Bounds () : Rectangle

3.2.13 Schnittstelle IRenderEffect

Beschreibung:

Eigenschaften:

`public RenderTarget2D RenderTarget`

Methoden:

`public void Begin (GameTime)`

`public void End (GameTime)`

`public void DrawModel (GameTime, GameModel model)`

`public void RemapModel (GameModel model)`

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

3.3 Enumerationen

Kapitel 4

Abläufe

4.1 Sequenzdiagramme

Kapitel 5

Klassenindex

Kapitel 6

Anmerkungen

Kapitel 7

Glossar

Test Test

7.1 Fachausdrücke

Test (Test-Beschreibung) ... 50

7.2 Abkürzungen

Test Test 50