Summary

Generated on:	18.02.2014 - 14:16:27
Parser:	OpenCoverParser
Assemblies:	1
Classes:	41
Files:	40
Coverage:	55.9%
Covered lines:	1418
Uncovered lines:	1115
Coverable lines:	2533
Total lines:	8365

Assemblies

Knot3	**55.9%**
Knot3.Audio.AudioManager	45.2%
Knot3.Audio.LoopPlaylist	0%
Knot3.Audio.OggVorbisFile	23.5%
Knot3.Audio.SoundEffectFile	0%
Knot3.Core.Angles3	100%
Knot3.Core.BooleanOptionInfo	0%
Knot3.Core.Camera	65.3%
Knot3.Core.ConfigFile	100%
Knot3.Core.DisplayLayer	98.1%
Knot3.Core.DistinctOptionInfo	0%
Knot3.Core.FloatOptionInfo	0%
Knot3.Core.KeyOptionInfo	0%
Knot3.Core.Localizer	0%
Knot3.Core.OptionInfo	0%
Knot3.Core.Options	81.2%
Knot3.Core.World	15.9%
Knot3.Data.Challenge	0%
Knot3.Data.ChallengeFileIO	0%
Knot3.Data.ChallengeMetaData	0%
Knot3.Data.CircleEntry‘1	92.3%
Knot3.Data.CircleExtensions	100%
Knot3.Data.Direction	98.6%
Knot3.Data.Edge	95%
Knot3.Data.Knot	84.8%
Knot3.Data.KnotFileIO	22.2%
Knot3.Data.KnotMetaData	64%
Knot3.Data.KnotStringIO	41.1%
Knot3.Data.Node	73.7%
Knot3.Data.NodeMap	88.5%
Knot3.Data.PrinterIO	0%
Knot3.Data.RectangleMap	0%
Knot3.Data.ZipHelper	0%
Knot3.Platform.SystemInfo	88.6%
Knot3.Utilities.BoundingCylinder	0%
Knot3.Utilities.FileIndex	0%
Knot3.Utilities.FileUtility	36.1%
Knot3.Utilities.IniFile	81.1%
Knot3.Utilities.RayExtensions	0%
Knot3.Utilities.SavegameLoader‘2	0%
Knot3.Widgets.Bounds	83.5%
Knot3.Widgets.ScreenPoint	50.8%

Knot3.Audio.AudioManager

Summary

Class:	Knot3.Audio.AudioManager
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\AudioManager.cs
Coverage:	45.2%
Covered lines:	48
Uncovered lines:	58
Coverable lines:	106
Total lines:	247

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	4	100	71.43
Reset()	1	100	100
AddXnaAudioFile(...)	3	0	0
LoadXnaSoundEffect(.	1	0	0
AddOggAudioFile(...)	3	100	80
LoadOggAudioFile(...	1	90.91	100
StartBackgroundMusic	2	0	0
PlaySound(...)	2	0	0
UnloadContent()	1	0	0
Volume(...)	1	0	0
SetVolume(...)	1	100	100
ValidVolume(...)	1	100	100
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\AudioManager.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`

```
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.IO;
35  using System.Linq;
36
37  using Microsoft.Xna.Framework;
38  using Microsoft.Xna.Framework.Audio;
39  using Microsoft.Xna.Framework.Content;
40  using Microsoft.Xna.Framework.GamerServices;
41  using Microsoft.Xna.Framework.Graphics;
42  using Microsoft.Xna.Framework.Input;
43  using Microsoft.Xna.Framework.Media;
44  using Microsoft.Xna.Framework.Net;
45  using Microsoft.Xna.Framework.Storage;
46
47  using Knot3.Core;
48  using Knot3.Data;
49  using Knot3.Development;
50  using Knot3.GameObjects;
51  using Knot3.Input;
52  using Knot3.RenderEffects;
53  using Knot3.Screens;
54  using Knot3.Utilities;
55  using Knot3.Widgets;
56
57  #endregion
58
59  namespace Knot3.Audio
60  {
61    public class AudioManager : DrawableGameScreenComponent
62    {
63      /// <summary>
64      /// Eine Zuordnung zwischen dem Typ der Audiodateien und den Ordnern unter
65      /// in denen sich die Audiodateien befinden.
66      /// </summary>
1     67      private static readonly Dictionary<Sound, string> AudioDirectories
68      = new Dictionary<Sound, string> {
69        { Sound.CreativeMusic,      "Music/Creative" },
70        { Sound.ChallengeMusic,      "Music/Challenge" },
71        { Sound.MenuMusic,        "Music/Menu" },
72        { Sound.PipeMoveSound,      "Sound/Pipe/Move" },
73        { Sound.PipeInvalidMoveSound,  "Sound/Pipe/Invalid-Move" },
74      };
75
76      // Enthlt alle gefunden Audiodateien, sortiert nach ihrem Zweck
1     77      private static Dictionary<Sound, HashSet<IAudioFile>> AudioFiles
78          = new Dictionary<Sound, HashSet<IAudioFile>> ();
79
80      /// <summary>
81      /// Die aktuell verwendete Hintergrundmusik.
82      /// </summary>
83      public Sound BackgroundMusic
84      {
0     85        get {
```

```
  0    86          return _backgroundMusic;
  0    87        }
  0    88        set {
  0    89          if (value != Sound.None && value != _backgroundMusic) {
  0    90            _backgroundMusic = value;
  0    91            StartBackgroundMusic ();
  0    92          }
  0    93        }
       94      }
       95
  1    96      private static Sound _backgroundMusic = Sound.None;
       97
       98      /// <summary>
       99      /// Enthlt die Playlist, die aktuell abgespielt wird,
      100      /// oder null, falls keine Playlist abgespielt wird.
      101      /// </summary>
  0   102      public static IPlaylist Playlist { get; set; }
      103
  1   104      private static Dictionary<Sound, float> VolumeMap = new Dictionary<Sound,
      105
      106      /// <summary>
      107      /// Erstellt einen neuen AudioManager fr den angegebenen Spielzustand.
      108      /// </summary>
  3   109      public AudioManager (IGameScreen screen, string directory = ".")
      110      : base (screen, DisplayLayer.None)
  3   111      {
  6   112        if (AudioFiles.Count == 0) {
      113          // Erstelle fr alle Enum-Werte von Sound ein HashSet
 63   114          foreach (Sound soundType in typeof (Sound).ToEnumValues<Sound>()) {
 18   115            AudioFiles [soundType] = new HashSet<IAudioFile> ();
 18   116            VolumeMap [soundType] = ValidVolume (Options.Default ["volume", soun
 18   117          }
      118
      119          // Suche nach XNA-Audio-Dateien
  3   120          FileUtility.SearchFiles (directory, new string[] {".xnb"}, AddXnaAudio
      121
      122          // Suche nach OGG-Dateien
  3   123          FileUtility.SearchFiles (directory, new string[] {".ogg"}, AddOggAudio
  3   124        }
  3   125      }
      126
      127      public static void Reset ()
  3   128      {
  3   129        AudioFiles.Clear ();
  3   130        VolumeMap.Clear ();
  3   131      }
      132
      133      private void AddXnaAudioFile (string filepath)
  0   134      {
  0   135        filepath = filepath.Replace (".xnb", String.Empty).Replace (@"Content\",
      136
  0   137        foreach (KeyValuePair<Sound,string> pair in AudioDirectories) {
  0   138          Sound soundType = pair.Key;
  0   139          string directory = pair.Value;
  0   140          if (filepath.ToLower ().Contains (directory.ToLower ())) {
  0   141            string name = Path.GetFileName (filepath);
  0   142            LoadXnaSoundEffect (filepath, name, soundType);
  0   143            break;
      144          }
  0   145        }
  0   146      }
```

```
      147
      148       private void LoadXnaSoundEffect (string filepath, string name, Sound sound
    0 149       {
    0 150         try {
      151           // versuche, die Audiodatei als "SoundEffect" zu laden
    0 152           SoundEffect soundEffect = Screen.Content.Load<SoundEffect> (filepath);
    0 153           AudioFiles [soundType].Add (new SoundEffectFile (name, soundEffect, so
    0 154           Log.Debug ("Load sound effect (", soundType, "): ", filepath);
    0 155         }
    0 156         catch (Exception ex) {
    0 157           Log.Debug (ex);
    0 158         }
    0 159       }
      160
      161       private void AddOggAudioFile (string filepath)
    6 162       {
    6 163         filepath = filepath.Replace (@"\", "/");
      164
   57 165         foreach (KeyValuePair<Sound,string> pair in AudioDirectories) {
   15 166           Sound soundType = pair.Key;
   15 167           string directory = pair.Value;
   21 168           if (filepath.ToLower ().Contains (directory.ToLower ())) {
    6 169             string name = Path.GetFileName (filepath);
    6 170             LoadOggAudioFile (filepath, name, soundType);
    6 171             break;
      172           }
    9 173         }
    6 174       }
      175
      176       private void LoadOggAudioFile (string filepath, string name, Sound soundTy
    6 177       {
    6 178         try {
      179           // erstelle ein AudioFile-Objekt
    6 180           Log.Debug ("Load ogg audio file (", soundType, "): ", filepath);
    6 181           AudioFiles [soundType].Add (new OggVorbisFile (name, filepath, soundTy
    0 182         }
   12 183         catch (Exception ex) {
      184           // egal, warum das laden nicht klappt; mehr als die Fehlermeldung anze
      185           // macht wegen einer fehlenden Musikdatei keinen Sinn
    6 186           Log.Debug ("Failed to load ffmpeg audio file (", soundType, "): ", fil
    6 187           Log.Debug (ex);
    6 188         }
    6 189       }
      190
      191       private void StartBackgroundMusic ()
    0 192       {
    0 193         if (Playlist != null) {
    0 194           Playlist.Stop ();
    0 195         }
    0 196         Log.Debug ("Background Music: ", BackgroundMusic);
    0 197         Playlist = new LoopPlaylist (AudioFiles [BackgroundMusic]);
    0 198         Playlist.Shuffle ();
    0 199         Playlist.Start ();
    0 200       }
      201
      202       public void PlaySound (Sound sound)
    0 203       {
    0 204         Log.Debug ("Sound: ", sound);
    0 205         if (AudioFiles [sound].Count > 0) {
    0 206           AudioFiles [sound].RandomElement ().Play ();
    0 207         }
```

```
  0   208        else {
  0   209          Log.Debug ("There are no audio files for: ", sound);
  0   210        }
  0   211      }
      212
      213      [ExcludeFromCodeCoverageAttribute]
      214      public override void Update (GameTime time)
      215      {
      216        if (Playlist != null) {
      217          Playlist.Update (time);
      218        }
      219        base.Update (time);
      220      }
      221
      222      protected override void UnloadContent ()
  0   223      {
  0   224        Log.Debug ("UnloadContent ()");
  0   225        Playlist.Stop ();
  0   226        base.UnloadContent ();
  0   227      }
      228
      229      public static float Volume (Sound soundType)
  0   230      {
  0   231        return VolumeMap [soundType];
  0   232      }
      233
      234      public static void SetVolume (Sound soundType, float volume)
  1   235      {
  1   236        volume = ValidVolume (volume);
  1   237        VolumeMap [soundType] = volume;
  1   238        Options.Default ["volume", soundType.ToString (), 1] = volume;
  1   239        Log.Debug ("Set Volume (", soundType, "): ", volume);
  1   240      }
      241
      242      public static float ValidVolume (float volume)
 19   243      {
 19   244        return MathHelper.Clamp (volume, 0.0f, 2.0f);
 19   245      }
      246    }
      247  }
```

Knot3.Audio.LoopPlaylist

Summary

Class:	Knot3.Audio.LoopPlaylist
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\LoopPlaylist.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	26
Coverable lines:	26
Total lines:	132

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	0	0
Shuffle()	1	0	0
Start()	2	0	0
Stop()	2	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\LoopPlaylist.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;

```
      34  using System.IO;
      35  using System.Linq;
      36
      37  using Microsoft.Xna.Framework;
      38  using Microsoft.Xna.Framework.Audio;
      39  using Microsoft.Xna.Framework.Content;
      40  using Microsoft.Xna.Framework.GamerServices;
      41  using Microsoft.Xna.Framework.Graphics;
      42  using Microsoft.Xna.Framework.Input;
      43  using Microsoft.Xna.Framework.Media;
      44  using Microsoft.Xna.Framework.Net;
      45  using Microsoft.Xna.Framework.Storage;
      46
      47  using Knot3.Core;
      48  using Knot3.Data;
      49  using Knot3.Development;
      50  using Knot3.GameObjects;
      51  using Knot3.Input;
      52  using Knot3.RenderEffects;
      53  using Knot3.Screens;
      54  using Knot3.Utilities;
      55  using Knot3.Widgets;
      56
      57  #endregion
      58
      59  namespace Knot3.Audio
      60  {
      61    /// <summary>
      62    /// Diese Klasse reprsentiert eine Playlist, deren Audiodateien der reihe n
      63    /// Endlosschleife abgespielt werden.
      64    /// </summary>
      65    public class LoopPlaylist : IPlaylist
      66    {
      67      private List<IAudioFile> Sounds;
      68      private int index;
      69
  0   70      public SoundState State { get; private set; }
      71
      72      /// <summary>
      73      /// Erstellt eine neue Playlist.
      74      /// </summary>
      75      /// <param name='sounds'>
      76      /// Die abzuspielenden Audiodateien.
      77      /// </param>
  0   78      public LoopPlaylist (IEnumerable<IAudioFile> sounds)
  0   79      {
  0   80        Sounds = sounds.ToList ();
  0   81        index = 0;
  0   82        State = SoundState.Stopped;
      83
  0   84        Log.Debug ("Created new playlist (", Sounds.Count, " songs)");
  0   85        foreach (IAudioFile sound in Sounds) {
  0   86          Log.Debug ("  - ", sound.Name);
  0   87        }
  0   88      }
      89
      90      public void Shuffle ()
  0   91      {
  0   92        Sounds = Sounds.Shuffle ().ToList ();
  0   93      }
      94
```

```
        95      /// <summary>
        96      /// Starte die Wiedergabe.
        97      /// </summary>
        98      public void Start ()
0       99      {
0      100        if (Sounds.Count > 0) {
0      101          State = SoundState.Playing;
0      102          Sounds .At (index).Play ();
0      103        }
0      104      }
       105
       106      /// <summary>
       107      /// Stoppe die Wiedergabe.
       108      /// </summary>
       109      public void Stop ()
0      110      {
0      111        if (Sounds.Count > 0) {
0      112          State = SoundState.Stopped;
0      113          Sounds.At (index).Stop ();
0      114        }
0      115      }
       116
       117      /// <summary>
       118      /// Wird fr jeden Frame aufgerufen.
       119      /// </summary>
       120      [ExcludeFromCodeCoverageAttribute]
       121      public void Update (GameTime time)
       122      {
       123        if (Sounds.Count > 0) {
       124          if (State == SoundState.Playing && Sounds.At (index).State != SoundSta
       125            ++index;
       126            Sounds.At (index).Play ();
       127          }
       128        }
       129        Sounds.At (index).Update (time);
       130      }
       131    }
       132  }
```

Knot3.Audio.OggVorbisFile

Summary

Class:	Knot3.Audio.OggVorbisFile
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\OggVorbisFile.cs
Coverage:	23.5%
Covered lines:	12
Uncovered lines:	39
Coverable lines:	51
Total lines:	144

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	7	40	9.09
Play()	1	0	0
Stop()	1	0	0
WriteWave(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\OggVorbisFile.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;

```
        34  using System.IO;
        35  using System.Linq;
        36
        37  using Microsoft.Xna.Framework;
        38  using Microsoft.Xna.Framework.Audio;
        39  using Microsoft.Xna.Framework.Content;
        40  using Microsoft.Xna.Framework.GamerServices;
        41  using Microsoft.Xna.Framework.Graphics;
        42  using Microsoft.Xna.Framework.Input;
        43  using Microsoft.Xna.Framework.Media;
        44  using Microsoft.Xna.Framework.Net;
        45  using Microsoft.Xna.Framework.Storage;
        46
        47  using OggSharp;
        48
        49  using Knot3.Core;
        50  using Knot3.Data;
        51  using Knot3.Development;
        52  using Knot3.GameObjects;
        53  using Knot3.Input;
        54  using Knot3.Platform;
        55  using Knot3.RenderEffects;
        56  using Knot3.Screens;
        57  using Knot3.Utilities;
        58  using Knot3.Widgets;
        59
        60  #endregion
        61
        62  namespace Knot3.Audio
        63  {
        64    public class OggVorbisFile : IAudioFile
        65    {
 6      66      public string Name { get; private set; }
        67
 0      68      public SoundState State { get { return internalFile.State; } }
        69
        70      private SoundEffectFile internalFile;
        71
 6      72      public OggVorbisFile (string name, string filepath, Sound soundType)
 6      73      {
 6      74        Name = name;
 6      75        string cachefile = SystemInfo.DecodedMusicCache
        76                           + SystemInfo.PathSeparator.ToString ()
        77                           + soundType.ToString ()
        78                           + "_"
        79                           + name.GetHashCode ().ToString ()
        80                           + ".wav";
        81
        82        byte[] data;
 6      83        try {
 6      84          Log.Debug ("Read from cache: ", cachefile);
 6      85          data = File.ReadAllBytes (cachefile);
 0      86        }
12      87        catch (Exception) {
 6      88          Log.Debug ("Decode: ", name);
 6      89          OggDecoder decoder = new OggDecoder ();
 6      90          decoder.Initialize (TitleContainer.OpenStream (filepath));
 0      91          data = decoder.SelectMany (chunk => chunk.Bytes.Take (chunk.Length)).T
 0      92          using (MemoryStream stream = new MemoryStream ())
 0      93          using (BinaryWriter writer = new BinaryWriter (stream)) {
 0      94            WriteWave (writer, decoder.Stereo ? 2 : 1, decoder.SampleRate, data)
```

```
0    95            stream.Position = 0;
0    96            data = stream.ToArray ();
0    97          }
0    98          File.WriteAllBytes (cachefile, data);
0    99        }
     100
0    101       using (MemoryStream stream = new MemoryStream (data)) {
0    102         stream.Position = 0;
0    103         SoundEffect soundEffect = SoundEffect.FromStream (stream);
0    104         internalFile = new SoundEffectFile (name, soundEffect, soundType);
0    105       }
0    106     }
     107
     108     public void Play ()
0    109     {
0    110       internalFile.Play ();
0    111     }
     112
     113     public void Stop ()
0    114     {
0    115       internalFile.Stop ();
0    116     }
     117
     118     [ExcludeFromCodeCoverageAttribute]
     119     public void Update (GameTime time)
     120     {
     121       internalFile.Update (time);
     122     }
     123
     124     private static void WriteWave (BinaryWriter writer, int channels, int rate
0    125     {
0    126       writer.Write (new char[4] { 'R', 'I', 'F', 'F' });
0    127       writer.Write ((int)(36 + data.Length));
0    128       writer.Write (new char[4] { 'W', 'A', 'V', 'E' });
     129
0    130       writer.Write (new char[4] { 'f', 'm', 't', ' ' });
0    131       writer.Write ((int)16);
0    132       writer.Write ((short)1);
0    133       writer.Write ((short)channels);
0    134       writer.Write ((int)rate);
0    135       writer.Write ((int)(rate * ((16 * channels) / 8)));
0    136       writer.Write ((short)((16 * channels) / 8));
0    137       writer.Write ((short)16);
     138
0    139       writer.Write (new char[4] { 'd', 'a', 't', 'a' });
0    140       writer.Write ((int)data.Length);
0    141       writer.Write (data);
0    142     }
     143   }
     144 }
```

Knot3.Audio.SoundEffectFile

Summary

Class:	Knot3.Audio.SoundEffectFile
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\SoundEffectFile.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	19
Coverable lines:	19
Total lines:	115

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
Play()	1	0	0
Stop()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Audio\SoundEffectFile.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.IO;

```
        35  using System.Linq;
        36
        37  using Microsoft.Xna.Framework;
        38  using Microsoft.Xna.Framework.Audio;
        39  using Microsoft.Xna.Framework.Content;
        40  using Microsoft.Xna.Framework.GamerServices;
        41  using Microsoft.Xna.Framework.Graphics;
        42  using Microsoft.Xna.Framework.Input;
        43  using Microsoft.Xna.Framework.Media;
        44  using Microsoft.Xna.Framework.Net;
        45  using Microsoft.Xna.Framework.Storage;
        46
        47  using Knot3.Core;
        48  using Knot3.Data;
        49  using Knot3.Development;
        50  using Knot3.GameObjects;
        51  using Knot3.Input;
        52  using Knot3.RenderEffects;
        53  using Knot3.Screens;
        54  using Knot3.Utilities;
        55  using Knot3.Widgets;
        56
        57  #endregion
        58
        59  namespace Knot3.Audio
        60  {
        61    /// <summary>
        62    /// Ein Wrapper um die SoundEffect-Klasse des XNA-Frameworks.
        63    /// </summary>
        64    public class SoundEffectFile : IAudioFile
        65    {
        66      /// <summary>
        67      /// Der Anzeigename des SoundEffects.
        68      /// </summary>
    0   69      public string Name { get; private set; }
        70
        71      /// <summary>
        72      /// Gibt an, ob die Wiedergabe luft oder gestoppt bzw. pausiert ist.
        73      /// </summary>
    0   74      public SoundState State { get { return Instance.State; } }
        75
    0   76      public SoundEffect SoundEffect { get; private set; }
        77
        78      private SoundEffectInstance Instance;
        79
        80      private Sound SoundType;
        81      private float volume;
        82
        83      /// <summary>
        84      /// Erstellt eine neue SoundEffect-Datei mit dem angegebenen Anzeigenamen
        85      /// </summary>
    0   86      public SoundEffectFile (string name, SoundEffect soundEffect, Sound soundT
    0   87      {
    0   88        Name = name;
    0   89        SoundEffect = soundEffect;
    0   90        Instance = soundEffect.CreateInstance ();
    0   91        SoundType = soundType;
    0   92      }
        93
        94      public void Play ()
    0   95      {
```

```
0    96        Log.Debug ("Play: ", Name);
0    97        Instance.Volume = volume = AudioManager.Volume (SoundType);
0    98        Instance.Play ();
0    99      }
     100
     101     public void Stop ()
0    102     {
0    103       Log.Debug ("Stop: ", Name);
0    104       Instance.Stop ();
0    105     }
     106
     107     [ExcludeFromCodeCoverageAttribute]
     108     public void Update (GameTime time)
     109     {
     110       if (volume != AudioManager.Volume (SoundType)) {
     111         Instance.Volume = volume = AudioManager.Volume (SoundType);
     112       }
     113     }
     114   }
     115 }
```

Knot3.Core.Angles3

Summary

Class:	Knot3.Core.Angles3
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Angles3.cs
Coverage:	100%
Covered lines:	62
Uncovered lines:	0
Coverable lines:	62
Total lines:	226

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	100	100
FromDegrees(...)	1	100	100
ToDegrees(...)	1	100	100
Equals(...)	3	100	66.67
Equals(...)	1	100	100
op_Equality(...)	3	100	80
op_Inequality(...)	1	100	100
op_Addition(...)	1	100	100
op_UnaryNegation(...	1	100	100
op_Subtraction(...)	1	100	100
op_Multiply(...)	1	100	100
op_Multiply(...)	1	100	100
op_Multiply(...)	1	100	100
op_Division(...)	1	100	100
op_Division(...)	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Angles3.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`

```
22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
23   * SOFTWARE.
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37  using Microsoft.Xna.Framework.Audio;
38  using Microsoft.Xna.Framework.Content;
39  using Microsoft.Xna.Framework.GamerServices;
40  using Microsoft.Xna.Framework.Graphics;
41  using Microsoft.Xna.Framework.Input;
42  using Microsoft.Xna.Framework.Media;
43  using Microsoft.Xna.Framework.Net;
44  using Microsoft.Xna.Framework.Storage;
45
46  using Knot3.Data;
47  using Knot3.GameObjects;
48  using Knot3.RenderEffects;
49  using Knot3.Screens;
50  using Knot3.Widgets;
51
52  #endregion
53
54  namespace Knot3.Core
55  {
56    /// <summary>
57    /// Diese Klasse reprsentiert die Rollwinkel der drei Achsen X, Y und Z.
58    /// Sie bietet Mglichkeit vordefinierte Winkelwerte zu verwenden, z.B. stel
59    /// Die Umwandlung zwischen verschiedenen Winkelmaen wie Grad- und Bogenma
60    /// </summary>
61    public sealed class Angles3 : IEquatable<Angles3>
62    {
63      #region Properties
64
65      /// <summary>
66      /// Der Winkel im Bogenma fr das Rollen um die X-Achse. Siehe statische
67      /// </summary>
158   68      public float X { get; set; }
69
70      /// <summary>
71      /// Der Winkel im Bogenma fr das Rollen um die Y-Achse. Siehe statische
72      /// </summary>
154   73      public float Y { get; set; }
74
75      /// <summary>
76      /// Der Winkel im Bogenma fr das Rollen um die Z-Achse. Siehe statische
77      /// </summary>
154   78      public float Z { get; set; }
79
80      /// <summary>
81      /// Eine statische Eigenschaft mit dem Wert X = 0, Y = 0, Z = 0.
82      /// </summary>
```

```
        83      public static Angles3 Zero
        84      {
54      85        get { return new Angles3 (0f, 0f, 0f); }
        86      }
        87
        88      #endregion
        89
        90      #region Constructors
        91
        92      /// <summary>
        93      /// Konstruiert ein neues Angles3-Objekt mit drei gegebenen Winkeln im Bog
        94      /// </summary>
91      95      public Angles3 (float x, float y, float z)
91      96      {
91      97        X = x;
91      98        Y = y;
91      99        Z = z;
91     100      }
       101
 5     102      public Angles3 (Vector3 v)
 5     103      {
 5     104        X = v.X;
 5     105        Y = v.Y;
 5     106        Z = v.Z;
 5     107      }
       108
       109      #endregion
       110
       111      #region Methods
       112
       113      /// <summary>
       114      /// Eine statische Methode, die Grad in Bogenma konvertiert.
       115      /// </summary>
       116      public static Angles3 FromDegrees (float x, float y, float z)
46     117      {
46     118        return new Angles3 (
       119                    MathHelper.ToRadians (x),
       120                    MathHelper.ToRadians (y),
       121                    MathHelper.ToRadians (z)
       122                );
46     123      }
       124
       125      /// <summary>
       126      /// Konvertiert Bogenma in Grad.
       127      /// </summary>
       128      public void ToDegrees (out float x, out float y, out float z)
 2     129      {
 2     130        x = (int)MathHelper.ToDegrees (X) % 360;
 2     131        y = (int)MathHelper.ToDegrees (Y) % 360;
 2     132        z = (int)MathHelper.ToDegrees (Z) % 360;
 2     133      }
       134
       135      public override bool Equals (object obj)
 1     136      {
 1     137        return (obj is Angles3) ? this == (Angles3)obj : false;
 1     138      }
       139
       140      public bool Equals (Angles3 other)
12     141      {
12     142        return this == other;
12     143      }
```

```
144
145       [ExcludeFromCodeCoverageAttribute]
146       public override int GetHashCode ()
147       {
148         return (int)(this.X + this.Y + this.Z);
149       }
150
151       #endregion
152
153       #region Operators
154
155       public static bool operator == (Angles3 value1, Angles3 value2)
14    156 {
14    157   return value1.X == value2.X
158                && value1.Y == value2.Y
159                && value1.Z == value2.Z;
14    160 }
161
162       public static bool operator != (Angles3 value1, Angles3 value2)
 1    163 {
 1    164   return !(value1 == value2);
 1    165 }
166
167       public static Angles3 operator + (Angles3 value1, Angles3 value2)
 6    168 {
 6    169   return new Angles3 (value1.X + value2.X, value1.Y + value2.Y, value1.Z +
 6    170 }
171
172       public static Angles3 operator - (Angles3 value)
 1    173 {
 1    174   value = new Angles3 (-value.X, -value.Y, -value.Z);
 1    175   return value;
 1    176 }
177
178       public static Angles3 operator - (Angles3 value1, Angles3 value2)
 1    179 {
 1    180   return new Angles3 (value1.X - value2.X, value1.Y - value2.Y, value1.Z -
 1    181 }
182
183       public static Angles3 operator * (Angles3 value1, Angles3 value2)
 1    184 {
 1    185   return new Angles3 (value1.X * value2.X, value1.Y * value2.Y, value1.Z *
 1    186 }
187
188       public static Angles3 operator * (Angles3 value, float scaleFactor)
 1    189 {
 1    190   return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
 1    191 }
192
193       public static Angles3 operator * (float scaleFactor, Angles3 value)
 1    194 {
 1    195   return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
 1    196 }
197
198       public static Angles3 operator / (Angles3 value1, Angles3 value2)
 1    199 {
 1    200   return new Angles3 (value1.X / value2.X, value1.Y / value2.Y, value1.Z /
 1    201 }
202
203       public static Angles3 operator / (Angles3 value, float divider)
 1    204 {
```

```
1  205        float scaleFactor = 1 / divider;
1  206        return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
1  207      }
   208
   209      [ExcludeFromCodeCoverageAttribute]
   210      public override string ToString ()
   211      {
   212        float x, y, z;
   213        ToDegrees (out x, out y, out z);
   214
   215        return   "Angles3 ("
   216                 + x.ToString ()
   217                 + ","
   218                 + y.ToString ()
   219                 + ","
   220                 + z.ToString ()
   221                 + ")";
   222      }
   223
   224      #endregion
   225    }
   226  }
```

Knot3.Core.BooleanOptionInfo

Summary

Class:	Knot3.Core.BooleanOptionInfo
Assembly:	Knot3
File(s):	:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\BooleanOptionInfo.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	10
Coverable lines:	10
Total lines:	97

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	0	0
.cctor()	1	0	0

File(s)

:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\BooleanOptionInfo.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.Linq;
	35	

```
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.RenderEffects;
      49  using Knot3.Screens;
      50  using Knot3.Widgets;
      51
      52  #endregion
      53
      54  namespace Knot3.Core
      55  {
      56    /// <summary>
      57    /// Diese Klasse reprsentiert eine Option, welche die Werte \glqq Wahr\grqq
      58    /// </summary>
      59    public sealed class BooleanOptionInfo : DistinctOptionInfo
      60    {
      61      #region Properties
      62
      63      /// <summary>
      64      /// Eine Eigenschaft, die den aktuell abgespeicherten Wert zurckgibt.
      65      /// </summary>
      66      public new bool Value
      67      {
  0   68        get {
  0   69          return base.Value == ConfigFile.True ? true : false;
  0   70        }
  0   71        set {
  0   72          base.Value = value ? ConfigFile.True : ConfigFile.False;
  0   73        }
      74      }
      75
  0   76      public new static string[] ValidValues = new string[] {
      77        ConfigFile.True,
      78        ConfigFile.False
      79      };
      80
      81      #endregion
      82
      83      #region Constructors
      84
      85      /// <summary>
      86      /// Erstellt eine neue Option, welche die Werte \glqq Wahr\grqq oder \glqq
      87      /// angegebenen Abschnitt der angegebenen Einstellungsdatei.
      88      /// [base=section, name, defaultValue?ConfigFile.True:ConfigFile.False, Va
      89      /// </summary>
  0   90      public BooleanOptionInfo (string section, string name, bool defaultValue,
      91      : base (section, name, defaultValue?ConfigFile.True:ConfigFile.False, Vali
  0   92      {
  0   93      }
      94
      95      #endregion
      96    }
```

```
97  }
```

Knot3.Core.Camera

Summary

Class:	Knot3.Core.Camera
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Camera.cs
Coverage:	65.3%
Covered lines:	83
Uncovered lines:	44
Coverable lines:	127
Total lines:	381

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	100	100
UpdateMatrices(...)	1	100	100
GetMouseRay(...)	1	0	0
ResetCamera()	1	100	100
StartSmoothMove(...)	2	0	0
UpdateSmoothMove(...	2	0	0
To3D(...)	2	100	100
To2D(...)	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Camera.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
```

```
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37  using Microsoft.Xna.Framework.Audio;
38  using Microsoft.Xna.Framework.Content;
39  using Microsoft.Xna.Framework.GamerServices;
40  using Microsoft.Xna.Framework.Graphics;
41  using Microsoft.Xna.Framework.Input;
42  using Microsoft.Xna.Framework.Media;
43  using Microsoft.Xna.Framework.Net;
44  using Microsoft.Xna.Framework.Storage;
45
46  using Knot3.Data;
47  using Knot3.GameObjects;
48  using Knot3.RenderEffects;
49  using Knot3.Screens;
50  using Knot3.Utilities;
51  using Knot3.Widgets;
52
53  #endregion
54
55  namespace Knot3.Core
56  {
57    /// <summary>
58    /// Jede Instanz der World-Klasse hlt eine fr diese Spielwelt verwendete K
59    /// Die Hauptfunktion der Kamera-Klasse ist das Berechnen der drei Matrizen,
60    /// und Skalierung von 3D-Objekten in einer bestimmten Spielwelt bentigt we
61    /// Um diese Matrizen zu berechnen, bentigt die Kamera unter Anderem Inform
62    /// das aktuelle Kamera-Ziel und das Field of View.
63    /// </summary>
64    public sealed class Camera : GameScreenComponent
65    {
66      #region Properties
67
68      private Vector3 _position;
69
70      /// <summary>
71      /// Die Position der Kamera.
72      /// </summary>
73      public Vector3 Position
74      {
75  75        get { return _position; }
16  76        set {
16  77          OnViewChanged ();
16  78          if ((value.X.Abs () <= MaxPositionDistance && value.Y.Abs () <= MaxPos
16  79                  && value.Z.Abs () <= MaxPositionDistance) || MaxPositionDistan
16  80            _position = value;
16  81          }
16  82        }
83      }
84
85      private Vector3 _target;
86
87      /// <summary>
88      /// Das Ziel der Kamera.
89      /// </summary>
90      public Vector3 Target
```

```
      91      {
81    92        get { return _target; }
12    93        set {
12    94          OnViewChanged ();
12    95          _target = value;
12    96        }
      97      }
      98
      99      private float _foV;
     100
     101      /// <summary>
     102      /// Das Sichtfeld.
     103      /// </summary>
     104      public float FoV
     105      {
36   106        get { return _foV; }
11   107        set {
11   108          _foV = MathHelper.Clamp (value, 10, 70);
11   109          OnViewChanged ();
11   110        }
     111      }
     112
     113      /// <summary>
     114      /// Die View-Matrix wird ber die statische Methode CreateLookAt der Klass
     115      /// mit Matrix.CreateLookAt (Position, Target, Vector3.Up) berechnet.
     116      /// </summary>
23   117      public Matrix ViewMatrix { get; private set; }
     118
     119      /// <summary>
     120      /// Die World-Matrix wird mit Matrix.CreateFromYawPitchRoll und den drei R
     121      /// </summary>
12   122      public Matrix WorldMatrix { get; private set; }
     123
     124      /// <summary>
     125      /// Die Projektionsmatrix wird ber die statische XNA-Methode Matrix.Creat
     126      /// </summary>
23   127      public Matrix ProjectionMatrix { get; private set; }
     128
     129      /// <summary>
     130      /// Berechnet ein Bounding-Frustum, das bentigt wird, um festzustellen, o
     131      /// </summary>
 9   132      public BoundingFrustum ViewFrustum { get; private set; }
     133
     134      /// <summary>
     135      /// Eine Referenz auf die Spielwelt, fr welche die Kamera zustndig ist.
     136      /// </summary>
29   137      private World World { get; set; }
     138
     139      /// <summary>
     140      /// Die Rotationswinkel.
     141      /// </summary>
34   142      public Angles3 Rotation { get; set; }
     143
16   144      public Vector3 UpVector { get; private set; }
     145
56   146      public float MaxPositionDistance { get; set; }
     147
47   148      public Action OnViewChanged = () => {};
     149      private float aspectRatio;
     150      private float nearPlane;
     151      private float farPlane;
```

```
8   152      private Vector3 defaultPosition = new Vector3 (400, 400, 700);
    153
    154      #endregion
    155
    156      #region Constructors
    157
    158      /// <summary>
    159      /// Erstellt eine neue Kamera in einem bestimmten IGameScreen fr eine bes
    160      /// </summary>
8   161      public Camera (IGameScreen screen, World world)
    162      : base (screen, DisplayLayer.None)
8   163      {
8   164        World = world;
8   165        Position = defaultPosition;
8   166        Target = Vector3.Zero;
8   167        UpVector = Vector3.Up;
8   168        Rotation = Angles3.Zero;
8   169        MaxPositionDistance = 5000;
    170
8   171        FoV = 60;
8   172        nearPlane = 0.5f;
8   173        farPlane = 15000.0f;
    174
8   175        UpdateMatrices (null);
8   176      }
    177
    178      #endregion
    179
    180      #region Methods
    181
    182      /// <summary>
    183      /// Die Blickrichtung.
    184      /// </summary>
    185      public Vector3 PositionToTargetDirection
    186      {
0   187        get {
0   188          return Vector3.Normalize (Target - Position);
0   189        }
    190      }
    191
    192      public Vector3 PositionToArcballTargetDirection
    193      {
0   194        get {
0   195          return Vector3.Normalize (ArcballTarget - Position);
0   196        }
    197      }
    198
    199      /// <summary>
    200      /// Der Abstand zwischen der Kamera und dem Kamera-Ziel.
    201      /// </summary>
    202      public float PositionToTargetDistance
    203      {
4   204        get {
4   205          return Position.DistanceTo (Target);
4   206        }
4   207        set {
4   208          Position = Position.SetDistanceTo (Target, value);
4   209        }
    210      }
    211
    212      public float PositionToArcballTargetDistance
```

```
      213       {
    0 214         get {
    0 215           return Position.DistanceTo (ArcballTarget);
    0 216         }
    0 217         set {
    0 218           Position = Position.SetDistanceTo (ArcballTarget, value);
    0 219         }
      220       }
      221
      222       /// <summary>
      223       /// Wird fr jeden Frame aufgerufen.
      224       /// </summary>
      225       [ExcludeFromCodeCoverageAttribute]
      226       public override void Update (GameTime time)
      227       {
      228         // Setze den Viewport auf den der aktuellen Spielwelt
      229         Viewport original = Screen.Viewport;
      230         Screen.Viewport = World.Viewport;
      231
      232         UpdateMatrices (time);
      233         UpdateSmoothMove (time);
      234
      235         // Setze den Viewport wieder auf den ganzen Screen
      236         Screen.Viewport = original;
      237       }
      238
      239       private void UpdateMatrices (GameTime time)
    8 240       {
    8 241         aspectRatio = Screen.Viewport.AspectRatio;
    8 242         farPlane = MaxPositionDistance * 4;
    8 243         ViewMatrix = Matrix.CreateLookAt (Position, Target, UpVector);
    8 244         WorldMatrix = Matrix.CreateFromYawPitchRoll (Rotation.Y, Rotation.X, Rot
    8 245         ProjectionMatrix = Matrix.CreatePerspectiveFieldOfView (MathHelper.ToRad
    8 246         ViewFrustum = new BoundingFrustum (ViewMatrix * ProjectionMatrix);
    8 247       }
      248
      249       /// <summary>
      250       /// Berechnet einen Strahl fr die angegebenene 2D-Mausposition.
      251       /// </summary>
      252       public Ray GetMouseRay (Vector2 mousePosition)
    0 253       {
    0 254         Viewport viewport = World.Viewport;
      255
    0 256         Vector3 nearPoint = new Vector3 (mousePosition, 0);
    0 257         Vector3 farPoint = new Vector3 (mousePosition, 1);
      258
    0 259         nearPoint = viewport.Unproject (nearPoint, ProjectionMatrix, ViewMatrix,
    0 260         farPoint = viewport.Unproject (farPoint, ProjectionMatrix, ViewMatrix, M
      261
    0 262         Vector3 direction = farPoint - nearPoint;
    0 263         direction.Normalize ();
      264
    0 265         return new Ray (nearPoint, direction);
    0 266       }
      267
      268       /// <summary>
      269       /// Eine Position, um die rotiert werden soll, wenn der User die rechte Ma
      270       /// </summary>
      271       public Vector3 ArcballTarget
      272       {
    0 273         get {
```

```
0   274          if (World.SelectedObject != null) {
0   275            return World.SelectedObject.Center ();
    276          }
0   277          else {
0   278            return Vector3.Zero;
    279          }
0   280        }
    281      }
    282
    283      public void ResetCamera ()
2   284      {
2   285        Position = defaultPosition;
2   286        Target = new Vector3 (0, 0, 0);
2   287        Rotation = Angles3.Zero;
2   288        FoV = 45;
2   289      }
    290
8   291      private Vector3? smoothTarget = null;
8   292      private float smoothDistance = 0f;
8   293      private float smoothProgress = 0f;
    294
    295      public void StartSmoothMove (Vector3 target, GameTime time)
0   296      {
0   297        if (!InSmoothMove) {
0   298          smoothTarget = target;
0   299          smoothDistance = Math.Abs (Target.DistanceTo (target));
0   300          smoothProgress = 0f;
0   301        }
0   302      }
    303
0   304      public bool InSmoothMove { get { return smoothTarget.HasValue && smoothPro
    305
    306      private void UpdateSmoothMove (GameTime time)
0   307      {
0   308        if (InSmoothMove) {
0   309          float distance = MathHelper.SmoothStep (0, smoothDistance, smoothProgr
    310
0   311          smoothProgress += 0.05f;
    312
    313          //Log.Debug ("distance = ", distance);
0   314          Target = Target.SetDistanceTo (
    315                       target: smoothTarget.Value,
    316                       distance: Math.Max (0, smoothDistance - distance)
    317                   );
0   318          World.Redraw = true;
0   319        }
0   320      }
    321
    322      /// <summary>
    323      /// Berechne aus einer 2D-Positon (z.b. Mausposition) die entsprechende Po
    324      /// Fr die fehlende dritte Koordinate wird eine Angabe einer weiteren 3D-
    325      /// mit der die 3D-(Maus-)Position auf der selben Ebene liegen soll.
    326      /// </summary>
    327      public Vector3 To3D (Vector2 position, Vector3 nearTo)
2   328      {
3   329        if (Options.Default ["debug", "unproject", "SelectedObject"] == "NearFar
1   330          Vector3 nearScreenPoint = new Vector3 (position.X, position.Y, 0);
1   331          Vector3 farScreenPoint = new Vector3 (position.X, position.Y, 1);
1   332          Vector3 nearWorldPoint = World.Viewport.Unproject (
    333                                       source: nearScreenPoint,
    334                                       projection: World.Camera.ProjectionMatrix
```

```
      335                                     view: World.Camera.ViewMatrix,
      336                                     world: Matrix.Identity
      337                                 );
    1 338           Vector3 farWorldPoint = World.Viewport.Unproject (
      339                                     source: farScreenPoint,
      340                                     projection: World.Camera.ProjectionMatrix,
      341                                     view: World.Camera.ViewMatrix,
      342                                     world: Matrix.Identity
      343                                 );
      344
    1 345           Vector3 direction = farWorldPoint - nearWorldPoint;
      346
    1 347           float zFactor = -nearWorldPoint.Y / direction.Y;
    1 348           Vector3 zeroWorldPoint = nearWorldPoint + direction * zFactor;
    1 349           return zeroWorldPoint;
      350         }
    1 351         else {
    1 352           Vector3 screenLocation = World.Viewport.Project (
      353                                       source: nearTo,
      354                                       projection: World.Camera.ProjectionMatrix
      355                                       view: World.Camera.ViewMatrix,
      356                                       world: World.Camera.WorldMatrix
      357                                   );
    1 358           Vector3 currentMousePosition = World.Viewport.Unproject (
      359                                            source: new Vector3 (position, scre
      360                                            projection: World.Camera.Projection
      361                                            view: World.Camera.ViewMatrix,
      362                                            world: Matrix.Identity
      363                                       );
    1 364           return currentMousePosition;
      365         }
    2 366       }
      367
      368       public Vector2 To2D (Vector3 position)
    2 369       {
    2 370         Vector3 screenLocation = World.Viewport.Project (
      371                                     source: position,
      372                                     projection: World.Camera.ProjectionMatrix,
      373                                     view: World.Camera.ViewMatrix,
      374                                     world: World.Camera.WorldMatrix
      375                                 );
    2 376         return new Vector2 (screenLocation.X, screenLocation.Y);
    2 377       }
      378
      379       #endregion
      380     }
      381   }
```

Knot3.Core.ConfigFile

Summary

Class:	Knot3.Core.ConfigFile
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\ConfigFile.cs
Coverage:	100%
Covered lines:	55
Uncovered lines:	0
Coverable lines:	55
Total lines:	188

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	100	66.67
SetOption(...)	1	100	100
GetOption(...)	1	100	100
SetOption(...)	2	100	100
GetOption(...)	4	100	80
SetOption(...)	1	100	100
GetOption(...)	1	100	100
floatToString(...)	1	100	100
stringToFloat(...)	2	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\ConfigFile.cs

```
#   Line  Coverage
      1  #region Copyright
      2
      3  /*
      4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
      5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
      6   *
      7   * Permission is hereby granted, free of charge, to any person obtaining a cop
      8   * of this software and associated documentation files (the "Software"), to de
      9   * in the Software without restriction, including without limitation the right
     10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
     11   * copies of the Software, and to permit persons to whom the Software is
     12   * furnished to do so, subject to the following conditions:
     13   *
     14   * The above copyright notice and this permission notice shall be included in
     15   * copies or substantial portions of the Software.
     16   *
     17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
     18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
     19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
     20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
     22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
     23   * SOFTWARE.
     24   */
     25
     26  #endregion
     27
     28  #region Using
```

```
        29
        30  using System;
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.IO;
        35  using System.Linq;
        36
        37  using Microsoft.Xna.Framework;
        38  using Microsoft.Xna.Framework.Audio;
        39  using Microsoft.Xna.Framework.Content;
        40  using Microsoft.Xna.Framework.GamerServices;
        41  using Microsoft.Xna.Framework.Graphics;
        42  using Microsoft.Xna.Framework.Input;
        43  using Microsoft.Xna.Framework.Media;
        44  using Microsoft.Xna.Framework.Net;
        45  using Microsoft.Xna.Framework.Storage;
        46
        47  using Knot3.Data;
        48  using Knot3.GameObjects;
        49  using Knot3.RenderEffects;
        50  using Knot3.Screens;
        51  using Knot3.Utilities;
        52  using Knot3.Widgets;
        53
        54  #endregion
        55
        56  namespace Knot3.Core
        57  {
        58    /// <summary>
        59    /// Reprsentiert eine Einstellungsdatei.
        60    /// </summary>
        61    public sealed class ConfigFile
        62    {
        63      #region Properties
        64
        65      /// <summary>
        66      /// Die Reprsentation des Wahrheitswerts "wahr" als String in einer Einst
        67      /// </summary>
  447   68      public static string True { get { return "true"; } }
        69
        70      /// <summary>
        71      /// Die Reprsentation des Wahrheitswerts "falsch" als String in einer Ein
        72      /// </summary>
  453   73      public static string False { get { return "false"; } }
        74
        75      private string Filename;
        76      private IniFile ini;
        77
        78      #endregion
        79
        80      #region Constructors
        81
    3   82      public ConfigFile (string filename)
    3   83      {
        84        // load ini file
    3   85        Filename = filename;
        86
        87        // create a new ini parser
    6   88        using (StreamWriter w = File.AppendText (Filename)) {
    3   89        }
```

```
  3    90        ini = new IniFile (Filename);
  3    91      }
       92
       93      #endregion
       94
       95      #region Methods
       96
       97      /// <summary>
       98      /// Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen
       99      /// </summary>
      100      public void SetOption (string section, string option, string _value)
304   101      {
304   102        ini [section, option] = _value;
304   103      }
      104
      105      /// <summary>
      106      /// Gibt den aktuell in der Datei vorhandenen Wert fr die angegebene Opti
      107      /// </summary>
      108      public string GetOption (string section, string option, string defaultValu
525   109      {
525   110        return ini [section, option, defaultValue];
525   111      }
      112
      113      /// <summary>
      114      /// Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen
      115      /// </summary>
      116      public void SetOption (string section, string option, bool _value)
100   117      {
100   118        SetOption (section, option, _value ? True : False);
100   119      }
      120
      121      /// <summary>
      122      /// Gibt den aktuell in der Datei vorhandenen Wert fr die angegebene Opti
      123      /// </summary>
      124      public bool GetOption (string section, string option, bool defaultValue)
100   125      {
100   126        return GetOption (section, option, defaultValue ? True : False) == True
100   127      }
      128
      129      public void SetOption (string section, string option, float _value)
101   130      {
101   131        SetOption (section, option, floatToString (_value));
101   132      }
      133
      134      public float GetOption (string section, string option, float defaultValue)
218   135      {
218   136        return stringToFloat (GetOption (section, option, floatToString (default
218   137      }
      138
      139      private string floatToString (float f)
319   140      {
319   141        return String.Empty + ((int) (f * 1000)).ToString ();
319   142      }
      143
      144      private float stringToFloat (string s)
218   145      {
      146        int i;
218   147        bool result = Int32.TryParse (s, out i);
336   148        if (true == result) {
118   149          return ((float)i) / 1000f;
      150        }
```

```
100   151        else {
100   152          return 0;
      153        }
218   154      }
      155
      156      public bool this [string section, string option, bool defaultValue = false
      157      {
100   158        get {
100   159          return GetOption (section, option, defaultValue);
100   160        }
100   161        set {
100   162          SetOption (section, option, value);
100   163        }
      164      }
      165
      166      public float this [string section, string option, float defaultValue = 0f]
      167      {
218   168        get {
218   169          return GetOption (section, option, defaultValue);
218   170        }
101   171        set {
101   172          SetOption (section, option, value);
101   173        }
      174      }
      175
      176      public string this [string section, string option, string defaultValue = n
      177      {
207   178        get {
207   179          return GetOption (section, option, defaultValue);
207   180        }
103   181        set {
103   182          SetOption (section, option, value);
103   183        }
      184      }
      185
      186      #endregion
      187    }
      188  }
```

Knot3.Core.DisplayLayer

Summary

Class:	Knot3.Core.DisplayLayer
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\DisplayLayer.cs
Coverage:	98.1%
Covered lines:	52
Uncovered lines:	1
Coverable lines:	53
Total lines:	207

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	100	100
op_Addition(...)	1	100	100
op_Addition(...)	1	100	100
op_Multiply(...)	1	100	100
op_Equality(...)	4	88.89	71.43
op_Inequality(...)	1	100	100
Equals(...)	2	100	100
Equals(...)	2	100	66.67
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\DisplayLayer.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                          Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
```

```
        26  #endregion
        27
        28  #region Using
        29
        30  using System;
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.Linq;
        35
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Data;
        47  using Knot3.GameObjects;
        48  using Knot3.RenderEffects;
        49  using Knot3.Screens;
        50  using Knot3.Widgets;
        51
        52  #endregion
        53
        54  namespace Knot3.Core
        55  {
        56    /// <summary>
        57    /// Die Zeichenreihenfolge der Elemente der grafischen Benutzeroberflche.
        58    /// </summary>
        59    public class DisplayLayer : IEquatable<DisplayLayer>
        60    {
        61      #region Enumeration Values
        62
        63      /// <summary>
        64      /// Steht fr die hinterste Ebene bei der Zeichenreihenfolge.
        65      /// </summary>
    1   66      public static readonly DisplayLayer None = new DisplayLayer (0, "None");
        67      /// <summary>
        68      /// Steht fr eine Ebene hinter der Spielwelt, z.B. um
        69      /// Hintergrundbilder darzustellen.
        70      /// </summary>
    1   71      public static readonly DisplayLayer Background = new DisplayLayer (10, "Ba
        72      /// <summary>
        73      /// Steht fr die Ebene in der die Spielwelt dargestellt wird.
        74      /// </summary>
    1   75      public static readonly DisplayLayer GameWorld = new DisplayLayer (20, "Gam
    1   76      public static readonly DisplayLayer ScreenUI = new DisplayLayer (30, "Scre
        77      /// <summary>
        78      /// Steht fr die Ebene in der die Dialoge dargestellt werden.
        79      /// Dialoge werden vor der Spielwelt gezeichnet, damit der Spieler damit i
        80      /// </summary>
    1   81      public static readonly DisplayLayer Dialog = new DisplayLayer (50, "Dialog
        82      /// <summary>
        83      /// Steht fr die Ebene in der Mens gezeichnet werden. Mens werden inner
        84      /// </summary>
    1   85      public static readonly DisplayLayer Menu = new DisplayLayer (10, "Menu");
        86      /// <summary>
```

```
        87      /// Steht fr die Ebene in der Meneintrge gezeichnet werden. Meneintrg
        88      /// </summary>
     1  89      public static readonly DisplayLayer MenuItem = new DisplayLayer (20, "Menu
        90      /// <summary>
        91      /// Zum Anzeigen zustzlicher Informationen bei der (Weiter-)Entwicklung o
        92      /// </summary>
     1  93      public static readonly DisplayLayer Overlay = new DisplayLayer (300, "Over
        94      /// <summary>
        95      /// Die Maus ist das Hauptinteraktionswerkzeug, welches der Spieler
        96      /// stndig verwendet. Daher muss die Maus bei der Interaktion immer
        97      /// im Vordergrund sein. Cursor steht fr die vorderste Ebene.
        98      /// </summary>
     1  99      public static readonly DisplayLayer Cursor = new DisplayLayer (500, "Curso
       100
     1 101      public static readonly DisplayLayer[] Values = {
       102        None, Background, GameWorld, ScreenUI, Dialog, Menu, MenuItem, Overlay,
       103      };
       104
       105      #endregion
       106
       107      #region Static Attributes
       108
       109      #endregion
       110
       111      #region Properties
       112
  1825 113      public int Index { get; private set; }
       114
  1001 115      public string Description { get; private set; }
       116
       117      #endregion
       118
       119      #region Constructors
       120
    18 121      private DisplayLayer (int index, string desciption)
    18 122      {
    18 123        Index = index;
    18 124        Description = desciption;
    18 125      }
       126
   324 127      private DisplayLayer (DisplayLayer layer1, DisplayLayer layer2)
   324 128      {
   324 129        Index = layer1.Index + layer2.Index;
   324 130        Description = layer1.Description + "+" + layer2.Description;
   324 131      }
       132
       133      #endregion
       134
       135      #region Methods and Operators
       136
       137      [ExcludeFromCodeCoverageAttribute]
       138      public override string ToString ()
       139      {
       140        return Description;
       141      }
       142
       143      public static DisplayLayer operator + (DisplayLayer layer1, DisplayLayer l
   243 144      {
   243 145        return new DisplayLayer (layer1, layer2);
   243 146      }
       147
```

```
        148      public static DisplayLayer operator + (DisplayLayer layer, Widget widget)
   81   149      {
   81   150        return new DisplayLayer (widget.Index, layer);
   81   151      }
        152
        153      public static DisplayLayer operator * (DisplayLayer layer, int i)
    9   154      {
    9   155        return new DisplayLayer (layer.Index * i, "(" + layer + "*" + i + ")");
    9   156      }
        157
        158      public static bool operator == (DisplayLayer a, DisplayLayer b)
   27   159      {
        160        // If both are null, or both are same instance, return true.
   36   161        if (System.Object.ReferenceEquals (a, b)) {
    9   162          return true;
        163        }
        164
        165        // If one is null, but not both, return false.
   36   166        if (((object)a == null) || ((object)b == null)) {
   18   167          return false;
        168        }
        169
        170        // Return true if the fields match:
    0   171        return a.Index == b.Index;
   27   172      }
        173
        174      public static bool operator != (DisplayLayer d1, DisplayLayer d2)
   27   175      {
   27   176        return !(d1 == d2);
   27   177      }
        178
        179      public bool Equals (DisplayLayer other)
   27   180      {
   27   181        return other != null && Index == other.Index;
   27   182      }
        183
        184      public override bool Equals (object other)
    9   185      {
    9   186        return other != null && Equals (other as DisplayLayer);
    9   187      }
        188
        189      public static implicit operator string (DisplayLayer layer)
    9   190      {
    9   191        return layer.Description;
    9   192      }
        193
        194      public static implicit operator int (DisplayLayer layer)
  601   195      {
  601   196        return layer.Index;
  601   197      }
        198
        199      [ExcludeFromCodeCoverageAttribute]
        200      public override int GetHashCode ()
        201      {
        202        return Description.GetHashCode ();
        203      }
        204
        205      #endregion
        206    }
        207  }
```

Knot3.Core.DistinctOptionInfo

Summary

Class:	Knot3.Core.DistinctOptionInfo
Assembly:	Knot3
File(s):	\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\DistinctOptionInfo.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	22
Coverable lines:	22
Total lines:	112

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	3	0	0

File(s)

\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\DistinctOptionInfo.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`
	32	`using System.Collections.Generic;`
	33	`using System.Diagnostics.CodeAnalysis;`
	34	`using System.Linq;`
	35	
	36	`using Microsoft.Xna.Framework;`
	37	`using Microsoft.Xna.Framework.Audio;`

```
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.RenderEffects;
      49  using Knot3.Screens;
      50  using Knot3.Widgets;
      51
      52  #endregion
      53
      54  namespace Knot3.Core
      55  {
      56    /// <summary>
      57    /// Diese Klasse reprsentiert eine Option, die einen Wert aus einer distink
      58    /// </summary>
      59    public class DistinctOptionInfo : OptionInfo
      60    {
      61      #region Properties
      62
      63      /// <summary>
      64      /// Eine Menge von Texten, welche die fr die Option gltigen Werte beschr
      65      /// </summary>
    0 66      public HashSet<string> ValidValues { get; private set; }
      67
    0 68      public virtual Dictionary<string,string> DisplayValidValues { get; private
      69      /// <summary>
      70      /// Eine Eigenschaft, die den aktuell abgespeicherten Wert zurck gibt.
      71      /// </summary>
      72      public override string Value
      73      {
    0 74        get {
    0 75          return base.Value;
    0 76        }
    0 77        set {
    0 78          if (ValidValues.Contains (value)) {
    0 79            base.Value = value;
    0 80          }
    0 81          else {
    0 82            base.Value = DefaultValue;
    0 83          }
    0 84        }
      85      }
      86      public virtual string DisplayValue
      87      {
    0 88        get {
    0 89          return Value;
    0 90        }
      91      }
      92
      93      #endregion
      94
      95      #region Constructors
      96
      97      /// <summary>
      98      /// Erstellt eine neue Option, die einen der angegebenen Werte aus validVa
```

```
        99      /// angegebenen Abschnitt der angegebenen Einstellungsdatei.
       100      /// [base=section, name, defaultValue, configFile]
       101      /// </summary>
  0    102      public DistinctOptionInfo (string section, string name, string defaultValu
       103      : base (section, name, defaultValue, configFile)
  0    104      {
  0    105        ValidValues = new HashSet<string> (validValues);
  0    106        ValidValues.Add (defaultValue);
  0    107        DisplayValidValues = new Dictionary<string,string> (ValidValues.ToDictio
  0    108      }
       109
       110      #endregion
       111    }
       112  }
```

Knot3.Core.FloatOptionInfo

Summary

Class:	Knot3.Core.FloatOptionInfo
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\FloatOptionInfo.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	25
Coverable lines:	25
Total lines:	121

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
convertToString(...)	1	0	0
stringToFloat(...)	2	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\FloatOptionInfo.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
      31  using System.Collections;
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.Linq;
```

```
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.RenderEffects;
      49  using Knot3.Screens;
      50  using Knot3.Widgets;
      51
      52  #endregion
      53
      54  namespace Knot3.Core
      55  {
      56    /// <summary>
      57    /// Diese Klasse reprsentiert eine Option, welche die Werte \glqq Wahr\grqq
      58    /// </summary>
      59    public sealed class FloatOptionInfo : DistinctOptionInfo
      60    {
      61      #region Properties
      62
      63      /// <summary>
      64      /// Eine Eigenschaft, die den aktuell abgespeicherten Wert zurckgibt.
      65      /// </summary>
      66      public new float Value
      67      {
  0   68        get {
  0   69          return stringToFloat (base.Value);
  0   70        }
  0   71        set {
  0   72          base.Value = convertToString (value);
  0   73        }
      74      }
      75
      76      public override string DisplayValue
      77      {
  0   78        get {
  0   79          return String.Empty + stringToFloat (base.Value);
  0   80        }
      81      }
      82
      83      public override Dictionary<string,string> DisplayValidValues
      84      {
  0   85        get {
  0   86          return new Dictionary<string, string>(base.ValidValues.ToDictionary (s
  0   87        }
      88      }
      89
      90      #endregion
      91
      92      #region Constructors
      93
      94      /// <summary>
      95      /// Erstellt eine neue Option, welche die Werte \glqq Wahr\grqq oder \glqq
```

```
        96      /// angegebenen Abschnitt der angegebenen Einstellungsdatei.
        97      /// [base=section, name, defaultValue?ConfigFile.True:ConfigFile.False, Va
        98      /// </summary>
0       99      public FloatOptionInfo (string section, string name, float defaultValue, I
       100      : base (section, name, convertToString ( defaultValue),validValues.Select
0      101      {
0      102      }
       103
       104      private static string convertToString (float f)
0      105      {
0      106        return (String.Empty + (int)(f * 1000f));
0      107      }
       108      private static float stringToFloat (string s)
0      109      {
       110        int i;
0      111        bool result = Int32.TryParse (s, out i);
0      112        if (true == result) {
0      113          return ((float)i) / 1000f;
       114        }
0      115        else {
0      116          return 0;
       117        }
0      118      }
       119      #endregion
       120    }
       121  }
```

Knot3.Core.KeyOptionInfo

Summary

Class:	Knot3.Core.KeyOptionInfo
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\KeyOptionInfo.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	10
Coverable lines:	10
Total lines:	87

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
.cctor()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\KeyOptionInfo.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.Linq;
	35	

```
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Data;
        47  using Knot3.GameObjects;
        48  using Knot3.RenderEffects;
        49  using Knot3.Screens;
        50  using Knot3.Utilities;
        51  using Knot3.Widgets;
        52
        53  #endregion
        54
        55  namespace Knot3.Core
        56  {
        57    public class KeyOptionInfo : DistinctOptionInfo
        58    {
        59      #region Properties
        60
        61      /// <summary>
        62      /// Eine Eigenschaft, die den aktuell abgespeicherten Wert zurckgibt.
        63      /// </summary>
        64      public new Keys Value
        65      {
    0   66        get {
    0   67          return base.Value.ToEnumValue<Keys> ();
    0   68        }
    0   69        set {
    0   70          base.Value = value.ToEnumDescription<Keys> ();
    0   71        }
        72      }
        73
    0   74      public new static IEnumerable<string> ValidValues = typeof (Keys).ToEnumVa
        75
        76      #endregion
        77
        78      #region Constructors
        79
    0   80      public KeyOptionInfo (string section, string name, Keys defaultValue, Conf
        81      : base (section, name, defaultValue.ToEnumDescription<Keys> (), ValidValue
    0   82      {
    0   83      }
        84
        85      #endregion
        86    }
        87  }
```

Knot3.Core.Localizer

Summary

Class:	Knot3.Core.Localizer
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Localizer.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	3
Coverable lines:	3
Total lines:	83

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
Localize(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Localizer.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`
	32	`using System.Collections.Generic;`
	33	`using System.Diagnostics.CodeAnalysis;`
	34	`using System.Linq;`
	35	
	36	`using Microsoft.Xna.Framework;`
	37	`using Microsoft.Xna.Framework.Audio;`

```
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.RenderEffects;
      49  using Knot3.Screens;
      50  using Knot3.Widgets;
      51
      52  #endregion
      53
      54  namespace Knot3.Core
      55  {
      56    /// <summary>
      57    /// Eine statische Klasse, die Bezeichner in lokalisierten Text umsetzen kan
      58    /// </summary>
      59    public static class Localizer
      60    {
      61      #region Properties
      62
      63      /// <summary>
      64      /// Die Datei, welche Informationen fr die Lokalisierung enthlt.
      65      /// </summary>
0     66      private static ConfigFile localization { get; set; }
      67
      68      #endregion
      69
      70      #region Methods
      71
      72      /// <summary>
      73      /// Liefert zu dem bergebenen Bezeichner den zugehrigen Text aus der Lok
      74      /// aktuellen Sprache zurck, die dabei aus der Einstellungsdatei des Spie
      75      /// </summary>
      76      public static string Localize (string text)
0     77      {
0     78        throw new System.NotImplementedException ();
      79      }
      80
      81      #endregion
      82    }
      83  }
```

Knot3.Core.OptionInfo

Summary

Class:	Knot3.Core.OptionInfo
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\OptionInfo.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	18
Coverable lines:	18
Total lines:	116

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\OptionInfo.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`
	32	`using System.Collections.Generic;`
	33	`using System.Diagnostics.CodeAnalysis;`
	34	`using System.Linq;`
	35	
	36	`using Microsoft.Xna.Framework;`
	37	`using Microsoft.Xna.Framework.Audio;`

```
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.Development;
      48  using Knot3.GameObjects;
      49  using Knot3.RenderEffects;
      50  using Knot3.Screens;
      51  using Knot3.Widgets;
      52
      53  #endregion
      54
      55  namespace Knot3.Core
      56  {
      57    /// <summary>
      58    /// Enthlt Informationen ber einen Eintrag in einer Einstellungsdatei.
      59    /// </summary>
      60    public class OptionInfo
      61    {
      62      #region Properties
      63
      64      /// <summary>
      65      /// Die Einstellungsdatei.
      66      /// </summary>
      67      private ConfigFile configFile;
      68
      69      /// <summary>
      70      /// Der Abschnitt der Einstellungsdatei.
      71      /// </summary>
   0  72      public string Section { get; private set; }
      73
      74      /// <summary>
      75      /// Der Name der Option.
      76      /// </summary>
   0  77      public string Name { get; private set; }
      78
      79      /// <summary>
      80      /// Der Standardwert der Option.
      81      /// </summary>
   0  82      public string DefaultValue { get; private set; }
      83
      84      /// <summary>
      85      /// Der Wert der Option.
      86      /// </summary>
      87      public virtual string Value
      88      {
   0  89        get {
   0  90          Log.Debug ("OptionInfo: ", Section, ".", Name, " => ", configFile [Sec
   0  91          return configFile [Section, Name, DefaultValue];
   0  92        }
   0  93        set {
   0  94          Log.Debug ("OptionInfo: ", Section, ".", Name, " <= ", value);
   0  95          configFile [Section, Name, DefaultValue] = value;
   0  96        }
      97      }
      98
```

```
      99      #endregion
     100
     101      #region Constructors
     102
     103      /// <summary>
     104      /// Erstellt ein neues OptionsInfo-Objekt aus den bergegebenen Werten.
     105      /// </summary>
0    106      public OptionInfo (string section, string name, string defaultValue, Confi
0    107      {
0    108        Section = section;
0    109        Name = name;
0    110        DefaultValue = defaultValue;
0    111        this.configFile = configFile != null ? configFile : Options.Default;
0    112      }
     113
     114      #endregion
     115    }
     116  }
```

Knot3.Core.Options

Summary

Class:	Knot3.Core.Options
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Options.cs
Coverage:	81.2%
Covered lines:	13
Uncovered lines:	3
Coverable lines:	16
Total lines:	99

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Options.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.Linq;
	35	
	36	using Microsoft.Xna.Framework;
	37	using Microsoft.Xna.Framework.Audio;
	38	using Microsoft.Xna.Framework.Content;
	39	using Microsoft.Xna.Framework.GamerServices;
	40	using Microsoft.Xna.Framework.Graphics;
	41	using Microsoft.Xna.Framework.Input;
	42	using Microsoft.Xna.Framework.Media;

```
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.Platform;
      49  using Knot3.RenderEffects;
      50  using Knot3.Screens;
      51  using Knot3.Utilities;
      52  using Knot3.Widgets;
      53
      54  #endregion
      55
      56  namespace Knot3.Core
      57  {
      58    /// <summary>
      59    /// Eine statische Klasse, die eine Referenz auf die zentrale Einstellungsda
      60    /// </summary>
      61    public static class Options
      62    {
      63      #region Properties
      64
      65      /// <summary>
      66      /// Die zentrale Einstellungsdatei des Spiels.
      67      /// </summary>
      68      public static ConfigFile Default
      69      {
25    70        get {
26    71          if (_default == null) {
 1    72            _default = new ConfigFile (SystemInfo.SettingsDirectory + SystemInfo
 1    73          }
25    74          return _default;
25    75        }
 0    76        set {
 0    77          _default = value;
 0    78        }
      79      }
      80
      81      private static ConfigFile _default;
      82
      83      public static ConfigFile Models
      84      {
 4    85        get {
 5    86          if (_models == null) {
 1    87            String seperatorString = SystemInfo.PathSeparator.ToString ();
 1    88            _models = new ConfigFile (SystemInfo.BaseDirectory + seperatorString
      89                                      + "Content" + seperatorString + "models.in
 1    90          }
 4    91          return _models;
 4    92        }
      93      }
      94
      95      private static ConfigFile _models;
      96
      97      #endregion
      98    }
      99  }
```

Knot3.Core.World

Summary

Class:	Knot3.Core.World
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\World.cs
Coverage:	15.9%
Covered lines:	23
Uncovered lines:	121
Coverable lines:	144
Total lines:	383

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	5	80	66.67
.ctor(...)	2	0	0
DefaultEffect(...)	1	0	0
Add(...)	2	0	0
Remove(...)	2	0	0
System.Collections.I	1	0	0
MoveNext()	5	0	0
MoveNext()	8	0	0
MoveNext()	7	0	0
MoveNext()	9	0	0
MoveNext()	9	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\World.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion

```
      27
      28  #region Using
      29
      30  using System;
      31  using System.Collections;
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.Linq;
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Data;
      47  using Knot3.GameObjects;
      48  using Knot3.RenderEffects;
      49  using Knot3.Screens;
      50  using Knot3.Utilities;
      51  using Knot3.Widgets;
      52
      53  #endregion
      54
      55  namespace Knot3.Core
      56  {
      57    /// <summary>
      58    /// Reprsentiert eine Spielwelt, in der sich 3D-Modelle befinden und gezeic
      59    /// </summary>
      60    public sealed class World : DrawableGameScreenComponent, IEnumerable<IGameOb
      61    {
      62      #region Properties
      63
      64      /// <summary>
      65      /// Die Kamera dieser Spielwelt.
      66      /// </summary>
      67      public Camera Camera
      68      {
15    69        get {
15    70          return _camera;
15    71        }
 0    72        set {
 0    73          _camera = value;
 0    74          useInternalCamera = false;
 0    75        }
      76      }
      77
      78      private Camera _camera;
 4    79      private bool useInternalCamera = true;
      80
      81      /// <summary>
      82      /// Die Liste von Spielobjekten.
      83      /// </summary>
 4    84      public HashSet<IGameObject> Objects { get; set; }
      85
      86      private IGameObject _selectedObject;
      87
```

```
      88      /// <summary>
      89      /// Das aktuell ausgewhlte Spielobjekt.
      90      /// </summary>
      91      public IGameObject SelectedObject
      92      {
0     93        get {
0     94          return _selectedObject;
0     95        }
0     96        set {
0     97          if (_selectedObject != value) {
0     98            _selectedObject = value;
0     99            SelectionChanged (_selectedObject);
0    100            Redraw = true;
0    101          }
0    102        }
      103     }
      104
      105     public float SelectedObjectDistance
      106     {
0     107       get {
0     108         if (SelectedObject != null) {
0     109           Vector3 toTarget = SelectedObject.Center () - Camera.Position;
0     110           return toTarget.Length ();
      111         }
0     112         else {
0     113           return 0;
      114         }
0     115       }
      116     }
      117
      118     /// <summary>
      119     /// Der aktuell angewendete Rendereffekt.
      120     /// </summary>
4     121     public IRenderEffect CurrentEffect { get; set; }
      122
      123     /// <summary>
      124     /// Wird ausgelst, wenn das selektierte Spielobjekt gendert wurde.
      125     /// </summary>
4     126     public Action<IGameObject> SelectionChanged = (o) => {};
      127
      128     /// <summary>
      129     /// Gibt an, ob die Spielwelt im folgenden Frame neugezeichnet wird
      130     /// oder nur der letzte Frame wiedergegeben wird.
      131     /// </summary>
0     132     public bool Redraw { get; set; }
      133
      134     /// <summary>
      135     /// Wird ausgelst, wenn die Spielwelt neu gezeichnet wird.
      136     /// </summary>
4     137     public Action OnRedraw = () => {};
      138
      139     /// <summary>
      140     /// Die Ausmae der Welt auf dem Screen.
      141     /// </summary>
4     142     public Bounds Bounds { get; private set; }
      143
      144     #endregion
      145
      146     #region Constructors
      147
      148     /// <summary>
```

```
     149      /// Erstellt eine neue Spielwelt im angegebenen Spielzustand.
     150      /// </summary>
4    151      public World (IGameScreen screen, DisplayLayer drawIndex, IRenderEffect ef
     152      : base (screen, drawIndex)
4    153      {
     154        // die Kamera fr diese Spielwelt
4    155        _camera = new Camera (screen, this);
     156
     157        // die Liste der Spielobjekte
4    158        Objects = new HashSet<IGameObject> ();
     159
4    160        CurrentEffect = effect;
     161
     162        // Die relative Standard-Position und Gre
4    163        Bounds = bounds;
     164
4    165        if (Screen.Game != null) {
0    166          Screen.Game.FullScreenChanged += () => viewportCache.Clear ();
0    167        }
4    168      }
     169
0    170      public World (IGameScreen screen, DisplayLayer drawIndex, Bounds bounds)
     171      : this (screen, drawIndex, DefaultEffect (screen), bounds)
0    172      {
0    173        RenderEffectLibrary.RenderEffectChanged += (newEffectName, time) => {
0    174          CurrentEffect = RenderEffectLibrary.CreateEffect (screen: screen, name
     175        };
0    176      }
     177
     178      private static IRenderEffect DefaultEffect (IGameScreen screen)
0    179      {
     180        // suche den eingestellten Standardeffekt heraus
0    181        string effectName = Options.Default ["video", "knot-shader", "default"];
0    182        IRenderEffect effect = RenderEffectLibrary.CreateEffect (screen: screen,
0    183        return effect;
0    184      }
     185
     186      #endregion
     187
     188      #region Methods
     189
     190      public void Add (IGameObject obj)
0    191      {
0    192        if (obj != null) {
0    193          Objects.Add (obj);
0    194          obj.World = this;
0    195        }
0    196        Redraw = true;
0    197      }
     198
     199      public void Remove (IGameObject obj)
0    200      {
0    201        if (obj != null) {
0    202          Objects.Remove (obj);
0    203        }
0    204        Redraw = true;
0    205      }
     206
     207      /// <summary>
     208      /// Ruft auf allen Spielobjekten die Update ()-Methode auf.
     209      /// </summary>
```

```
      210       [ExcludeFromCodeCoverageAttribute]
      211       public override void Update (GameTime time)
      212       {
      213         if (!Options.Default ["video", "selectiveRendering", false]) {
      214           Redraw = true;
      215         }
      216         if (Screen.PostProcessingEffect is FadeEffect) {
      217           Redraw = true;
      218         }
      219
      220         // run the update method on all game objects
      221         foreach (IGameObject obj in Objects) {
      222           obj.Update (time);
      223         }
      224       }
      225
    4 226       private Dictionary<Point,Dictionary<Vector4, Viewport>> viewportCache
      227           = new Dictionary<Point,Dictionary<Vector4, Viewport>> ();
      228
      229       public Viewport Viewport
      230       {
    6 231         get {
      232           // when we have a graphics device
    6 233           if (Screen.Device != null) {
    0 234             PresentationParameters pp = Screen.Device.PresentationParameters;
    0 235             Point resolution = new Point (pp.BackBufferWidth, pp.BackBufferHeigh
    0 236             Vector4 key = Bounds.Vector4;
    0 237             if (!viewportCache.ContainsKey (resolution)) {
    0 238               viewportCache [resolution] = new Dictionary<Vector4, Viewport> ();
    0 239             }
    0 240             if (!viewportCache [resolution].ContainsKey (key)) {
    0 241               Rectangle subScreen = Bounds.Rectangle;
    0 242               viewportCache [resolution] [key] = new Viewport (subScreen.X, subS
      243                 MinDepth = 0,
      244                 MaxDepth = 1
      245               };
    0 246             }
    0 247             return viewportCache [resolution] [key];
      248           }
      249           // for unit tests
    6 250           else {
    6 251             return Screen.Viewport;
      252           }
    6 253         }
      254       }
      255
      256       /// <summary>
      257       /// Ruft auf allen Spielobjekten die Draw ()-Methode auf.
      258       /// </summary>
      259       [ExcludeFromCodeCoverageAttribute]
      260       public override void Draw (GameTime time)
      261       {
      262         if (Redraw) {
      263           OnRedraw ();
      264           Redraw = false;
      265
      266           //Screen.BackgroundColor = CurrentEffect is CelShadingEffect ? Color.C
      267
      268           // begin the knot render effect
      269           CurrentEffect.Begin (time);
      270
```

```
      271          foreach (IGameObject obj in Objects) {
      272            obj.World = this;
      273            obj.Draw (time);
      274          }
      275
      276          // end of the knot render effect
      277          CurrentEffect.End (time);
      278        }
      279        else {
      280          CurrentEffect.DrawLastFrame (time);
      281        }
      282      }
      283
      284      /// <summary>
      285      /// Liefert einen Enumerator ber die Spielobjekte dieser Spielwelt.
      286      /// [returntype=IEnumerator<IGameObject>]
      287      /// </summary>
      288      public IEnumerator<IGameObject> GetEnumerator ()
    0 289      {
    0 290        foreach (IGameObject obj in flat (Objects)) {
    0 291          yield return obj;
    0 292        }
    0 293      }
      294
      295      private IEnumerable<IGameObject> flat (IEnumerable<IGameObject> enumerable
    0 296      {
    0 297        foreach (IGameObject obj in enumerable) {
    0 298          if (obj is IEnumerable<IGameObject>) {
    0 299            foreach (IGameObject subobj in flat (obj as IEnumerable<IGameObject>
    0 300              yield return subobj;
    0 301            }
    0 302          }
    0 303          else {
    0 304            yield return obj;
    0 305          }
    0 306        }
    0 307      }
      308      // Explicit interface implementation for nongeneric interface
      309      IEnumerator IEnumerable.GetEnumerator ()
    0 310      {
    0 311        return GetEnumerator (); // Just return the generic version
    0 312      }
      313
      314      public override IEnumerable<IGameScreenComponent> SubComponents (GameTime
    0 315      {
    0 316        foreach (DrawableGameScreenComponent component in base.SubComponents (ti
    0 317          yield return component;
    0 318        }
    0 319        if (useInternalCamera) {
    0 320          yield return Camera;
    0 321        }
    0 322      }
      323
      324      /// <summary>
      325      /// Gibt einen Iterator ber alle Spielobjekte zurck, der so sortiert ist
      326      /// Spielobjekte, die der angegebenen 2D-Position am nchsten sind, am Anf
      327      /// Dazu wird die 2D-Position in eine 3D-Position konvertiert.
      328      /// </summary>
      329      public IEnumerable<IGameObject> FindNearestObjects (Vector2 nearTo)
    0 330      {
    0 331        Dictionary<float, IGameObject> distances = new Dictionary<float, IGameOb
```

```
0   332        foreach (IGameObject obj in this) {
0   333          if (obj.Info.IsSelectable) {
    334            // Berechne aus der angegebenen 2D-Position eine 3D-Position
0   335            Vector3 position3D = Camera.To3D (
    336                                    position: nearTo,
    337                                    nearTo: obj.Center ()
    338                                );
    339            // Berechne die Distanz zwischen 3D-Mausposition und dem Spielobjekt
0   340            float distance = Math.Abs ((position3D - obj.Center ()).Length ());
0   341            distances [distance] = obj;
0   342          }
0   343        }
0   344        if (distances.Count > 0) {
0   345          IEnumerable<float> sorted = distances.Keys.OrderBy (k => k);
0   346          foreach (float where in sorted) {
0   347            yield return distances [where];
    348            // Log.Debug ("where=", where, " = ", distances [where].Center ());
0   349          }
0   350        }
0   351        else {
0   352          yield break;
    353        }
0   354      }
    355
    356      /// <summary>
    357      /// Gibt einen Iterator ber alle Spielobjekte zurck, der so sortiert ist
    358      /// Spielobjekte, die der angegebenen 3D-Position am nchsten sind, am Anf
    359      /// </summary>
    360      public IEnumerable<IGameObject> FindNearestObjects (Vector3 nearTo)
0   361      {
0   362        Dictionary<float, IGameObject> distances = new Dictionary<float, IGameOb
0   363        foreach (IGameObject obj in this) {
0   364          if (obj.Info.IsSelectable) {
    365            // Berechne die Distanz zwischen 3D-Mausposition und dem Spielobjekt
0   366            float distance = Math.Abs ((nearTo - obj.Center ()).Length ());
0   367            distances [distance] = obj;
0   368          }
0   369        }
0   370        if (distances.Count > 0) {
0   371          IEnumerable<float> sorted = distances.Keys.OrderBy (k => k);
0   372          foreach (float where in sorted) {
0   373            yield return distances [where];
0   374          }
0   375        }
0   376        else {
0   377          yield break;
    378        }
0   379      }
    380
    381      #endregion
    382    }
    383  }
```

Knot3.Data.Challenge

Summary

Class:	Knot3.Data.Challenge
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Challenge.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	20
Coverable lines:	20
Total lines:	136

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
AddToHighscore(...)	1	0	0
Save()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Challenge.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.Linq;

```
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Core;
      47  using Knot3.GameObjects;
      48  using Knot3.Input;
      49  using Knot3.RenderEffects;
      50  using Knot3.Screens;
      51  using Knot3.Widgets;
      52
      53  #endregion
      54
      55  namespace Knot3.Data
      56  {
      57    /// <summary>
      58    /// Ein Objekt dieser Klasse reprsentiert eine Challenge.
      59    /// </summary>
      60    public sealed class Challenge
      61    {
      62      #region Properties
      63
      64      /// <summary>
      65      /// Der Ausgangsknoten, den der Spieler in den Referenzknoten transformier
      66      /// </summary>
  0   67      public Knot Start { get; private set; }
      68
      69      /// <summary>
      70      /// Der Referenzknoten, in den der Spieler den Ausgangsknoten transformier
      71      /// </summary>
  0   72      public Knot Target { get; private set; }
      73
      74      /// <summary>
      75      /// Eine sortierte Bestenliste.
      76      /// </summary>
  0   77      private SortedList<int, string> highscore { get; set; }
      78
      79      /// <summary>
      80      /// Ein ffentlicher Enumerator, der die Bestenliste unabhngig von der da
      81      /// </summary>
  0   82      public IEnumerable<KeyValuePair<string, int>> Highscore { get { return Met
      83
      84      /// <summary>
      85      /// Die Metadaten der Challenge.
      86      /// </summary>
  0   87      public ChallengeMetaData MetaData { get; private set; }
      88
      89      /// <summary>
      90      /// Der Name der Challenge.
      91      /// </summary>
      92      public string Name
      93      {
  0   94        get { return MetaData.Name; }
  0   95        set { MetaData.Name = value; }
```

```
        96      }
        97
        98      #endregion
        99
       100      #region Constructors
       101
       102      /// <summary>
       103      /// Erstellt ein Challenge-Objekt aus einem gegebenen Challenge-Metadaten-
       104      /// Erstellt ein Challenge-Objekt aus einer gegebenen Challenge-Datei.
       105      /// </summary>
  0    106      public Challenge (ChallengeMetaData meta, Knot start, Knot target)
  0    107      {
  0    108        MetaData = meta;
  0    109        Start = start;
  0    110        Target = target;
  0    111      }
       112
       113      #endregion
       114
       115      #region Methods
       116
       117      /// <summary>
       118      /// Fgt eine neue Bestzeit eines bestimmten Spielers in die Bestenliste e
       119      /// </summary>
       120      public void AddToHighscore (string name, int time)
  0    121      {
  0    122        MetaData.AddToHighscore (name, time);
  0    123        Save ();
  0    124      }
       125
       126      /// <summary>
       127      /// Speichert die Challenge.
       128      /// </summary>
       129      public void Save ()
  0    130      {
  0    131        MetaData.Format.Save (this);
  0    132      }
       133
       134      #endregion
       135    }
       136  }
```

Knot3.Data.ChallengeFileIO

Summary

Class:	Knot3.Data.ChallengeFileIO
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeFileIO.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	95
Coverable lines:	95
Total lines:	262

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	1	0	0
Save(...)	2	0	0
Load(...)	9	0	0
LoadMetaData(...)	11	0	0
MoveNext()	8	0	0
MoveNext()	5	0	0
MoveNext()	7	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeFileIO.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                          Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
```

```
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.IO;
        35  using System.Linq;
        36  using System.Text;
        37
        38  using Microsoft.Xna.Framework;
        39  using Microsoft.Xna.Framework.Audio;
        40  using Microsoft.Xna.Framework.Content;
        41  using Microsoft.Xna.Framework.GamerServices;
        42  using Microsoft.Xna.Framework.Graphics;
        43  using Microsoft.Xna.Framework.Input;
        44  using Microsoft.Xna.Framework.Media;
        45  using Microsoft.Xna.Framework.Net;
        46  using Microsoft.Xna.Framework.Storage;
        47
        48  using Ionic.Zip;
        49
        50  using Knot3.Core;
        51  using Knot3.Development;
        52  using Knot3.GameObjects;
        53  using Knot3.Input;
        54  using Knot3.RenderEffects;
        55  using Knot3.Screens;
        56  using Knot3.Widgets;
        57
        58  #endregion
        59
        60  namespace Knot3.Data
        61  {
        62    /// <summary>
        63    /// Implementiert das Speicherformat fr Challenges.
        64    /// </summary>
        65    public sealed class ChallengeFileIO : IChallengeIO
        66    {
        67      #region Properties
        68
        69      /// <summary>
        70      /// Die fr eine Knoten-Datei gltigen Dateiendungen.
        71      /// </summary>
    0   72      public IEnumerable<string> FileExtensions
        73      {
    0   74        get {
    0   75          yield return ".challenge";
    0   76          yield return ".chl";
    0   77          yield return ".chn";
    0   78          yield return ".chg";
    0   79          yield return ".chlng";
    0   80        }
        81      }
        82
        83      #endregion
        84
        85      #region Constructors
        86
        87      /// <summary>
        88      /// Erstellt ein ChallengeFileIO-Objekt.
        89      /// </summary>
    0   90      public ChallengeFileIO ()
    0   91      {
```

```
0    92      }
     93
     94      #endregion
     95
     96      #region Methods
     97
     98      /// <summary>
     99      /// Speichert eine Challenge in dem Dateinamen, der in dem Challenge-Objek
    100      /// </summary>
    101      public void Save (Challenge challenge)
0   102      {
0   103        using (ZipFile zip = new ZipFile ()) {
    104          // Namen
0   105          zip.AddEntry ("name.txt", challenge.Name);
    106          // Startknoten
0   107          KnotStringIO parser = new KnotStringIO (challenge.Start);
0   108          zip.AddEntry ("start.knot", parser.Content);
    109          // Zielknoten
0   110          parser = new KnotStringIO (challenge.Target);
0   111          zip.AddEntry ("target.knot", parser.Content);
    112          // Highscore
0   113          zip.AddEntry ("highscore.txt", string.Join ("\n", printHighscore (chal
    114          // ZIP-Datei speichern
0   115          zip.Save (challenge.MetaData.Filename);
0   116        }
0   117      }
    118
    119      /// <summary>
    120      /// Ldt eine Challenge aus einer angegebenen Datei.
    121      /// </summary>
    122      public Challenge Load (string filename)
0   123      {
0   124        ChallengeMetaData meta = LoadMetaData (filename: filename);
0   125        Knot start = null;
0   126        Knot target = null;
    127
0   128        using (ZipFile zip = ZipFile.Read (filename)) {
0   129          foreach (ZipEntry entry in zip) {
0   130            string content = entry.ReadContent ();
    131
    132            // fr die Datei mit dem Startknoten
0   133            if (entry.FileName.ToLower ().Contains ("start")) {
0   134              KnotStringIO parser = new KnotStringIO (content: content);
0   135              start = new Knot (
0   136                  new KnotMetaData (parser.Name, () => parser.CountEdges, null,
    137                  parser.Edges
    138              );
0   139            }
    140
    141            // fr die Datei mit dem Zielknoten
0   142            else if (entry.FileName.ToLower ().Contains ("target")) {
0   143              KnotStringIO parser = new KnotStringIO (content: content);
0   144              target = new Knot (
0   145                  new KnotMetaData (parser.Name, () => parser.CountEdges, null,
    146                  parser.Edges
    147              );
0   148            }
0   149          }
0   150        }
    151
0   152        if (meta != null && start != null && target != null) {
```

```
0   153          return new Challenge (meta, start, target);
    154        }
0   155        else {
0   156          throw new IOException (
    157              "Error! Invalid challenge file: " + filename
    158              + " (meta=" + meta + ",start=" + start + ",target=" + target + ")"
    159          );
    160        }
0   161      }
    162
    163      /// <summary>
    164      /// Ldt die Metadaten einer Challenge aus einer angegebenen Datei.
    165      /// </summary>
    166      public ChallengeMetaData LoadMetaData (string filename)
0   167      {
0   168        string name = null;
0   169        KnotMetaData start = null;
0   170        KnotMetaData target = null;
0   171        IEnumerable<KeyValuePair<string, int>> highscore = null;
0   172        using (ZipFile zip = ZipFile.Read (filename)) {
0   173          foreach (ZipEntry entry in zip) {
0   174            string content = entry.ReadContent ();
    175
    176            // fr die Datei mit dem Startknoten
0   177            if (entry.FileName.ToLower ().Contains ("start")) {
0   178              KnotStringIO parser = new KnotStringIO (content: content);
0   179              start = new KnotMetaData (parser.Name, () => parser.CountEdges, nu
0   180            }
    181
    182            // fr die Datei mit dem Zielknoten
0   183            else if (entry.FileName.ToLower ().Contains ("target")) {
0   184              KnotStringIO parser = new KnotStringIO (content: content);
0   185              target = new KnotMetaData (parser.Name, () => parser.CountEdges, n
0   186            }
    187
    188            // fr die Datei mit dem Namen
0   189            else if (entry.FileName.ToLower ().Contains ("name")) {
0   190              name = content.Trim ();
0   191            }
    192
    193            // fr die Datei mit den Highscores
0   194            else if (entry.FileName.ToLower ().Contains ("highscore")) {
0   195              highscore = parseHighscore (content.Split (new char[] {'\r','\n'},
0   196            }
0   197          }
0   198        }
0   199        if (name != null && start != null && target != null) {
0   200          Log.Debug ("Load challenge file: ", filename, " (name=", name, ",start
0   201          return new ChallengeMetaData (
    202                      name: name,
    203                      start: start,
    204                      target: target,
    205                      filename: filename,
    206                      format: this,
    207                      highscore: highscore
    208                  );
    209        }
0   210        else {
0   211          throw new IOException (
    212              "Error! Invalid challenge file: " + filename
    213              + " (name=" + name + ",start=" + start + ",target=" + target + ",h
```

```
      214          );
      215        }
0     216      }
      217
      218      IEnumerable<string> printHighscore (IEnumerable<KeyValuePair<string, int>>
0     219      {
0     220        foreach (KeyValuePair<string, int> entry in highscore) {
0     221          Log.Debug (
      222              "Save Highscore: "
      223              + entry.Value.ToString ()
      224              + ":"
      225              + entry.Key.ToString ()
      226          );
      227
0     228          yield return entry.Value + ":" + entry.Key;
0     229        }
0     230      }
      231
      232      IEnumerable<KeyValuePair<string, int>> parseHighscore (IEnumerable<string>
0     233      {
0     234        foreach (string line in highscore) {
0     235          Log.Debug ("Load Highscore: ",line);
0     236          if (line.Contains (":")) {
0     237            string[] entry = line.Split (new char[] {':'}, 2, StringSplitOptions
0     238            string name = entry [1].Trim ();
      239            int time;
0     240            if (Int32.TryParse (entry [0], out time)) {
0     241              Log.Debug ("=> ", name, ":", time);
0     242              yield return new KeyValuePair<string, int> (name, time);
0     243            }
0     244          }
0     245        }
0     246      }
      247
      248      #endregion
      249    }
      250
      251    static class ZipHelper
      252    {
      253      public static string ReadContent (this ZipEntry entry)
      254      {
      255        MemoryStream memory = new MemoryStream ();
      256        entry.Extract (memory);
      257        memory.Position = 0;
      258        var sr = new StreamReader (memory);
      259        return sr.ReadToEnd ();
      260      }
      261    }
      262  }
```

Knot3.Data.ChallengeMetaData

Summary

Class:	Knot3.Data.ChallengeMetaData
Assembly:	Knot3
File(s):	:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeMetaData.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	82
Coverable lines:	82
Total lines:	242

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	6	0	0
AddToHighscore(...)	2	0	0
formatTime(...)	1	0	0
Equals(...)	2	0	0
Equals(...)	2	0	0
op_Equality(...)	4	0	0
op_Inequality(...)	1	0	0

File(s)

:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeMetaData.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                          Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
```

```
      31  using System.Collections;
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.Linq;
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Core;
      47  using Knot3.Development;
      48  using Knot3.GameObjects;
      49  using Knot3.Input;
      50  using Knot3.Platform;
      51  using Knot3.RenderEffects;
      52  using Knot3.Screens;
      53  using Knot3.Utilities;
      54  using Knot3.Widgets;
      55
      56  #endregion
      57
      58  namespace Knot3.Data
      59  {
      60    /// <summary>
      61    /// Enthlt Metadaten zu einer Challenge.
      62    /// </summary>
      63    public class ChallengeMetaData
      64    {
      65      #region Properties
      66
      67      /// <summary>
      68      /// Der Name der Challenge.
      69      /// </summary>
      70      public string Name
      71      {
0     72        get {
0     73          return name;
0     74        }
0     75        set {
0     76          name = value;
0     77          if (Format == null) {
0     78            Format = new ChallengeFileIO ();
0     79          }
      80          string extension;
0     81          if (Format.FileExtensions.Any ()) {
0     82            extension = Format.FileExtensions.ElementAt (0);
0     83          }
0     84          else {
0     85            throw new ArgumentException ("Every implementation of IChallengeIO m
      86          }
0     87          Filename = SystemInfo.SavegameDirectory + SystemInfo.PathSeparator.ToS
0     88        }
      89      }
      90
      91      private string name;
```

```
        92
        93      /// <summary>
        94      /// Der Ausgangsknoten, den der Spieler in den Referenzknoten transformier
        95      /// </summary>
0       96      public KnotMetaData Start { get; private set; }
        97
        98      /// <summary>
        99      /// Der Referenzknoten, in den der Spieler den Ausgangsknoten transformier
       100      /// </summary>
0      101      public KnotMetaData Target { get; private set; }
       102
       103      /// <summary>
       104      /// Das Format, aus dem die Metadaten der Challenge gelesen wurden oder nu
       105      /// </summary>
0      106      public IChallengeIO Format { get; private set; }
       107
       108      /// <summary>
       109      /// Der Dateiname, aus dem die Metadaten der Challenge gelesen wurden oder
       110      /// </summary>
0      111      public string Filename { get; private set; }
       112
       113      /// <summary>
       114      /// Ein ffentlicher Enumerator, der die Bestenliste unabhngig von der da
       115      /// </summary>
0      116      public IEnumerable<KeyValuePair<string, int>> Highscore { get { return hig
       117
       118      private List<KeyValuePair<string, int>> highscore;
       119
       120      public float AvgTime
       121      {
0      122        get {
0      123          if (   highscore != null
0      124                  && highscore.Any ()) {
0      125            float amount =0;
0      126            foreach (KeyValuePair<string, int> entry in highscore) {
0      127              amount += (float)entry.Value;
0      128            }
0      129            return amount/((float)highscore.Count);
       130          }
0      131          return 0f;
0      132        }
       133
0      134        private set {}
       135      }
       136
       137      public string FormatedAvgTime
       138      {
0      139        get {
0      140          float time = AvgTime;
0      141          Log.Debug (time);
0      142          if (time != 0f) {
0      143            return formatTime (time);
       144          }
0      145          return "Not yet set.";
0      146        }
0      147        private set {
0      148        }
       149      }
       150
       151      #endregion
       152
```

```
      153     #region Constructors
      154
      155     /// <summary>
      156     /// Erstellt ein Challenge-Metadaten-Objekt mit einem gegebenen Namen und
      157     /// </summary>
   0  158     public ChallengeMetaData (string name, KnotMetaData start, KnotMetaData ta
      159                               string filename, IChallengeIO format,
      160                               IEnumerable<KeyValuePair<string, int>> highscore
   0  161     {
   0  162       Name = name;
   0  163       Start = start;
   0  164       Target = target;
   0  165       Format = format ?? Format;
   0  166       Filename = filename ?? Filename;
      167
   0  168       this.highscore = new List<KeyValuePair<string, int>> ();
   0  169       if (highscore != null) {
   0  170         foreach (KeyValuePair<string, int> entry in highscore) {
   0  171           this.highscore.Add (entry);
   0  172         }
   0  173       }
   0  174     }
      175
      176     #endregion
      177
      178     #region Methods
      179
      180     /// <summary>
      181     /// Fgt eine neue Bestzeit eines bestimmten Spielers in die Bestenliste e
      182     /// </summary>
      183     public void AddToHighscore (string name, int time)
   0  184     {
   0  185       KeyValuePair<string, int> entry = new KeyValuePair<string, int> (name, t
   0  186       if (!highscore.Contains (entry)) {
   0  187         highscore.Add (entry);
   0  188       }
   0  189     }
      190
      191     public static string formatTime (float secs)
   0  192     {
   0  193       Log.Debug (secs);
   0  194       TimeSpan t = TimeSpan.FromSeconds ( secs );
      195
   0  196       string answer = string.Format ("{0:D2}h:{1:D2}m:{2:D2}s",
      197                                      t.Hours,
      198                                      t.Minutes,
      199                                      t.Seconds);
   0  200       return answer;
   0  201     }
      202
      203     public bool Equals (ChallengeMetaData other)
   0  204     {
   0  205       return other != null && name == other.name;
   0  206     }
      207
      208     public override bool Equals (object other)
   0  209     {
   0  210       return other != null && Equals (other as ChallengeMetaData);
   0  211     }
      212
      213     [ExcludeFromCodeCoverageAttribute]
```

```
      214      public override int GetHashCode ()
      215      {
      216        return (name ?? String.Empty).GetHashCode ();
      217      }
      218
      219      public static bool operator == (ChallengeMetaData a, ChallengeMetaData b)
    0 220      {
      221        // If both are null, or both are same instance, return true.
    0 222        if (System.Object.ReferenceEquals (a, b)) {
    0 223          return true;
      224        }
      225
      226        // If one is null, but not both, return false.
    0 227        if (((object)a == null) || ((object)b == null)) {
    0 228          return false;
      229        }
      230
      231        // Return true if the fields match:
    0 232        return a.Equals (b);
    0 233      }
      234
      235      public static bool operator != (ChallengeMetaData a, ChallengeMetaData b)
    0 236      {
    0 237        return !(a == b);
    0 238      }
      239
      240      #endregion
      241    }
      242  }
```

Knot3.Data.CircleEntry‘1

Summary

Class:	Knot3.Data.CircleEntry‘1
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\CircleEntry.cs
Coverage:	92.3%
Covered lines:	205
Uncovered lines:	17
Coverable lines:	222
Total lines:	402

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor()	1	100	100
.ctor(...)	4	100	85.71
InsertBefore(...)	1	100	100
InsertAfter(...)	1	100	100
Remove()	1	100	100
Contains(...)	1	100	100
Contains(...)	1	100	100
Contains(...)	1	100	100
Contains(...)	1	100	100
Find(...)	1	100	100
IndexOf(...)	1	100	100
IndexOf(...)	3	92.31	80
System.Collections.I	1	0	0
op_Addition(...)	3	100	100
op_Subtraction(...)	1	100	100
op_Increment(...)	1	100	100
op_Decrement(...)	1	100	100
op_Implicit(...)	1	100	100
Contains(...)	1	100	100
Remove(...)	2	100	100
RemoveAt(...)	1	100	100
Insert(...)	1	0	0
Add(...)	2	100	100
Clear()	1	100	100
CopyTo(...)	3	100	80
MoveNext()	6	100	87.5
MoveNext()	6	100	87.5
MoveNext()	6	81.82	75
MoveNext()	5	0	0
MoveNext()	5	100	83.33
MoveNext()	5	100	83.33

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\CircleEntry.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	`* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`

```
 5   *                          Gerd Augsburg, Christina Erler, Daniel Warzel
 6   *
 7   * Permission is hereby granted, free of charge, to any person obtaining a cop
 8   * of this software and associated documentation files (the "Software"), to de
 9   * in the Software without restriction, including without limitation the right
10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11   * copies of the Software, and to permit persons to whom the Software is
12   * furnished to do so, subject to the following conditions:
13   *
14   * The above copyright notice and this permission notice shall be included in
15   * copies or substantial portions of the Software.
16   *
17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
23   * SOFTWARE.
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37  using Microsoft.Xna.Framework.Audio;
38  using Microsoft.Xna.Framework.Content;
39  using Microsoft.Xna.Framework.GamerServices;
40  using Microsoft.Xna.Framework.Graphics;
41  using Microsoft.Xna.Framework.Input;
42  using Microsoft.Xna.Framework.Media;
43  using Microsoft.Xna.Framework.Net;
44  using Microsoft.Xna.Framework.Storage;
45
46  using Knot3.Core;
47  using Knot3.GameObjects;
48  using Knot3.Input;
49  using Knot3.RenderEffects;
50  using Knot3.Screens;
51  using Knot3.Widgets;
52
53  #endregion
54
55  namespace Knot3.Data
56  {
57    /// <summary>
58    /// Eine doppelt verkettete Liste.
59    /// </summary>
60    public class CircleEntry<T> : IEnumerable<T>, ICollection<T>, IList<T>
61    {
62      public T Value { get; set; }
63
64      public CircleEntry<T> Next { get; set; }
65
```

```
        66      public CircleEntry<T> Previous { get; set; }
        67
1095    68      public CircleEntry (T value)
1095    69      {
1095    70        Value = value;
1095    71        Previous = this;
1095    72        Next = this;
1095    73      }
        74
   7    75      private CircleEntry ()
   7    76      {
   7    77        Previous = this;
   7    78        Next = this;
   7    79      }
        80
  41    81      public CircleEntry (IEnumerable<T> list)
  41    82      {
  41    83        bool first = true;
  41    84        CircleEntry<T> inserted = this;
3384    85        foreach (T obj in list) {
1128    86          if (first) {
  41    87            Value = obj;
  41    88            Previous = this;
  41    89            Next = this;
  41    90          }
1046    91          else {
1046    92            inserted = inserted.InsertAfter (obj);
1046    93          }
1087    94          first = false;
1087    95        }
  41    96      }
        97
        98      public static CircleEntry<T> Empty
        99      {
   7   100        get {
   7   101          return new CircleEntry<T> ();
   7   102        }
       103      }
       104
       105      public CircleEntry<T> InsertBefore (T obj)
  47   106      {
  47   107        CircleEntry<T> insert = new CircleEntry<T> (obj);
  47   108        insert.Previous = this.Previous;
  47   109        insert.Next = this;
  47   110        this.Previous.Next = insert;
  47   111        this.Previous = insert;
  47   112        return insert;
  47   113      }
       114
       115      public CircleEntry<T> InsertAfter (T obj)
1048   116      {
       117        //Log.Debug (this, ".InsertAfter (", obj, ")");
1048   118        CircleEntry<T> insert = new CircleEntry<T> (obj);
1048   119        insert.Next = this.Next;
1048   120        insert.Previous = this;
1048   121        this.Next.Previous = insert;
1048   122        this.Next = insert;
1048   123        return insert;
1048   124      }
       125
       126      public void Remove ()
```

```
   115   127      {
   115   128        Previous.Next = Next;
   115   129        Next.Previous = Previous;
   115   130        Previous = null;
   115   131        Next = null;
   115   132      }
         133
         134      private bool IsEmpty
         135      {
    29   136        get {
    29   137          return (Next == this || Next == null) && (Previous == this || Previous
    29   138        }
         139      }
         140
         141      public int Count
         142      {
    27   143        get {
    27   144          if (IsEmpty) {
     0   145            return 0;
         146          }
    27   147          else {
    27   148            CircleEntry<T> current = this;
    27   149            int count = 0;
   244   150            do {
   244   151              ++count;
   244   152              current = current.Next;
   244   153            }
   244   154            while (current != this);
    27   155            return count;
         156          }
    27   157        }
         158      }
         159
         160      public bool Contains (T obj, out IEnumerable<CircleEntry<T>> item)
   199   161      {
   199   162        item = Find (obj);
   199   163        return item.Count () > 0;
   199   164      }
         165
         166      public bool Contains (Func<T, bool> func, out IEnumerable<CircleEntry<T>>
   199   167      {
   199   168        item = Find (func);
   199   169        return item.Count () > 0;
   199   170      }
         171
         172      public bool Contains (T obj, out CircleEntry<T> item)
   301   173      {
   301   174        item = Find (obj).ElementAtOrDefault (0);
   301   175        return item != null;
   301   176      }
         177
         178      public bool Contains (Func<T, bool> func, out CircleEntry<T> item)
   100   179      {
   100   180        item = Find (func).ElementAtOrDefault (0);
   100   181        return item != null;
   100   182      }
         183
         184      public IEnumerable<CircleEntry<T>> Find (T obj)
   707   185      {
 27279   186        return Find ((t) => t.Equals (obj));
   707   187      }
```

```
                188
                189       public IEnumerable<CircleEntry<T>> Find (Func<T, bool> func)
       1909     190       {
       1909     191         CircleEntry<T> current = this;
      52516     192         do {
      53929     193           if (func (current.Value)) {
       1413     194             yield return current;
        600     195           }
      51703     196           current = current.Next;
      51703     197         }
      51703     198         while (current != this);
       1096     199         yield break;
                200       }
                201
                202       public int IndexOf (T obj)
        100     203       {
       5150     204         return IndexOf ((t) => t.Equals (obj));
        100     205       }
                206
                207       public int IndexOf (Func<T, bool> func)
        200     208       {
        200     209         int i = 0;
        200     210         CircleEntry<T> current = this;
      10100     211         do {
      10300     212           if (func (current.Value)) {
        200     213             return i;
                214           }
       9900     215           current = current.Next;
       9900     216           ++ i;
       9900     217         }
       9900     218         while (current != this);
          0     219         return -1;
        200     220       }
                221
                222       public IEnumerable<T> RangeTo (CircleEntry<T> other)
          4     223       {
          4     224         CircleEntry<T> current = this;
         16     225         do {
         16     226           yield return current.Value;
         16     227           current = current.Next;
         16     228         }
         16     229         while (current != other.Next && current != this);
          4     230       }
                231
                232       public IEnumerable<T> WayTo (T other)
          1     233       {
          1     234         CircleEntry<T> current = this;
          3     235         while (!current.Value.Equals (other)) {
          1     236           yield return current.Value;
          1     237           current = current.Next;
          1     238           if (current == this) {
          0     239             break;
                240           }
          1     241         }
          1     242       }
                243
                244       public IEnumerable<Tuple<T,T>> Pairs
                245       {
          0     246         get {
          0     247           CircleEntry<T> current = this;
          0     248           do {
```

```
     0   249            yield return Tuple.Create (current.Value, current.Next.Value);
     0   250            current = current.Next;
     0   251          }
     0   252          while (current != this);
     0   253        }
         254      }
         255
         256      public IEnumerable<Tuple<T,T,T>> Triples
         257      {
    12   258        get {
    12   259          CircleEntry<T> current = this;
    52   260          do {
    52   261            yield return Tuple.Create (current.Previous.Value, current.Value, cu
    50   262            current = current.Next;
    50   263          }
    50   264          while (current != this);
    10   265        }
         266      }
         267
         268      public IEnumerator<T> GetEnumerator ()
    91   269      {
    91   270        CircleEntry<T> current = this;
   826   271        do {
         272          //Log.Debug (this, " => ", current.Content);
   826   273          yield return current.Value;
   818   274          current = current.Next;
   818   275        }
   818   276        while (current != this);
    83   277      }
         278
         279      // explicit interface implementation for nongeneric interface
         280      IEnumerator IEnumerable.GetEnumerator ()
     0   281      {
     0   282        return GetEnumerator (); // just return the generic version
     0   283      }
         284
         285      [ExcludeFromCodeCoverageAttribute]
         286      public override string ToString ()
         287      {
         288        if (IsEmpty) {
         289          return "CircleEntry (" + Value.ToString () + ")";
         290        }
         291        else {
         292          return "CircleEntry.Empty";
         293        }
         294      }
         295
         296      public static CircleEntry<T> operator + (CircleEntry<T> circle, int i)
   579   297      {
   579   298        CircleEntry<T> next = circle;
 40549   299        while (i > 0) {
 19985   300          next = next.Next;
 19985   301          i--;
 19985   302        }
   989   303        while (i < 0) {
   205   304          next = next.Previous;
   205   305          i++;
   205   306        }
   579   307        return next;
   579   308      }
         309
```

```
        310       public T this [int index]
        311       {
   238  312         get {
   238  313           return (this + index).Value;
   238  314         }
   100  315         set {
   100  316           (this + index).Value = value;
   100  317         }
        318       }
        319
        320       public static CircleEntry<T> operator - (CircleEntry<T> circle, int i)
    30  321       {
    30  322         return circle + (-i);
    30  323       }
        324
        325       public static CircleEntry<T> operator ++ (CircleEntry<T> circle)
   189  326       {
   189  327         return circle.Next;
   189  328       }
        329
        330       public static CircleEntry<T> operator -- (CircleEntry<T> circle)
    11  331       {
    11  332         return circle.Previous;
    11  333       }
        334
        335       public static implicit operator T (CircleEntry<T> circle)
   700  336       {
   700  337         return circle.Value;
   700  338       }
        339
     3  340       public bool IsReadOnly { get { return false; } }
        341
        342       public bool Contains (T obj)
   102  343       {
   102  344         CircleEntry<T> item = Find (obj).ElementAtOrDefault (0);
   102  345         return item != null;
   102  346       }
        347
        348       public bool Remove (T value)
   198  349       {
        350         CircleEntry<T> item;
   297  351         if (Contains (value, out item)) {
    99  352           item.Remove ();
    99  353           return true;
        354         }
    99  355         else {
    99  356           return false;
        357         }
   198  358       }
        359
        360       public void RemoveAt (int i)
     1  361       {
     1  362         (this + i).Remove ();
     1  363       }
        364
        365       public void Insert (int i, T value)
     0  366       {
     0  367         (this + i).InsertBefore (value);
     0  368       }
        369
        370       public void Add (T value)
```

```
 50  371      {
 56  372        if (Value == null) {
  6  373          Value = value;
  6  374        }
 44  375        else {
 44  376          InsertBefore (value);
 44  377        }
 50  378      }
     379
     380      public void Clear ()
  1  381      {
  1  382        Remove ();
  1  383        Next = Previous = this;
  1  384      }
     385
     386      public void CopyTo (T[] array, int start)
  1  387      {
303  388        foreach (T value in this) {
100  389          array.SetValue (value, start);
100  390          ++start;
100  391        }
  1  392      }
     393    }
     394
     395    public static class CircleExtensions
     396    {
     397      public static CircleEntry<T> ToCircle<T> (this IEnumerable<T> enumerable)
     398      {
     399        return new CircleEntry<T> (enumerable);
     400      }
     401    }
     402  }
```

Knot3.Data.CircleExtensions

Summary

Class:	Knot3.Data.CircleExtensions
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\CircleEntry.cs
Coverage:	100%
Covered lines:	3
Uncovered lines:	0
Coverable lines:	3
Total lines:	402

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
ToCircle(...)	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\CircleEntry.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`
	32	`using System.Collections.Generic;`
	33	`using System.Diagnostics.CodeAnalysis;`
	34	`using System.Linq;`
	35	
	36	`using Microsoft.Xna.Framework;`
	37	`using Microsoft.Xna.Framework.Audio;`

```
38  using Microsoft.Xna.Framework.Content;
39  using Microsoft.Xna.Framework.GamerServices;
40  using Microsoft.Xna.Framework.Graphics;
41  using Microsoft.Xna.Framework.Input;
42  using Microsoft.Xna.Framework.Media;
43  using Microsoft.Xna.Framework.Net;
44  using Microsoft.Xna.Framework.Storage;
45
46  using Knot3.Core;
47  using Knot3.GameObjects;
48  using Knot3.Input;
49  using Knot3.RenderEffects;
50  using Knot3.Screens;
51  using Knot3.Widgets;
52
53  #endregion
54
55  namespace Knot3.Data
56  {
57    /// <summary>
58    /// Eine doppelt verkettete Liste.
59    /// </summary>
60    public class CircleEntry<T> : IEnumerable<T>, ICollection<T>, IList<T>
61    {
62      public T Value { get; set; }
63
64      public CircleEntry<T> Next { get; set; }
65
66      public CircleEntry<T> Previous { get; set; }
67
68      public CircleEntry (T value)
69      {
70        Value = value;
71        Previous = this;
72        Next = this;
73      }
74
75      private CircleEntry ()
76      {
77        Previous = this;
78        Next = this;
79      }
80
81      public CircleEntry (IEnumerable<T> list)
82      {
83        bool first = true;
84        CircleEntry<T> inserted = this;
85        foreach (T obj in list) {
86          if (first) {
87            Value = obj;
88            Previous = this;
89            Next = this;
90          }
91          else {
92            inserted = inserted.InsertAfter (obj);
93          }
94          first = false;
95        }
96      }
97
98      public static CircleEntry<T> Empty
```

```
 99      {
100        get {
101          return new CircleEntry<T> ();
102        }
103      }
104
105      public CircleEntry<T> InsertBefore (T obj)
106      {
107        CircleEntry<T> insert = new CircleEntry<T> (obj);
108        insert.Previous = this.Previous;
109        insert.Next = this;
110        this.Previous.Next = insert;
111        this.Previous = insert;
112        return insert;
113      }
114
115      public CircleEntry<T> InsertAfter (T obj)
116      {
117        //Log.Debug (this, ".InsertAfter (", obj, ")");
118        CircleEntry<T> insert = new CircleEntry<T> (obj);
119        insert.Next = this.Next;
120        insert.Previous = this;
121        this.Next.Previous = insert;
122        this.Next = insert;
123        return insert;
124      }
125
126      public void Remove ()
127      {
128        Previous.Next = Next;
129        Next.Previous = Previous;
130        Previous = null;
131        Next = null;
132      }
133
134      private bool IsEmpty
135      {
136        get {
137          return (Next == this || Next == null) && (Previous == this || Previous
138        }
139      }
140
141      public int Count
142      {
143        get {
144          if (IsEmpty) {
145            return 0;
146          }
147          else {
148            CircleEntry<T> current = this;
149            int count = 0;
150            do {
151              ++count;
152              current = current.Next;
153            }
154            while (current != this);
155            return count;
156          }
157        }
158      }
159
```

```
160       public bool Contains (T obj, out IEnumerable<CircleEntry<T>> item)
161       {
162         item = Find (obj);
163         return item.Count () > 0;
164       }
165
166       public bool Contains (Func<T, bool> func, out IEnumerable<CircleEntry<T>>
167       {
168         item = Find (func);
169         return item.Count () > 0;
170       }
171
172       public bool Contains (T obj, out CircleEntry<T> item)
173       {
174         item = Find (obj).ElementAtOrDefault (0);
175         return item != null;
176       }
177
178       public bool Contains (Func<T, bool> func, out CircleEntry<T> item)
179       {
180         item = Find (func).ElementAtOrDefault (0);
181         return item != null;
182       }
183
184       public IEnumerable<CircleEntry<T>> Find (T obj)
185       {
186         return Find ((t) => t.Equals (obj));
187       }
188
189       public IEnumerable<CircleEntry<T>> Find (Func<T, bool> func)
190       {
191         CircleEntry<T> current = this;
192         do {
193           if (func (current.Value)) {
194             yield return current;
195           }
196           current = current.Next;
197         }
198         while (current != this);
199         yield break;
200       }
201
202       public int IndexOf (T obj)
203       {
204         return IndexOf ((t) => t.Equals (obj));
205       }
206
207       public int IndexOf (Func<T, bool> func)
208       {
209         int i = 0;
210         CircleEntry<T> current = this;
211         do {
212           if (func (current.Value)) {
213             return i;
214           }
215           current = current.Next;
216           ++ i;
217         }
218         while (current != this);
219         return -1;
220       }
```

```
221
222       public IEnumerable<T> RangeTo (CircleEntry<T> other)
223       {
224         CircleEntry<T> current = this;
225         do {
226           yield return current.Value;
227           current = current.Next;
228         }
229         while (current != other.Next && current != this);
230       }
231
232       public IEnumerable<T> WayTo (T other)
233       {
234         CircleEntry<T> current = this;
235         while (!current.Value.Equals (other)) {
236           yield return current.Value;
237           current = current.Next;
238           if (current == this) {
239             break;
240           }
241         }
242       }
243
244       public IEnumerable<Tuple<T,T>> Pairs
245       {
246         get {
247           CircleEntry<T> current = this;
248           do {
249             yield return Tuple.Create (current.Value, current.Next.Value);
250             current = current.Next;
251           }
252           while (current != this);
253         }
254       }
255
256       public IEnumerable<Tuple<T,T,T>> Triples
257       {
258         get {
259           CircleEntry<T> current = this;
260           do {
261             yield return Tuple.Create (current.Previous.Value, current.Value, cu
262             current = current.Next;
263           }
264           while (current != this);
265         }
266       }
267
268       public IEnumerator<T> GetEnumerator ()
269       {
270         CircleEntry<T> current = this;
271         do {
272           //Log.Debug (this, " => ", current.Content);
273           yield return current.Value;
274           current = current.Next;
275         }
276         while (current != this);
277       }
278
279       // explicit interface implementation for nongeneric interface
280       IEnumerator IEnumerable.GetEnumerator ()
281       {
```

```
282       return GetEnumerator (); // just return the generic version
283     }
284
285     [ExcludeFromCodeCoverageAttribute]
286     public override string ToString ()
287     {
288       if (IsEmpty) {
289         return "CircleEntry (" + Value.ToString () + ")";
290       }
291       else {
292         return "CircleEntry.Empty";
293       }
294     }
295
296     public static CircleEntry<T> operator + (CircleEntry<T> circle, int i)
297     {
298       CircleEntry<T> next = circle;
299       while (i > 0) {
300         next = next.Next;
301         i--;
302       }
303       while (i < 0) {
304         next = next.Previous;
305         i++;
306       }
307       return next;
308     }
309
310     public T this [int index]
311     {
312       get {
313         return (this + index).Value;
314       }
315       set {
316         (this + index).Value = value;
317       }
318     }
319
320     public static CircleEntry<T> operator - (CircleEntry<T> circle, int i)
321     {
322       return circle + (-i);
323     }
324
325     public static CircleEntry<T> operator ++ (CircleEntry<T> circle)
326     {
327       return circle.Next;
328     }
329
330     public static CircleEntry<T> operator -- (CircleEntry<T> circle)
331     {
332       return circle.Previous;
333     }
334
335     public static implicit operator T (CircleEntry<T> circle)
336     {
337       return circle.Value;
338     }
339
340     public bool IsReadOnly { get { return false; } }
341
342     public bool Contains (T obj)
```

```
343       {
344         CircleEntry<T> item = Find (obj).ElementAtOrDefault (0);
345         return item != null;
346       }
347
348       public bool Remove (T value)
349       {
350         CircleEntry<T> item;
351         if (Contains (value, out item)) {
352           item.Remove ();
353           return true;
354         }
355         else {
356           return false;
357         }
358       }
359
360       public void RemoveAt (int i)
361       {
362         (this + i).Remove ();
363       }
364
365       public void Insert (int i, T value)
366       {
367         (this + i).InsertBefore (value);
368       }
369
370       public void Add (T value)
371       {
372         if (Value == null) {
373           Value = value;
374         }
375         else {
376           InsertBefore (value);
377         }
378       }
379
380       public void Clear ()
381       {
382         Remove ();
383         Next = Previous = this;
384       }
385
386       public void CopyTo (T[] array, int start)
387       {
388         foreach (T value in this) {
389           array.SetValue (value, start);
390           ++start;
391         }
392       }
393     }
394
395     public static class CircleExtensions
396     {
397       public static CircleEntry<T> ToCircle<T> (this IEnumerable<T> enumerable)
1   398       {
1   399         return new CircleEntry<T> (enumerable);
1   400       }
401     }
402   }
```

Knot3.Data.Direction

Summary

Class:	Knot3.Data.Direction
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Direction.cs
Coverage:	98.6%
Covered lines:	71
Uncovered lines:	1
Coverable lines:	72
Total lines:	255

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
FromAxis(...)	5	100	85.71
FromString(...)	3	91.67	80
op_Addition(...)	1	100	100
op_Subtraction(...)	1	100	100
op_Division(...)	1	100	100
op_Multiply(...)	1	100	100
op_Equality(...)	4	100	85.71
op_Inequality(...)	1	100	100
Equals(...)	2	100	66.67
Equals(...)	5	100	100
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Direction.cs

```
#  Line  Coverage
   1  #region Copyright
   2
   3  /*
   4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
   5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
   6   *
   7   * Permission is hereby granted, free of charge, to any person obtaining a cop
   8   * of this software and associated documentation files (the "Software"), to de
   9   * in the Software without restriction, including without limitation the right
  10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
  11   * copies of the Software, and to permit persons to whom the Software is
  12   * furnished to do so, subject to the following conditions:
  13   *
  14   * The above copyright notice and this permission notice shall be included in
  15   * copies or substantial portions of the Software.
  16   *
  17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
  22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
  23   * SOFTWARE.
```

```
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37  using Microsoft.Xna.Framework.Audio;
38  using Microsoft.Xna.Framework.Content;
39  using Microsoft.Xna.Framework.GamerServices;
40  using Microsoft.Xna.Framework.Graphics;
41  using Microsoft.Xna.Framework.Input;
42  using Microsoft.Xna.Framework.Media;
43  using Microsoft.Xna.Framework.Net;
44  using Microsoft.Xna.Framework.Storage;
45
46  using Knot3.Core;
47  using Knot3.GameObjects;
48  using Knot3.Input;
49  using Knot3.RenderEffects;
50  using Knot3.Screens;
51  using Knot3.Widgets;
52
53  #endregion
54
55  namespace Knot3.Data
56  {
57    /// <summary>
58    /// Eine Wertesammlung der mglichen Richtungen in einem dreidimensionalen R
59    /// Wird benutzt, damit keine ungltigen Kantenrichtungen angegeben werden k
60    /// Dies ist eine Klasse und kein Enum, kann aber
61    /// uneingeschrnkt wie eines benutzt werden (Typesafe Enum Pattern).
62    /// </summary>
63    public sealed class Direction : IEquatable<Direction>
64    {
65      #region Enumeration Values
66
67      /// <summary>
68      /// Links.
69      /// </summary>
1     70      public static readonly Direction Left = new Direction (Vector3.Left, "Left
71      /// <summary>
72      /// Rechts.
73      /// </summary>
1     74      public static readonly Direction Right = new Direction (Vector3.Right, "Ri
75      /// <summary>
76      /// Hoch.
77      /// </summary>
1     78      public static readonly Direction Up = new Direction (Vector3.Up, "Up");
79      /// <summary>
80      /// Runter.
81      /// </summary>
1     82      public static readonly Direction Down = new Direction (Vector3.Down, "Down
83      /// <summary>
84      /// Vorwrts.
```

```
        85      /// </summary>
    1   86      public static readonly Direction Forward = new Direction (Vector3.Forward,
        87      /// <summary>
        88      /// Rckwrts.
        89      /// </summary>
    1   90      public static readonly Direction Backward = new Direction (Vector3.Backwar
        91      /// <summary>
        92      /// Keine Richtung.
        93      /// </summary>
    1   94      public static readonly Direction Zero = new Direction (Vector3.Zero, "Zero
        95
        96      #endregion
        97
        98      #region Static Attributes
        99
    1  100      public static readonly Direction[] Values = {
       101        Left, Right, Up, Down, Forward,  Backward
       102      };
    1  103      private static readonly Dictionary<Direction, Direction> ReverseMap
       104          = new Dictionary<Direction, Direction> ()
       105      {
       106        { Left, Right }, { Right, Left },
       107        { Up, Down }, { Down, Up },
       108        { Forward, Backward }, { Backward, Forward },
       109        { Zero, Zero }
       110      };
       111
    1  112      private static readonly Dictionary<Direction, Axis> AxisMap
       113          = new Dictionary<Direction, Axis> ()
       114      {
       115        { Left, Axis.X }, { Right, Axis.X },
       116        { Up, Axis.Y }, { Down, Axis.Y },
       117        { Forward, Axis.Z }, { Backward, Axis.Z },
       118        { Zero, Axis.Zero }
       119      };
       120
       121      #endregion
       122
       123      #region Properties
       124
 1969  125      public Vector3 Vector { get; private set; }
       126
  384  127      public string Description { get; private set; }
       128
  183  129      public Direction Reverse { get { return ReverseMap [this]; } }
       130
   33  131      public Axis Axis { get { return AxisMap[this]; } }
       132
       133      #endregion
       134
       135      #region Constructors
       136
    7  137      private Direction (Vector3 vector, string desciption)
    7  138      {
    7  139        Vector = vector;
    7  140        Description = desciption;
    7  141      }
       142
       143      #endregion
       144
       145      #region Methods and Operators
```

```
        146
        147       public static Direction FromAxis (Axis axis)
      3 148       {
      3 149         return axis == Axis.X ? Right : axis == Axis.Y ? Up : axis == Axis.Z ? B
      3 150       }
        151
        152       public static Direction FromString (string str)
      6 153       {
     75 154         foreach (Direction direction in Values) {
     27 155           if (str.ToLower () == direction.Description.ToLower ()) {
      6 156             return direction;
        157           }
     15 158         }
      0 159         return null;
      6 160       }
        161
        162       [ExcludeFromCodeCoverageAttribute]
        163       public override string ToString ()
        164       {
        165         return Description;
        166       }
        167
        168       public static Vector3 operator + (Vector3 v, Direction d)
    317 169       {
    317 170         return v + d.Vector;
    317 171       }
        172
        173       public static Vector3 operator - (Vector3 v, Direction d)
      1 174       {
      1 175         return v - d.Vector;
      1 176       }
        177
        178       public static Vector3 operator / (Direction d, float i)
    600 179       {
    600 180         return d.Vector / i;
    600 181       }
        182
        183       public static Vector3 operator * (Direction d, float i)
      1 184       {
      1 185         return d.Vector * i;
      1 186       }
        187
        188       public static bool operator == (Direction a, Direction b)
    792 189       {
        190         // If both are null, or both are same instance, return true.
   1082 191         if (System.Object.ReferenceEquals (a, b)) {
    290 192           return true;
        193         }
        194
        195         // If one is null, but not both, return false.
    579 196         if (((object)a == null) || ((object)b == null)) {
     77 197           return false;
        198         }
        199
        200         // Return true if the fields match:
    425 201         return a.Vector == b.Vector;
    792 202       }
        203
        204       public static bool operator != (Direction d1, Direction d2)
    109 205       {
    109 206         return !(d1 == d2);
```

```
109  207      }
     208
     209      public bool Equals (Direction other)
 77  210      {
 77  211        return other != null && Vector == other.Vector;
 77  212      }
     213
     214      public override bool Equals (object other)
  7  215      {
  8  216        if (other == null) {
  1  217          return false;
     218        }
  7  219        else if (other is Direction) {
  1  220          return Equals (other as Direction);
     221        }
  6  222        else if (other is Vector3) {
  1  223          return Vector.Equals ((Vector3)other);
     224        }
  6  225        else if (other is string) {
  2  226          return Description.Equals ((string)other);
     227        }
  2  228        else {
  2  229          return false;
     230        }
  7  231      }
     232
     233      public static implicit operator string (Direction direction)
  9  234      {
  9  235        return direction.Description;
  9  236      }
     237
     238      public static implicit operator Vector3 (Direction direction)
 23  239      {
 23  240        return direction.Vector;
 23  241      }
     242
     243      [ExcludeFromCodeCoverageAttribute]
     244      public override int GetHashCode ()
     245      {
     246        return Description.GetHashCode ();
     247      }
     248
     249      #endregion
     250    }
     251
     252    public enum Axis {
     253      X, Y, Z, Zero
     254    }
     255  }
```

Knot3.Data.Edge

Summary

Class:	Knot3.Data.Edge
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Edge.cs
Coverage:	95%
Covered lines:	76
Uncovered lines:	4
Coverable lines:	80
Total lines:	246

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	100	100
op_Equality(...)	4	100	85.71
op_Inequality(...)	1	100	100
Equals(...)	2	100	100
Equals(...)	6	57.89	63.64
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
RandomColor()	1	100	100
RandomColor(...)	1	100	100
RandomEdge()	6	100	54.55
Clone()	1	100	100
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Edge.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.

```
         24   */
         25
         26  #endregion
         27
         28  #region Using
         29
         30  using System;
         31  using System.Collections;
         32  using System.Collections.Generic;
         33  using System.Diagnostics.CodeAnalysis;
         34  using System.Linq;
         35
         36  using Microsoft.Xna.Framework;
         37  using Microsoft.Xna.Framework.Audio;
         38  using Microsoft.Xna.Framework.Content;
         39  using Microsoft.Xna.Framework.GamerServices;
         40  using Microsoft.Xna.Framework.Graphics;
         41  using Microsoft.Xna.Framework.Input;
         42  using Microsoft.Xna.Framework.Media;
         43  using Microsoft.Xna.Framework.Net;
         44  using Microsoft.Xna.Framework.Storage;
         45
         46  using Knot3.Core;
         47  using Knot3.GameObjects;
         48  using Knot3.Input;
         49  using Knot3.RenderEffects;
         50  using Knot3.Screens;
         51  using Knot3.Widgets;
         52
         53  #endregion
         54
         55  namespace Knot3.Data
         56  {
         57    /// <summary>
         58    /// Eine Kante eines Knotens, die aus einer Richtung und einer Farbe, sowie
         59    /// </summary>
         60    public sealed class Edge : IEquatable<Edge>, ICloneable
         61    {
         62      #region Properties
         63
         64      /// <summary>
         65      /// Die Farbe der Kante.
         66      /// </summary>
  439    67      public Color Color { get; set; }
         68
         69      /// <summary>
         70      /// Die Richtung der Kante.
         71      /// </summary>
 1612    72      public Direction Direction { get; private set; }
         73
         74      /// <summary>
         75      /// Die Liste der Flchennummern, die an die Kante angrenzen.
         76      /// </summary>
  420    77      public HashSet<int> Rectangles { get; private set; }
         78
         79      private int id;
    1    80      private static int previousId = 0;
         81
         82      #endregion
         83
         84      #region Constructors
```

```
          85
          86      /// <summary>
          87      /// Erstellt eine neue Kante mit der angegebenen Richtung.
          88      /// </summary>
   161    89      public Edge (Direction direction)
   161    90      {
   161    91        Direction = direction;
   161    92        Color = DefaultColor;
   161    93        id = ++previousId;
   161    94        Rectangles = new HashSet<int> ();
   161    95      }
          96
          97      /// <summary>
          98      /// Erstellt eine neue Kante mit der angegebenen Richtung und Farbe.
          99      /// </summary>
    19   100      public Edge (Direction direction, Color color)
    19   101      {
    19   102        Direction = direction;
    19   103        Color = color;
    19   104        id = ++previousId;
    19   105        Rectangles = new HashSet<int>();
    19   106      }
         107
         108      #endregion
         109
         110      #region Methods
         111
         112      public static bool operator == (Edge a, Edge b)
    76   113      {
         114        // If both are null, or both are same instance, return true.
    78   115        if (System.Object.ReferenceEquals (a, b)) {
     2   116          return true;
         117        }
         118
         119        // If one is null, but not both, return false.
   145   120        if (((object)a == null) || ((object)b == null)) {
    71   121          return false;
         122        }
         123
         124        // Return true if the fields match:
     3   125        return a.id == b.id;
    76   126      }
         127
         128      public static bool operator != (Edge a, Edge b)
    74   129      {
    74   130        return !(a == b);
    74   131      }
         132
         133      public bool Equals (Edge other)
    72   134      {
    72   135        return other != null && this.id == other.id;
    72   136      }
         137
         138      public override bool Equals (object other)
    30   139      {
    30   140        if (other == null) {
     0   141          return false;
         142        }
    59   143        else if (other is Edge) {
    29   144          return Equals (other as Edge);
         145        }
```

```
1   146         else if (other is Direction) {
0   147           return Direction.Equals (other as Direction);
    148         }
1   149         else if (other is Vector3) {
0   150           return Direction.Vector.Equals ((Vector3)other);
    151         }
1   152         else if (other is Color) {
0   153           return Color.Equals ((Color)other);
    154         }
1   155         else {
1   156           return false;
    157         }
30  158       }
    159
    160       [ExcludeFromCodeCoverageAttribute]
    161       public override int GetHashCode ()
    162       {
    163         return id;
    164       }
    165
    166       [ExcludeFromCodeCoverageAttribute]
    167       public override string ToString ()
    168       {
    169         return Direction + "/" + id.ToString ();
    170       }
    171
    172       public static implicit operator Direction (Edge edge)
9   173       {
9   174         return edge.Direction;
9   175       }
    176
    177       public static implicit operator Vector3 (Edge edge)
7   178       {
7   179         return edge.Direction;
7   180       }
    181
    182       public static implicit operator Color (Edge edge)
18  183       {
18  184         return edge.Color;
18  185       }
    186
    187       #endregion
    188
    189       #region Helper Methods
    190
1   191       private static Random r = new Random ();
    192
    193       public static Color RandomColor ()
1   194       {
1   195         return Colors [r.Next () % Colors.Count];
1   196       }
    197
    198       public static Color RandomColor (GameTime time)
1   199       {
1   200         return Colors [(int)time.TotalGameTime.TotalSeconds % Colors.Count];
1   201       }
    202
    203       public static Edge RandomEdge ()
1   204       {
1   205         int i = r.Next () % 6;
1   206         return i == 0 ? Left : i == 1 ? Right : i == 2 ? Up : i == 3 ? Down : i
```

```
  1  207      }
     208
     209      public object Clone ()
  1  210      {
  1  211        return new Edge (Direction, Color);
  1  212      }
     213
     214      #endregion
     215
     216      #region Static Properties
     217
  1  218      public static List<Color> Colors = new List<Color> ()
     219      {
     220        Color.Red, Color.Green, Color.Blue, Color.Yellow, Color.Orange
     221      };
  1  222      public static Color DefaultColor = RandomColor ();
     223
 21  224      public static Edge Zero { get { return new Edge (Direction.Zero); } }
     225
  3  226      public static Edge UnitX { get { return new Edge (Direction.Right); } }
     227
  3  228      public static Edge UnitY { get { return new Edge (Direction.Up); } }
     229
  3  230      public static Edge UnitZ { get { return new Edge (Direction.Backward); } }
     231
108  232      public static Edge Up { get { return new Edge (Direction.Up); } }
     233
 93  234      public static Edge Down { get { return new Edge (Direction.Down); } }
     235
108  236      public static Edge Right { get { return new Edge (Direction.Right); } }
     237
108  238      public static Edge Left { get { return new Edge (Direction.Left); } }
     239
 18  240      public static Edge Forward { get { return new Edge (Direction.Forward); }
     241
 18  242      public static Edge Backward { get { return new Edge (Direction.Backward);
     243
     244      #endregion
     245    }
     246  }
```

Knot3.Data.Knot

Summary

Class:	Knot3.Data.Knot
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Knot.cs
Coverage:	84.8%
Covered lines:	263
Uncovered lines:	47
Coverable lines:	310
Total lines:	645

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	5	100	55.56
.ctor(...)	8	100	86.67
.ctor(...)	5	100	88.89
IsValidStructure(...	6	83.33	72.73
IsValidStructure(...	2	100	100
TryMove(...)	21	87.30	87.80
MoveCenterToZero()	8	100	86.67
Move(...)	2	100	100
IsValidDirection(...	16	81.25	77.42
onEdgesChanged()	1	0	0
GetEnumerator()	1	100	100
Save()	3	0	0
Clone()	2	100	100
OnSelectionChanged()	1	100	100
AddToSelection(...)	3	100	60
RemoveFromSelection(	2	100	100
ClearSelection()	1	100	100
AddRangeToSelection(	9	100	88.24
IsSelected(...)	1	0	0
System.Collections.I	1	100	100
Save(...)	1	0	0
Equals(...)	8	51.43	46.67
Charakteristic()	9	100	100
.ctor(...)	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Knot.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	`* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	`*  Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	`*`
	7	`* Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	`* of this software and associated documentation files (the "Software"), to de`
	9	`* in the Software without restriction, including without limitation the right`
	10	`* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	`* copies of the Software, and to permit persons to whom the Software is`
	12	`* furnished to do so, subject to the following conditions:`
	13	`*`

```
14   * The above copyright notice and this permission notice shall be included in
15   * copies or substantial portions of the Software.
16   *
17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
23   * SOFTWARE.
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.IO;
35  using System.Linq;
36
37  using Microsoft.Xna.Framework;
38  using Microsoft.Xna.Framework.Audio;
39  using Microsoft.Xna.Framework.Content;
40  using Microsoft.Xna.Framework.GamerServices;
41  using Microsoft.Xna.Framework.Graphics;
42  using Microsoft.Xna.Framework.Input;
43  using Microsoft.Xna.Framework.Media;
44  using Microsoft.Xna.Framework.Net;
45  using Microsoft.Xna.Framework.Storage;
46
47  using Knot3.Core;
48  using Knot3.Development;
49  using Knot3.GameObjects;
50  using Knot3.Input;
51  using Knot3.RenderEffects;
52  using Knot3.Screens;
53  using Knot3.Utilities;
54  using Knot3.Widgets;
55
56  #endregion
57
58  namespace Knot3.Data
59  {
60    /// <summary>
61    /// Diese Klasse reprsentiert einen Knoten, bestehend aus einem Knoten-Meta
62    /// </summary>
63    public sealed class Knot : ICloneable, IEnumerable<Edge>, IEquatable<Knot>
64    {
65      #region Properties
66
67      /// <summary>
68      /// Der Name des Knotens, welcher auch leer sein kann.
69      /// Beim Speichern muss der Nutzer in diesem Fall zwingend einen nichtleer
70      /// Der Wert dieser Eigenschaft wird aus der \glqq Name\grqq -Eigenschaft
71      /// und bei nderungen wieder in diesem gespeichert.
72      /// Beim ndern dieser Eigenschaft wird automatisch auch der im Metadaten-
73      /// </summary>
74      public string Name
```

```
      75      {
30    76        get { return MetaData.Name; }
 3    77        set { MetaData.Name = value; }
      78      }
      79
      80      /// <summary>
      81      /// Das Startelement der doppelt-verketteten Liste, in der die Kanten gesp
      82      /// </summary>
      83      private CircleEntry<Edge> startElement;
      84
      85      /// <summary>
      86      /// Die Metadaten des Knotens.
      87      /// </summary>
51    88      public KnotMetaData MetaData { get; private set; }
      89
      90      /// <summary>
      91      /// Ein Ereignis, das in der Move-Methode ausgelst wird, wenn sich die St
      92      /// </summary>
28    93      public Action EdgesChanged = () => {};
      94
      95      /// <summary>
      96      /// Enthlt die aktuell vom Spieler selektierten Kanten in der Reihenfolge
      97      /// </summary>
24    98      public IEnumerable<Edge> SelectedEdges { get { return selectedEdges; } }
      99
     100      /// <summary>
     101      /// Enthlt die selektierten Kanten.
     102      /// </summary>
     103      private HashSet<Edge> selectedEdges;
     104
     105      /// <summary>
     106      /// WTF?!
     107      /// </summary>
     108      public int debugId;
     109
     110      /// <summary>
     111      /// Wird aufgerufen, wenn sich die Selektion gendert hat.
     112      /// </summary>
39   113      public Action SelectionChanged = () => {};
     114
     115      /// <summary>
     116      /// Enthlt die zuletzt selektierte Kante.
     117      /// </summary>
     118      private CircleEntry<Edge> lastSelected;
     119
     120      /// <summary>
     121      /// Wird aufgerufen, wenn sich die Startkante gendert hat.
     122      /// </summary>
28   123      public Action<Vector3> StartEdgeChanged = (v) => {};
     124
     125      /// <summary>
     126      /// Der Cache fr die Knotencharakteristik.
     127      /// </summary>
28   128      private KnotCharakteristic? CharakteristicCache = null;
     129
35   130      public Vector3 OffSet { get; private set;}
     131
     132      #endregion
     133
     134      #region Constructors
     135
```

```
        136      /// <summary>
        137      /// Erstellt einen minimalen Standardknoten. Das Metadaten-Objekt enthlt
        138      /// die das Speicherformat und den Dateinamen beinhalten, den Wert \glqq n
        139      /// </summary>
      1 140      public Knot ()
      1 141      {
      1 142        debugId++;
      1 143        MetaData = new KnotMetaData (String.Empty, () => startElement.Count, nul
      1 144        startElement = new CircleEntry<Edge> (new Edge[] {
        145          // Edge.Up, Edge.Right, Edge.Right, Edge.Down, Edge.Backward,
        146          // Edge.Up, Edge.Left, Edge.Left, Edge.Down, Edge.Forward
        147          Edge.Up, Edge.Right, Edge.Down, Edge.Left
        148        }
        149                                              );
      1 150        selectedEdges = new HashSet<Edge> ();
      1 151        OffSet = Vector3.Zero;
      1 152      }
        153
        154      /// <summary>
        155      /// Erstellt einen neuen Knoten mit dem angegebenen Metadaten-Objekt und d
        156      /// die in der doppelt verketteten Liste gespeichert werden.
        157      /// Die Eigenschaft des Metadaten-Objektes, die die Anzahl der Kanten enth
        158      /// wird auf ein Delegate gesetzt, welches jeweils die aktuelle Anzahl der
        159      /// </summary>
     24 160      public Knot (KnotMetaData metaData, IEnumerable<Edge> edges)
     24 161      {
     24 162        debugId++;
     24 163        Stack<Direction> structure = new Stack<Direction> ();
    726 164        foreach (Edge edge in edges) {
    218 165          structure.Push (edge.Direction);
    218 166        }
     25 167        if (!IsValidStructure (structure)) {
      1 168          throw new InvalidDataException ();
        169        }
     23 170        MetaData = new KnotMetaData (
        171            name: metaData.Name,
      1 172            countEdges: () => this.startElement.Count,
        173            format: metaData.Format,
        174            filename: metaData.Filename
        175        );
     23 176        this.startElement = new CircleEntry<Edge> (edges);
     23 177        selectedEdges = new HashSet<Edge> ();
     23 178        OffSet = Vector3.Zero;
     23 179      }
        180
      3 181      private Knot (KnotMetaData metaData, CircleEntry<Edge> start, HashSet<Edge
      3 182      {
      3 183        startElement = start;
      3 184        MetaData = new KnotMetaData (
        185            name: metaData.Name,
      0 186            countEdges: () => this.startElement.Count,
        187            format: metaData.Format,
        188            filename: metaData.Filename
        189        );
      3 190        selectedEdges = selected;
      3 191        OffSet = offset;
      3 192      }
        193
        194      #endregion
        195
        196      #region Methods
```

```
      197
      198       /// <summary>
      199       /// Prft ob die gegeben Struktur einen gltigen Knoten darstellt.
      200       /// </summary>
      201       public bool IsValidStructure (IEnumerable<Direction> knot)
   27 202       {
   27 203         Vector3 position3D = Vector3.Zero;
   27 204         HashSet<Vector3> occupancy = new HashSet<Vector3> ();
   27 205         if (knot.Count () < 4) {
    0 206           return false;
      207         }
  795 208         foreach (Direction peek in knot) {
  238 209           if (occupancy.Contains (position3D + (peek / 2))) {
    0 210             return false;
      211           }
  238 212           else {
  238 213             occupancy.Add (position3D + (peek / 2));
  238 214             position3D += peek;
  238 215           }
  238 216         }
   28 217         if (position3D.DistanceTo (Vector3.Zero) > 0.00001f) {
    1 218           return false;
      219         }
   26 220         return true;
   27 221       }
      222
      223       private bool IsValidStructure (IEnumerable<Edge> edges)
    3 224       {
   43 225         return IsValidStructure (from e in edges select e.Direction);
    3 226       }
      227
      228       /// <summary>
      229       /// Verschiebt die aktuelle Kantenauswahl in die angegebene Richtung um di
      230       /// </summary>
      231       public bool TryMove (Direction direction, int distance, out Knot newknot)
    9 232       {
   12 233         if (direction == Direction.Zero || distance == 0) {
    3 234           newknot = this;
    3 235           return true;
      236         }
      237
    6 238         Log.Debug ("TryMove: direction = ", direction, ", distance = ", distance
   38 239         Log.Debug ("Current Knot #", startElement.Count, " = ", string.Join (",
      240
    6 241         HashSet<Edge> selected = new HashSet<Edge> (selectedEdges);
    6 242         CircleEntry<Edge> newCircle = CircleEntry<Edge>.Empty;
      243
  114 244         foreach (Tuple<Edge, Edge, Edge> triple in startElement.Triples) {
   32 245           Edge previousEdge = triple.Item1;
   32 246           Edge currentEdge = triple.Item2;
   32 247           Edge nextEdge = triple.Item3;
      248
   38 249           if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (p
   15 250             distance.Repeat (i => newCircle.Add (new Edge (direction: direction,
    6 251           }
      252
   32 253           newCircle.Add (currentEdge);
      254
   38 255           if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (n
   15 256             distance.Repeat (i => newCircle.Add (new Edge (direction: direction.
    6 257           }
```

```
 32  258        }
     259
 56  260        Log.Debug ("New Knot #", newCircle.Count, " = ", string.Join (", ", from
     261
  6  262        Vector3 localOffset = OffSet;
  6  263        CircleEntry<Edge> current = newCircle;
 45  264        do {
 55  265          if (current [- 1].Direction == current [- 2].Direction.Reverse) {
     266            // Selektierte nicht lschen
 13  267            if (selected.Contains (current [- 1]) || selected.Contains (current
  3  268              Log.Debug ("Error: Selektierte nicht lschen");
  3  269              newknot = null;
  3  270              return false;
     271            }
  7  272            if (newCircle == current - 1) {
  0  273              localOffset += (current - 1).Value;
  0  274              newCircle = current;
  0  275            }
  8  276            else if (newCircle == current - 2) {
  1  277              localOffset += (current - 1).Value.Direction + (current - 1).Value
  1  278              newCircle = current;
  1  279            }
  7  280            (current - 2).Remove ();
  7  281            (current - 1).Remove ();
  7  282          }
 42  283          ++ current;
 42  284        }
 42  285        while (current != newCircle);
     286
 23  287        Log.Debug ("New Knot after Remove #", newCircle.Count, " = ", string.Joi
     288
  3  289        if (!IsValidStructure (newCircle)) {
  0  290          Log.Debug ("Error: newCircle ist keine valide Struktur");
  0  291          newknot = null;
  0  292          return false;
     293        }
  3  294        newknot = new Knot (MetaData, newCircle, selected, localOffset);
  3  295        return true;
  9  296      }
     297
     298      public Vector3 MoveCenterToZero ()
  1  299      {
  1  300        Vector3 position3D = Vector3.Zero;
  1  301        Dictionary<Vector3, Edge> occupancy = new Dictionary<Vector3, Edge>();
 21  302        foreach (Edge edge in startElement) {
  6  303          occupancy.Add (position3D + (edge.Direction / 2), edge);
  6  304          position3D += edge;
  6  305        }
  1  306        Vector3 mid = Vector3.Zero;
 21  307        foreach (KeyValuePair<Vector3,Edge> pos in occupancy) {
  6  308          mid += pos.Key;
  6  309        }
  1  310        mid /= startElement.Count;
  1  311        float minDistance = mid.Length ();
  1  312        Edge newStart = startElement.Value;
 21  313        foreach (KeyValuePair<Vector3,Edge> pos in occupancy) {
  6  314          float testDistance = pos.Key.DistanceTo (mid);
  8  315          if (testDistance < minDistance) {
  2  316            newStart = pos.Value;
  2  317            minDistance = testDistance;
  2  318          }
```

```
  6  319         }
  1  320         Vector3 offset = Vector3.Zero;
  6  321         foreach (Edge edge in startElement.WayTo (newStart)) {
  1  322           offset += edge;
  1  323         }
  1  324         startElement.Contains (newStart, out startElement);
  1  325         offset += OffSet;
  1  326         OffSet = Vector3.Zero;
  1  327         return offset;
  1  328       }
     329
     330       /// <summary>
     331       /// Verschiebt die aktuelle Kantenauswahl in die angegebene Richtung um di
     332       /// </summary>
     333       public bool Move (Direction direction, int distance)
  9  334       {
     335         Knot newKnot;
 15  336         if (TryMove (direction, distance, out newKnot)) {
  6  337           startElement = newKnot.startElement;
  6  338           selectedEdges = newKnot.selectedEdges;
  6  339           return true;
     340         }
  3  341         else {
  3  342           return false;
     343         }
  9  344       }
     345
     346       /// <summary>
     347       /// Gibt an ob ein Move in diese Richtung berhaupt mglich ist.
     348       /// </summary>
     349       public bool IsValidDirection (Direction direction)
  6  350       {
     351         // Nichts selektiert
  6  352         if (selectedEdges.Count == 0) {
  0  353           return false;
     354         }
     355         // Alles selektiert
  6  356         if (selectedEdges.Count == startElement.Count) {
  0  357           return true;
     358         }
     359
  6  360         HashSet<Axis> axes = new HashSet<Axis> ();
 76  361         foreach (Tuple<Edge, Edge, Edge> triple in startElement.Triples) {
 20  362           Edge previousEdge = triple.Item1;
 20  363           Edge currentEdge = triple.Item2;
 20  364           Edge nextEdge = triple.Item3;
     365
     366           // Wenn Kante nach der Bewegung gelscht werden msste ist ein Zug nic
 20  367           if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (p
  1  368                   && currentEdge.Direction == direction.Reverse && previousEdge.
  1  369             return false;
     370           }
     371           // Wenn Kante nach der Bewegung gelscht werden msste ist ein Zug nic
 19  372           if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (n
  1  373                   && currentEdge.Direction == direction && nextEdge.Direction !=
  1  374             return false;
     375           }
     376
 22  377           if (selectedEdges.Contains (currentEdge)) {
  4  378             axes.Add (currentEdge.Direction.Axis);
  4  379           }
```

```
 18  380        }
     381        // Wenn alle Kanten entlang einer Achse angeordnet sind und die Verschie
  4  382        if (axes.Count == 1 && axes.Contains (direction.Axis)) {
  0  383          return false;
     384        }
  4  385        return true;
  6  386      }
     387
     388      private void onEdgesChanged ()
  0  389      {
  0  390        CharakteristicCache = null;
  0  391        EdgesChanged ();
  0  392      }
     393
     394      /// <summary>
     395      /// Gibt die doppelt-verkettete Kantenliste als Enumerator zurck.
     396      /// </summary>
     397      public IEnumerator<Edge> GetEnumerator ()
 36  398      {
 36  399        return startElement.GetEnumerator ();
 36  400      }
     401
     402      /// <summary>
     403      /// Speichert den Knoten unter dem Dateinamen in dem Dateiformat, das in d
     404      /// Enthalten entweder die Dateiname-Eigenschaft, die Dateiformat-Eigensch
     405      /// oder beide den Wert \glqq null\grqq, dann wird eine IOException geworf
     406      /// </summary>
     407      public void Save ()
  0  408      {
  0  409        if (MetaData.Format == null) {
  0  410          throw new IOException ("Error: Knot: MetaData.Format is null!");
     411        }
  0  412        else if (MetaData.Filename == null) {
  0  413          throw new IOException ("Error: Knot: MetaData.Filename is null!");
     414        }
  0  415        else {
  0  416          MetaData.Format.Save (this);
  0  417        }
  0  418      }
     419
     420      /// <summary>
     421      /// Erstellt eine vollstndige Kopie des Knotens, inklusive der Kanten-Dat
     422      /// </summary>
     423      public object Clone ()
  2  424      {
  2  425        CircleEntry<Edge> newCircle = new CircleEntry<Edge> (startElement as IEn
  2  426        KnotMetaData metaData = new KnotMetaData (
     427            name: MetaData.Name,
  0  428            countEdges: () => 0,
     429            format: MetaData.Format,
     430            filename: MetaData.Filename
     431        );
  2  432        return new Knot (metaData: metaData, edges: newCircle) {
     433          selectedEdges = new HashSet<Edge> (selectedEdges),
     434          EdgesChanged = null,
     435          SelectionChanged = null,
     436        };
  2  437      }
     438
     439      private void OnSelectionChanged ()
 11  440      {
```

```
11  441       SelectionChanged ();
11  442     }
    443
    444     /// <summary>
    445     /// Fgt die angegebene Kante zur aktuellen Kantenauswahl hinzu.
    446     /// </summary>
    447     public void AddToSelection (Edge edge)
 5  448     {
 5  449       IEnumerable<CircleEntry<Edge>> found = startElement.Find (edge);
10  450       if (found.Any ()) {
10  451         if (!selectedEdges.Contains (edge)) {
 5  452           selectedEdges.Add (edge);
 5  453         }
 5  454         lastSelected = found.ElementAt (0);
 5  455       }
 5  456       OnSelectionChanged ();
 5  457     }
    458
    459     /// <summary>
    460     /// Entfernt die angegebene Kante von der aktuellen Kantenauswahl.
    461     /// </summary>
    462     public void RemoveFromSelection (Edge edge)
 2  463     {
 2  464       selectedEdges.Remove (edge);
 3  465       if (lastSelected.Value == edge) {
 1  466         lastSelected = null;
 1  467       }
 2  468       OnSelectionChanged ();
 2  469     }
    470
    471     /// <summary>
    472     /// Hebt die aktuelle Kantenauswahl auf.
    473     /// </summary>
    474     public void ClearSelection ()
 2  475     {
 2  476       selectedEdges.Clear ();
 2  477       lastSelected = null;
 2  478       OnSelectionChanged ();
 2  479     }
    480
    481     /// <summary>
    482     /// Fgt alle Kanten auf dem krzesten Weg zwischen der zuletzt ausgewhlt
    483     /// zur aktuellen Kantenauswahl hinzu. Sind beide Wege gleich lang,
    484     /// wird der Weg in Richtung der ersten Kante ausgewhlt.
    485     /// </summary>
    486     public void AddRangeToSelection (Edge selectedEdge)
 3  487     {
 4  488       if (lastSelected == null) {
 1  489         AddToSelection (selectedEdge);
 1  490         return;
    491       }
 2  492       CircleEntry<Edge> selectedCircle = null;
 4  493       if (startElement.Contains (selectedEdge, out selectedCircle) && selected
 2  494         List<Edge> forward = new List<Edge> (lastSelected.RangeTo (selectedCir
 2  495         List<Edge> backward = new List<Edge> (selectedCircle.RangeTo (lastSele
    496
 3  497         if (forward.Count < backward.Count) {
12  498           foreach (Edge e in forward) {
 5  499             if (!selectedEdges.Contains (e)) {
 2  500               selectedEdges.Add (e);
 2  501             }
```

```
  3   502            }
  1   503          }
  1   504          else {
  9   505            foreach (Edge e in backward) {
  3   506              if (!selectedEdges.Contains (e)) {
  1   507                selectedEdges.Add (e);
  1   508              }
  2   509            }
  1   510          }
  2   511          lastSelected = selectedCircle;
  2   512        }
  2   513        OnSelectionChanged ();
  3   514      }
      515
      516      /// <summary>
      517      /// Prft, ob die angegebene Kante in der aktuellen Kantenauswahl enthalte
      518      /// </summary>
      519      public Boolean IsSelected (Edge edge)
  0   520      {
  0   521        return selectedEdges.Contains (edge);
  0   522      }
      523
      524      /// <summary>
      525      /// Gibt die doppelt-verkettete Kantenliste als Enumerator zurck.
      526      /// [name=IEnumerable.GetEnumerator]
      527      /// [keywords= ]
      528      /// </summary>
      529      IEnumerator IEnumerable.GetEnumerator ()
  2   530      {
  2   531        return GetEnumerator (); // just return the generic version
  2   532      }
      533
      534      /// <summary>
      535      /// Speichert den Knoten unter dem angegebenen Dateinamen in dem angegeben
      536      /// </summary>
      537      public void Save (IKnotIO format, string filename)
  0   538      {
  0   539        KnotMetaData metaData = new KnotMetaData (MetaData.Name, () => MetaData.
  0   540        Knot knotToSave = new Knot (metaData, startElement);
  0   541        format.Save (knotToSave);
  0   542      }
      543
      544      /// <summary>
      545      /// Prft, ob die rumliche Struktur identisch ist, unabhngig von dem Sta
      546      /// [parameters=Knot other]
      547      /// </summary>
      548      public bool Equals (Knot other)
 12   549      {
 12   550        KnotCharakteristic thisCharakteristik = Charakteristic ();
 12   551        KnotCharakteristic otherCharakteristik = other.Charakteristic ();
 18   552        if (thisCharakteristik.CountEdges != otherCharakteristik.CountEdges) {
  6   553          return false;
      554        }
      555        // Bei Struktur im gleicher Richtung
 12   556        if (thisCharakteristik.CharacteristicalEdge.Value.Direction == otherChar
  6   557          CircleEntry<Edge> currentThisElement = thisCharakteristik.Characterist
  6   558          CircleEntry<Edge> currentOtherElement = otherCharakteristik.Characteri
 66   559          while (currentThisElement != thisCharakteristik.CharacteristicalEdge)
 30   560            if (currentThisElement.Value.Direction != currentOtherElement.Value.
  0   561              return false;
      562            }
```

```
30   563            currentThisElement++;
30   564            currentOtherElement++;
30   565          }
 6   566          return true;
     567        }
     568        // Bei Struktur in entgegengesetzter Richtung
 0   569        else if (thisCharakteristik.CharacteristicalEdge.Value.Direction == othe
 0   570          CircleEntry<Edge> currentThisElement = thisCharakteristik.Characterist
 0   571          CircleEntry<Edge> currentOtherElement = otherCharakteristik.Characteri
 0   572          while (currentThisElement != thisCharakteristik.CharacteristicalEdge)
 0   573            if (currentThisElement.Value.Direction != currentOtherElement.Value.
 0   574              return false;
     575            }
 0   576            currentThisElement++;
 0   577            currentOtherElement++;
 0   578          }
 0   579          return true;
     580        }
 0   581        else {
 0   582          return false;
     583        }
12   584      }
     585
     586      /// <summary>
     587      /// Gibt chrakteristische Werte zurck, die bei gleichen Knoten gleich sin
     588      /// Einmal als Key ein eindeutiges Circle\<Edge\> Element und als Value
     589      /// einen Charakteristischen Integer. Momentan die Anzahl der Kanten.
     590      /// </summary>
     591      private KnotCharakteristic Charakteristic ()
24   592      {
35   593        if (CharakteristicCache.HasValue) {
11   594          return CharakteristicCache.Value;
     595        }
     596
13   597        CircleEntry<Edge> charakteristikElement = startElement;
13   598        Vector3 position3D = startElement.Value.Direction;
13   599        Vector3 bestPosition3D = startElement.Value.Direction / 2;
13   600        CircleEntry<Edge> edgePointer = startElement.Next;
     601
13   602        int edgeCount = 1;
257  603        for (edgeCount = 1; edgePointer != startElement; edgePointer++, edgeCoun
77   604          Vector3 nextPosition3D = position3D + edgePointer.Value.Direction / 2;
77   605          if ((nextPosition3D.X < bestPosition3D.X)
     606                  || (nextPosition3D.X == bestPosition3D.X && nextPosition3D.Y <
27   607                  || (nextPosition3D.X == bestPosition3D.X && nextPosition3D.Y =
27   608            bestPosition3D = position3D + edgePointer.Value.Direction / 2;
27   609            charakteristikElement = edgePointer;
27   610          }
77   611          position3D += edgePointer.Value.Direction;
77   612        }
     613
13   614        CharakteristicCache = new KnotCharakteristic (charakteristikElement, edg
13   615        return CharakteristicCache.Value;
24   616      }
     617
     618      [ExcludeFromCodeCoverageAttribute]
     619      public override string ToString ()
     620      {
     621        return "Knot (name=" + Name + ",#edgecount=" + startElement.Count.ToStri
     622               + ",format=" + (MetaData.Format != null ? MetaData.ToString () :
     623               + ")";
```

```
      624       }
      625
      626       #endregion
      627
      628       #region Classes and Structs
      629
      630       private struct KnotCharakteristic {
      631         public CircleEntry<Edge> CharacteristicalEdge { get; private set; }
      632
      633         public int CountEdges { get; private set; }
      634
      635         public KnotCharakteristic (CircleEntry<Edge> characteristicalEdge, int c
   13 636         : this ()
   13 637         {
   13 638           CharacteristicalEdge = characteristicalEdge;
   13 639           CountEdges = countEdges;
   13 640         }
      641       }
      642
      643       #endregion
      644     }
      645   }
```

Knot3.Data.KnotFileIO

Summary

Class:	Knot3.Data.KnotFileIO
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotFileIO.cs
Coverage:	22.2%
Covered lines:	8
Uncovered lines:	28
Coverable lines:	36
Total lines:	156

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	1	100	100
Save(...)	2	0	0
Load(...)	2	0	0
LoadMetaData(...)	2	0	0
MoveNext()	5	50	40

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotFileIO.cs

#	Line	Coverage	
	1		#region Copyright
	2		
	3		/*
	4		* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5		* Gerd Augsburg, Christina Erler, Daniel Warzel
	6		*
	7		* Permission is hereby granted, free of charge, to any person obtaining a cop
	8		* of this software and associated documentation files (the "Software"), to de
	9		* in the Software without restriction, including without limitation the right
	10		* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11		* copies of the Software, and to permit persons to whom the Software is
	12		* furnished to do so, subject to the following conditions:
	13		*
	14		* The above copyright notice and this permission notice shall be included in
	15		* copies or substantial portions of the Software.
	16		*
	17		* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18		* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19		* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20		* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21		* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22		* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23		* SOFTWARE.
	24		*/
	25		
	26		#endregion
	27		
	28		#region Using
	29		
	30		using System;
	31		using System.Collections;
	32		using System.Collections.Generic;

```
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.IO;
      35  using System.Linq;
      36
      37  using Microsoft.Xna.Framework;
      38  using Microsoft.Xna.Framework.Audio;
      39  using Microsoft.Xna.Framework.Content;
      40  using Microsoft.Xna.Framework.GamerServices;
      41  using Microsoft.Xna.Framework.Graphics;
      42  using Microsoft.Xna.Framework.Input;
      43  using Microsoft.Xna.Framework.Media;
      44  using Microsoft.Xna.Framework.Net;
      45  using Microsoft.Xna.Framework.Storage;
      46
      47  using Knot3.Core;
      48  using Knot3.Development;
      49  using Knot3.GameObjects;
      50  using Knot3.Input;
      51  using Knot3.RenderEffects;
      52  using Knot3.Screens;
      53  using Knot3.Utilities;
      54  using Knot3.Widgets;
      55
      56  #endregion
      57
      58  namespace Knot3.Data
      59  {
      60    /// <summary>
      61    /// Implementiert das Speicherformat fr Knoten.
      62    /// </summary>
      63    public sealed class KnotFileIO : IKnotIO
      64    {
      65      #region Properties
      66
      67      /// <summary>
      68      /// Die fr eine Knoten-Datei gltigen Dateiendungen.
      69      /// </summary>
64    70      public IEnumerable<string> FileExtensions
      71      {
64    72        get {
64    73          yield return ".knot";
 0    74          yield return ".knt";
 0    75        }
      76      }
      77
32    78      private Dictionary<string, Knot> KnotCache = new Dictionary<string, Knot>
32    79      private Dictionary<string, KnotMetaData> KnotMetaDataCache = new Dictionar
      80
      81      #endregion
      82
      83      #region Constructors
      84
      85      /// <summary>
      86      /// Erstellt ein KnotFileIO-Objekt.
      87      /// </summary>
32    88      public KnotFileIO ()
32    89      {
32    90      }
      91
      92      #endregion
      93
```

```
        94      #region Methods
        95
        96      /// <summary>
        97      /// Speichert einen Knoten in dem Dateinamen, der in dem Knot-Objekt entha
        98      /// </summary>
        99      public void Save (Knot knot)
  0    100      {
  0    101        KnotStringIO parser = new KnotStringIO (knot);
  0    102        Log.Debug ("KnotFileIO.Save (", knot, ") = #", parser.Content.Length);
  0    103        if (knot.MetaData.Filename == null) {
  0    104          throw new IOException ("Error! knot has no filename: " + knot);
       105        }
  0    106        else {
  0    107          File.WriteAllText (knot.MetaData.Filename, parser.Content);
  0    108        }
  0    109      }
       110
       111      /// <summary>
       112      /// Ldt eines Knotens aus einer angegebenen Datei.
       113      /// </summary>
       114      public Knot Load (string filename)
  0    115      {
  0    116        if (KnotCache.ContainsKey (filename)) {
  0    117          return KnotCache [filename];
       118        }
  0    119        else {
  0    120          Log.Debug ("Load knot from ", filename);
  0    121          KnotStringIO parser = new KnotStringIO (content: string.Join ("\n", Fi
  0    122          return KnotCache [filename] = new Knot (
  0    123              new KnotMetaData (parser.Name, () => parser.CountEdges, this, file
       124              parser.Edges
       125          );
       126        }
  0    127      }
       128
       129      /// <summary>
       130      /// Ldt die Metadaten eines Knotens aus einer angegebenen Datei.
       131      /// </summary>
       132      public KnotMetaData LoadMetaData (string filename)
  0    133      {
  0    134        if (KnotMetaDataCache.ContainsKey (filename)) {
  0    135          return KnotMetaDataCache [filename];
       136        }
  0    137        else {
  0    138          KnotStringIO parser = new KnotStringIO (content: string.Join ("\n", Fi
  0    139          return KnotMetaDataCache [filename] = new KnotMetaData (
       140              name: parser.Name,
  0    141              countEdges: () => parser.CountEdges,
       142              format: this,
       143              filename: filename
       144          );
       145        }
  0    146      }
       147
       148      [ExcludeFromCodeCoverageAttribute]
       149      public override string ToString ()
       150      {
       151        return "KnotFileIO";
       152      }
       153
       154      #endregion
```

```
155     }
156   }
```

Knot3.Data.KnotMetaData

Summary

Class:	Knot3.Data.KnotMetaData
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotMetaData.cs
Coverage:	64%
Covered lines:	32
Uncovered lines:	18
Coverable lines:	50
Total lines:	196

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	3	100	100
.ctor(...)	1	100	100
Equals(...)	3	0	0
Equals(...)	2	0	0
op_Equality(...)	4	0	0
op_Inequality(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotMetaData.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`

```
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.Linq;
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Core;
      47  using Knot3.GameObjects;
      48  using Knot3.Input;
      49  using Knot3.Platform;
      50  using Knot3.RenderEffects;
      51  using Knot3.Screens;
      52  using Knot3.Utilities;
      53  using Knot3.Widgets;
      54
      55  #endregion
      56
      57  namespace Knot3.Data
      58  {
      59    /// <summary>
      60    /// Enthlt Metadaten eines Knotens, die aus einer Spielstand-Datei schnelle
      61    /// als der vollstndige Knoten. Dieses Objekt enthlt keine Datenstruktur z
      62    /// sondern nur Informationen ber den Namen des Knoten und die Anzahl seine
      63    /// dazugehriges Knoten-Objekt existieren, aber jedes Knoten-Objekt enthlt
      64    /// </summary>
      65    public class KnotMetaData : IEquatable<KnotMetaData>
      66    {
      67      #region Properties
      68
      69      /// <summary>
      70      /// Der Anzeigename des Knotens, welcher auch leer sein kann.
      71      /// Beim Speichern muss der Spieler in diesem Fall zwingend
      72      /// einen nichtleeren Namen whlen. Wird ein neuer Anzeigename festgelegt,
      73      /// dann wird der Dateiname ebenfalls auf einen neuen Wert gesetzt, unabh
      74      /// ob er bereits einen Wert enthlt oder \glqq null\grqq ist.
      75      /// Diese Eigenschaft kann ffentlich gelesen und gesetzt werden.
      76      /// </summary>
      77      public string Name
      78      {
39    79        get {
39    80          return name;
39    81        }
33    82        set {
33    83          name = value;
65    84          if (Format == null) {
32    85            Format = new KnotFileIO ();
32    86          }
65    87          if (name != null && name.Length > 0) {
      88            string extension;
64    89            if (Format.FileExtensions.Any ()) {
32    90              extension = Format.FileExtensions.ElementAt (0);
32    91            }
 0    92            else {
```

```
  0   93              throw new ArgumentException ("Every implementation of IKnotIO must
      94            }
 32   95            Filename = SystemInfo.SavegameDirectory + SystemInfo.PathSeparator.T
 32   96          }
 33   97        }
      98      }
      99
     100      private string name;
     101
     102      /// <summary>
     103      /// Das Format, aus dem die Metadaten geladen wurden.
     104      /// Es ist genau dann \glqq null\grqq, wenn die Metadaten nicht aus einer
     105      /// </summary>
220  106      public IKnotIO Format { get; private set; }
     107
     108      /// <summary>
     109      /// Ein Delegate, das die Anzahl der Kanten zurckliefert.
     110      /// Falls dieses Metadaten-Objekt Teil eines Knotens ist, gibt es dynamisc
     111      /// Kanten des Knoten-Objektes zurck. Anderenfalls gibt es eine statische
     112      /// die beim Einlesen der Metadaten vor dem Erstellen dieses Objektes gele
     113      /// </summary>
  3  114      public int CountEdges { get { return countEdges (); } }
     115
     116      private Func<int> countEdges;
     117
     118      /// <summary>
     119      /// Falls die Metadaten aus einer Datei eingelesen wurden, enthlt dieses
     120      /// sonst \glqq null\grqq.
     121      /// </summary>
122  122      public string Filename { get; private set; }
     123
     124      #endregion
     125
     126      #region Constructors
     127
     128      /// <summary>
     129      /// Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knote
     130      /// und einer angegebenen Funktion, welche eine Kantenanzahl zurck gibt.
     131      /// Zustzlich wird der Dateiname oder das Speicherformat angegeben, aus d
     132      /// </summary>
 32  133      public KnotMetaData (string name, Func<int> countEdges, IKnotIO format, st
 32  134      {
 32  135        Name = name;
 32  136        this.countEdges = countEdges;
 32  137        Format = format ?? Format;
 32  138        Filename = filename ?? Filename;
 32  139      }
     140
     141      /// <summary>
     142      /// Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knote
     143      /// und einer angegebenen Funktion, welche eine Kantenanzahl zurck gibt.
     144      /// </summary>
  7  145      public KnotMetaData (string name, Func<int> countEdges)
  7  146      {
  7  147        this.name = name;
  7  148        this.countEdges = countEdges;
  7  149        Format = null;
  7  150        Filename = null;
  7  151      }
     152
     153      #endregion
```

```
      154
      155       #region Methods
      156
      157       public bool Equals (KnotMetaData other)
    0 158       {
    0 159         return other != null && name == other.name && countEdges () == other.cou
    0 160       }
      161
      162       public override bool Equals (object other)
    0 163       {
    0 164         return other != null && Equals (other as KnotMetaData);
    0 165       }
      166
      167       [ExcludeFromCodeCoverageAttribute]
      168       public override int GetHashCode ()
      169       {
      170         return (countEdges ().ToString () + (name ?? String.Empty)).GetHashCode
      171       }
      172
      173       public static bool operator == (KnotMetaData a, KnotMetaData b)
    0 174       {
      175         // If both are null, or both are same instance, return true.
    0 176         if (System.Object.ReferenceEquals (a, b)) {
    0 177           return true;
      178         }
      179
      180         // If one is null, but not both, return false.
    0 181         if (((object)a == null) || ((object)b == null)) {
    0 182           return false;
      183         }
      184
      185         // Return true if the fields match:
    0 186         return a.Equals (b);
    0 187       }
      188
      189       public static bool operator != (KnotMetaData a, KnotMetaData b)
    0 190       {
    0 191         return !(a == b);
    0 192       }
      193
      194       #endregion
      195     }
      196   }
```

Knot3.Data.KnotStringIO

Summary

Class:	Knot3.Data.KnotStringIO
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotStringIO.cs
Coverage:	41.1%
Covered lines:	49
Uncovered lines:	70
Coverable lines:	119
Total lines:	266

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	63.64	100
DecodeEdge(...)	8	30	33.33
EncodeEdge(...)	7	63.64	61.54
EncodeColor(...)	1	100	100
DecodeColor(...)	4	0	0
MoveNext()	8	27.27	20
MoveNext()	5	100	71.43

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\KnotStringIO.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	

```
      30  using System;
      31  using System.Collections;
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.IO;
      35  using System.Linq;
      36
      37  using Microsoft.Xna.Framework;
      38  using Microsoft.Xna.Framework.Audio;
      39  using Microsoft.Xna.Framework.Content;
      40  using Microsoft.Xna.Framework.GamerServices;
      41  using Microsoft.Xna.Framework.Graphics;
      42  using Microsoft.Xna.Framework.Input;
      43  using Microsoft.Xna.Framework.Media;
      44  using Microsoft.Xna.Framework.Net;
      45  using Microsoft.Xna.Framework.Storage;
      46
      47  using Knot3.Core;
      48  using Knot3.Development;
      49  using Knot3.GameObjects;
      50  using Knot3.Input;
      51  using Knot3.RenderEffects;
      52  using Knot3.Screens;
      53  using Knot3.Widgets;
      54
      55  #endregion
      56
      57  namespace Knot3.Data
      58  {
      59    /// <summary>
      60    /// Diese Klasse reprsentiert einen Parser fr das Knoten-Austauschformat u
      61    /// eingelesenen Informationen wie den Namen des Knotens und die Kantenliste
      62    /// </summary>
      63    public sealed class KnotStringIO
      64    {
      65      #region Properties
      66
      67      /// <summary>
      68      /// Der Name der eingelesenen Knotendatei oder des zugewiesenen Knotenobje
      69      /// </summary>
16    70      public string Name { get; set; }
      71
      72      private IEnumerable<string> edgeLines;
      73
      74      /// <summary>
      75      /// Die Kanten der eingelesenen Knotendatei oder des zugewiesenen Knotenob
      76      /// </summary>
 2    77      public IEnumerable<Edge> Edges
      78      {
 2    79        get {
 2    80          Log.Debug ("KnotStringIO.Edges[get] = ", edgeLines.Count ());
10    81          foreach (string _line in edgeLines) {
 2    82            string line = _line;
 2    83            Edge edge = DecodeEdge (line [0]);
 0    84            line = line.Substring (1);
 0    85            if (line.StartsWith ("#")) {
 0    86              line = line.Substring (1);
 0    87            }
 0    88            edge.Color = DecodeColor (line.Substring (0, 8));
 0    89            line = line.Substring (8);
 0    90            if (line.StartsWith ("#")) {
```

```
  0   91              line = line.Substring (1);
  0   92            }
  0   93            if (line.Length > 0) {
  0   94              foreach (int rect in line.Split (',').Select (int.Parse).ToList ()
  0   95                edge.Rectangles.Add (rect);
  0   96              }
  0   97            }
  0   98            yield return edge;
  0   99          }
  0  100        }
  0  101        set {
  0  102          Log.Debug ("KnotStringIO.Edges[set] = #", value.Count ());
  0  103          try {
  0  104            edgeLines = ToLines (value);
  0  105          }
  0  106          catch (Exception ex) {
  0  107            Log.Debug (ex);
  0  108          }
  0  109        }
     110      }
     111
     112      /// <summary>
     113      /// Die Anzahl der Kanten der eingelesenen Knotendatei oder des zugewiesen
     114      /// </summary>
     115      public int CountEdges
     116      {
  0  117        get {
  0  118          return edgeLines.Where ((l) => l.Trim ().Length > 0).Count ();
  0  119        }
     120      }
     121
     122      /// <summary>
     123      /// Erstellt aus den \glqq Name\grqq - und \glqq Edges\grqq -Eigenschaften
     124      /// die als Dateiinhalt in einer Datei eines Spielstandes einen gltigen K
     125      /// </summary>
     126      public string Content
     127      {
 10  128        get {
 10  129          return Name + "\n" + string.Join ("\n", edgeLines);
 10  130        }
  4  131        set {
  8  132          if (value.Trim ().Contains ("\n")) {
  4  133            string[] parts = value.Split (new char[] {'\r','\n'}, StringSplitOpt
  4  134            Name = parts [0];
  4  135            edgeLines = parts.Skip (1);
  4  136          }
  0  137          else {
  0  138            Name = value;
  0  139            edgeLines = new string[] {};
  0  140          }
  4  141        }
     142      }
     143
     144      #endregion
     145
     146      #region Constructors
     147
     148      /// <summary>
     149      /// Liest das in der angegebenen Zeichenkette enthaltene Dateiformat ein.
     150      /// so werden die \glqq Name\grqq - und \glqq Edges\grqq -Eigenschaften au
     151      /// Enthlt es einen ungltigen Knoten, so wird eine IOException geworfen
```

```
      152       /// </summary>
    4 153       public KnotStringIO (string content)
    4 154       {
    4 155         Content = content;
    4 156       }
      157
      158       /// <summary>
      159       /// Erstellt ein neues Objekt und setzt die \glqq Name\grqq - und \glqq Ed
      160       /// im angegebenen Knoten enthaltenen Werte.
      161       /// </summary>
    2 162       public KnotStringIO (Knot knot)
    2 163       {
    2 164         Name = knot.Name;
    2 165         try {
    2 166           edgeLines = ToLines (knot);
    2 167         }
    0 168         catch (Exception ex) {
    0 169           Log.Debug (ex);
    0 170         }
    2 171       }
      172
      173       #endregion
      174
      175       #region Methods
      176
      177       private static IEnumerable<string> ToLines (IEnumerable<Edge> edges)
    6 178       {
  738 179         foreach (Edge edge in edges) {
  240 180           yield return EncodeEdge (edge) + "#" + EncodeColor (edge.Color) + "#"
  240 181         }
    6 182       }
      183
      184       private static Edge DecodeEdge (char c)
    2 185       {
    2 186         switch (c) {
      187         case 'X':
    0 188           return Edge.Right;
      189         case 'x':
    0 190           return Edge.Left;
      191         case 'Y':
    0 192           return Edge.Up;
      193         case 'y':
    0 194           return Edge.Down;
      195         case 'Z':
    0 196           return Edge.Backward;
      197         case 'z':
    0 198           return Edge.Forward;
      199         default:
    2 200           throw new IOException ("Failed to decode Edge: '" + c + "'!");
      201         }
    0 202       }
      203
      204       private static char EncodeEdge (Edge edge)
  240 205       {
  300 206         if (edge.Direction == Direction.Right) {
   60 207           return 'X';
      208         }
  240 209         else if (edge.Direction == Direction.Left) {
   60 210           return  'x';
      211         }
  180 212         else if (edge.Direction == Direction.Up) {
```

```
 60  213          return  'Y';
     214        }
120  215        else if (edge.Direction == Direction.Down) {
 60  216          return  'y';
     217        }
  0  218        else if (edge.Direction == Direction.Backward) {
  0  219          return  'Z';
     220        }
  0  221        else if (edge.Direction == Direction.Forward) {
  0  222          return  'z';
     223        }
  0  224        else {
  0  225          throw new IOException ("Failed to encode Edge: '" + edge + "'!");
     226        }
240  227      }
     228
     229      private static String EncodeColor (Color c)
240  230      {
240  231        return c.R.ToString ("X2") + c.G.ToString ("X2") + c.B.ToString ("X2") +
240  232      }
     233
     234      private static Color DecodeColor (string hexString)
  0  235      {
  0  236        if (hexString.StartsWith ("#")) {
  0  237          hexString = hexString.Substring (1);
  0  238        }
  0  239        uint hex = uint.Parse (hexString, System.Globalization.NumberStyles.HexN
  0  240        Color color = Color.White;
  0  241        if (hexString.Length == 8) {
  0  242          color.R = (byte)(hex >> 24);
  0  243          color.G = (byte)(hex >> 16);
  0  244          color.B = (byte)(hex >> 8);
  0  245          color.A = (byte)(hex);
  0  246        }
  0  247        else if (hexString.Length == 6) {
  0  248          color.R = (byte)(hex >> 16);
  0  249          color.G = (byte)(hex >> 8);
  0  250          color.B = (byte)(hex);
  0  251        }
  0  252        else {
  0  253          throw new IOException ("Invald hex representation of an ARGB or RGB co
     254        }
  0  255        return color;
  0  256      }
     257
     258      [ExcludeFromCodeCoverageAttribute]
     259      public override string ToString ()
     260      {
     261        return "KnotStringIO (length=" + Content.Length + ")";
     262      }
     263
     264      #endregion
     265    }
     266  }
```

Knot3.Data.Node

Summary

Class:	Knot3.Data.Node
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Node.cs
Coverage:	73.7%
Covered lines:	45
Uncovered lines:	16
Coverable lines:	61
Total lines:	211

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
op_Implicit(...)	1	100	100
CenterBetween(...)	1	100	100
op_Addition(...)	1	100	100
op_Subtraction(...)	1	100	100
op_Addition(...)	1	0	0
op_Subtraction(...)	1	0	0
op_Addition(...)	1	0	0
op_Subtraction(...)	1	0	0
Clone()	1	100	100
op_Equality(...)	6	55.56	54.55
op_Inequality(...)	1	100	100
Equals(...)	3	100	60
Equals(...)	2	71.43	66.67
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\Node.cs

#	Line	Coverage	
	1		#region Copyright
	2		
	3		/*
	4		* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5		* Gerd Augsburg, Christina Erler, Daniel Warzel
	6		*
	7		* Permission is hereby granted, free of charge, to any person obtaining a cop
	8		* of this software and associated documentation files (the "Software"), to de
	9		* in the Software without restriction, including without limitation the right
	10		* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11		* copies of the Software, and to permit persons to whom the Software is
	12		* furnished to do so, subject to the following conditions:
	13		*
	14		* The above copyright notice and this permission notice shall be included in
	15		* copies or substantial portions of the Software.
	16		*
	17		* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18		* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19		* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20		* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21		* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22		* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T

```
        23   * SOFTWARE.
        24   */
        25
        26  #endregion
        27
        28  #region Using
        29
        30  using System;
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.Linq;
        35
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Core;
        47  using Knot3.GameObjects;
        48  using Knot3.Input;
        49  using Knot3.RenderEffects;
        50  using Knot3.Screens;
        51  using Knot3.Widgets;
        52
        53  #endregion
        54
        55  namespace Knot3.Data
        56  {
        57    /// <summary>
        58    /// Eine Position im 3D-Raster. Die Werte fr alle drei Koordinaten sind Int
        59    /// Eine Skalierung auf Koordinaten im 3D-Raum und damit einhergehend eine K
        60    /// </summary>
        61    public sealed class Node : IEquatable<Node>, ICloneable
        62    {
        63      #region Properties
        64
        65      /// <summary>
        66      /// X steht fr eine x-Koordinate im dreidimensionalen Raster.
        67      /// </summary>
   152  68      public int X { get; private set; }
        69
        70      /// <summary>
        71      /// Y steht fr eine y-Koordinate im dreidimensionalen Raster.
        72      /// </summary>
   152  73      public int Y { get; private set; }
        74
        75      /// <summary>
        76      /// Z steht fr eine z-Koordinate im dreidimensionalen Raster.
        77      /// </summary>
   152  78      public int Z { get; private set; }
        79
        80      /// <summary>
        81      /// Ein Skalierungswert.
        82      /// </summary>
     1  83      public static readonly int Scale = 100;
```

```
                84
                85      #endregion
                86
                87      #region Constructors
                88
                89      /// <summary>
                90      /// Erzeugt eine neue Instanz eines Node-Objekts und initialisiert diese m
                91      /// fr die x-, y- und z-Koordinate.
                92      /// </summary>
    22          93      public Node (int x, int y, int z)
    22          94      {
    22          95        X = x;
    22          96        Y = y;
    22          97        Z = z;
    22          98      }
                99
               100      #endregion
               101
               102      #region Methods
               103
               104      /// <summary>
               105      /// Liefert die x-, y- und z-Koordinaten im 3D-Raum als ein Vektor3 der Fo
               106      /// </summary>
               107      public Vector3 Vector
               108      {
    22         109        get {
    22         110          return new Vector3 (X * Scale, Y * Scale, Z * Scale);
    22         111        }
               112      }
               113
               114      public static implicit operator Vector3 (Node node)
    14         115      {
    14         116        return node.Vector;
    14         117      }
               118
               119      public Vector3 CenterBetween (Node other)
     4         120      {
     4         121        Vector3 positionFrom = this.Vector;
     4         122        Vector3 positionTo = other.Vector;
     4         123        return positionFrom + (positionTo - positionFrom) / 2;
     4         124      }
               125
               126      public static Node operator + (Node a, Vector3 b)
     1         127      {
     1         128        return new Node (a.X + (int)b.X, a.Y + (int)b.Y, a.Z + (int)b.Z);
     1         129      }
               130
               131      public static Vector3 operator - (Node a, Node b)
     1         132      {
     1         133        return new Vector3 (a.X - b.X, a.Y - b.Y, a.Z - b.Z);
     1         134      }
               135
               136      public static Node operator + (Node a, Direction b)
     0         137      {
     0         138        return new Node (a.X + (int)b.Vector.X, a.Y + (int)b.Vector.Y, a.Z + (in
     0         139      }
               140
               141      public static Node operator - (Node a, Direction b)
     0         142      {
     0         143        return new Node (a.X - (int)b.Vector.X, a.Y - (int)b.Vector.Y, a.Z - (in
     0         144      }
```

```
        145
        146       public static Node operator + (Direction a, Node b)
    0   147       {
    0   148         return b+a;
    0   149       }
        150
        151       public static Node operator - (Direction a, Node b)
    0   152       {
    0   153         return b-a;
    0   154       }
        155
        156       [ExcludeFromCodeCoverageAttribute]
        157       public override int GetHashCode ()
        158       {
        159         return X * 10000 + Y * 100 + Z;
        160       }
        161
        162       [ExcludeFromCodeCoverageAttribute]
        163       public override string ToString ()
        164       {
        165         return "(" + X.ToString () + "," + Y.ToString () + "," + Z.ToString () +
        166       }
        167
        168       public object Clone ()
    1   169       {
    1   170         return new Node (X, Y, Z);
    1   171       }
        172
        173       public static bool operator == (Node a, Node b)
    2   174       {
        175         // If both are null, or both are same instance, return true.
    2   176         if (System.Object.ReferenceEquals (a, b)) {
    0   177           return true;
        178         }
        179
        180         // If one is null, but not both, return false.
    2   181         if (((object)a == null) || ((object)b == null)) {
    0   182           return false;
        183         }
        184
        185         // Return true if the fields match:
    2   186         return a.X == b.X && a.Y == b.Y && a.Z == b.Z;
    2   187       }
        188
        189       public static bool operator != (Node a, Node b)
    1   190       {
    1   191         return !(a == b);
    1   192       }
        193
        194       public bool Equals (Node other)
   32   195       {
   32   196         return this.X == other.X && this.Y == other.Y && this.Z == other.Z;
   32   197       }
        198
        199       public override bool Equals (object obj)
    2   200       {
    4   201         if (obj is Node) {
    2   202           return Equals ((Node)obj);
        203         }
    0   204         else {
    0   205           return false;
```

```
206         }
2   207     }
208
209     #endregion
210   }
211 }
```

Knot3.Data.NodeMap

Summary

Class:	Knot3.Data.NodeMap
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\NodeMap.cs
Coverage:	88.5%
Covered lines:	54
Uncovered lines:	7
Coverable lines:	61
Total lines:	176

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	2	100	66.67
.ctor(...)	1	100	100
NodeBeforeEdge(...)	1	100	100
NodeAfterEdge(...)	1	100	100
JunctionsAtNode(...)	1	100	100
JunctionsBeforeEdge(	1	100	100
JunctionsAfterEdge(.	1	100	100
OnEdgesChanged()	1	0	0
BuildIndex()	5	100	77.78

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\NodeMap.cs

```
#  Line  Coverage
      1  #region Copyright
      2
      3  /*
      4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
      5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
      6   *
      7   * Permission is hereby granted, free of charge, to any person obtaining a cop
      8   * of this software and associated documentation files (the "Software"), to de
      9   * in the Software without restriction, including without limitation the right
     10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
     11   * copies of the Software, and to permit persons to whom the Software is
     12   * furnished to do so, subject to the following conditions:
     13   *
     14   * The above copyright notice and this permission notice shall be included in
     15   * copies or substantial portions of the Software.
     16   *
     17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
     18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
     19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
     20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
     22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
     23   * SOFTWARE.
     24   */
     25
     26  #endregion
     27
     28  #region Using
```

```
        29
        30  using System;
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.Linq;
        35
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Core;
        47  using Knot3.GameObjects;
        48  using Knot3.Input;
        49  using Knot3.RenderEffects;
        50  using Knot3.Screens;
        51  using Knot3.Utilities;
        52  using Knot3.Widgets;
        53
        54  #endregion
        55
        56  namespace Knot3.Data
        57  {
        58    /// <summary>
        59    /// Eine Zuordnung zwischen Kanten und den dreidimensionalen Rasterpunkten,
        60    /// </summary>
        61    public sealed class NodeMap : INodeMap
        62    {
        63      #region Properties
        64
     1  65      private Hashtable fromMap = new Hashtable ();
     1  66      private Hashtable toMap = new Hashtable ();
     1  67      private Dictionary<Node, List<IJunction>> junctionMap = new Dictionary<Nod
        68
        69      /// <summary>
        70      /// Die Skalierung, die bei einer Konvertierung in einen Vector3 des XNA-F
        71      /// </summary>
     0  72      public int Scale { get; set; }
        73
    11  74      public IEnumerable<Edge> Edges { get; set; }
        75
     3  76      public Vector3 Offset { get; set; }
        77
    11  78      public Action IndexRebuilt { get; set; }
        79
        80      #endregion
        81
        82      #region Constructors
        83
     1  84      public NodeMap ()
     1  85      {
     1  86        IndexRebuilt = () => {};
     1  87      }
        88
     1  89      public NodeMap (IEnumerable<Edge> edges)
```

```
      90      : this ()
 1    91      {
 1    92        Edges = edges;
 1    93        BuildIndex ();
 1    94      }
      95
      96      #endregion
      97
      98      #region Methods
      99
     100      /// <summary>
     101      /// Gibt die Rasterposition des bergangs am Anfang der Kante zurck.
     102      /// </summary>
     103      public Node NodeBeforeEdge (Edge edge)
 8   104      {
 8   105        return (Node)fromMap [edge];
 8   106      }
     107
     108      /// <summary>
     109      /// Gibt die Rasterposition des bergangs am Ende der Kante zurck.
     110      /// </summary>
     111      public Node NodeAfterEdge (Edge edge)
32   112      {
32   113        return (Node)toMap [edge];
32   114      }
     115
     116      public List<IJunction> JunctionsAtNode (Node node)
16   117      {
16   118        return junctionMap [node];
16   119      }
     120
     121      public List<IJunction> JunctionsBeforeEdge (Edge edge)
 4   122      {
 4   123        return junctionMap [NodeBeforeEdge (edge)];
 4   124      }
     125
     126      public List<IJunction> JunctionsAfterEdge (Edge edge)
 4   127      {
 4   128        return junctionMap [NodeAfterEdge (edge)];
 4   129      }
     130
     131      public IEnumerable<Node> Nodes
     132      {
 0   133        get {
 0   134          return junctionMap.Keys;
 0   135        }
     136      }
     137
     138      /// <summary>
     139      /// Aktualisiert die Zuordnung, wenn sich die Kanten gendert haben.
     140      /// </summary>
     141      public void OnEdgesChanged ()
 0   142      {
 0   143        BuildIndex ();
 0   144      }
     145
     146      private void BuildIndex ()
 1   147      {
 1   148        fromMap.Clear ();
 1   149        toMap.Clear ();
 3   150        float x = Offset.X, y = Offset.Y, z = Offset.Z;
```

```
15  151          foreach (Edge edge in Edges) {
 4  152            fromMap [edge] = new Node ((int)x, (int)y, (int)z);
 4  153            Vector3 v = edge.Direction.Vector;
 4  154            x += v.X;
 4  155            y += v.Y;
 4  156            z += v.Z;
 4  157            toMap [edge] = new Node ((int)x, (int)y, (int)z);
 4  158          }
     159
 2  160          IndexRebuilt = () => {};
 1  161          junctionMap.Clear ();
 1  162          List<Edge> EdgeList = Edges.ToList ();
14  163          for (int n = 0; n < EdgeList.Count; n++) {
 4  164            Edge edgeA = Edges.At (n);
 4  165            Edge edgeB = Edges.At (n + 1);
 4  166            Node node = NodeAfterEdge (edgeA);
 4  167            IJunction junction = new NodeModelInfo (nodeMap: this, from: edgeA, to
 4  168            junctionMap.Add (node, junction);
 4  169          }
     170
 1  171          IndexRebuilt ();
 1  172        }
     173
     174        #endregion
     175      }
     176    }
```

Knot3.Data.PrinterIO

Summary

Class:	Knot3.Data.PrinterIO
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\PrinterIO.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	10
Coverable lines:	10
Total lines:	111

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	1	0	0
Save(...)	1	0	0
Load(...)	1	0	0
LoadMetaData(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\PrinterIO.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                          Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
      31  using System.Collections;
      32  using System.Collections.Generic;
      33  using System.Diagnostics.CodeAnalysis;
```

```
        34  using System.Linq;
        35
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Core;
        47  using Knot3.GameObjects;
        48  using Knot3.Input;
        49  using Knot3.RenderEffects;
        50  using Knot3.Screens;
        51  using Knot3.Widgets;
        52
        53  #endregion
        54
        55  namespace Knot3.Data
        56  {
        57    /// <summary>
        58    /// Ein Exportformat fr 3D-Drucker.
        59    /// </summary>
        60    public class PrinterIO : IKnotIO
        61    {
        62      #region Properties
        63
        64      /// <summary>
        65      /// Die gltigen Dateiendungen fr das 3D-Drucker-Format.
        66      /// </summary>
   0    67      public IEnumerable<string> FileExtensions { get; set; }
        68
        69      #endregion
        70
        71      #region Constructors
        72
        73      /// <summary>
        74      /// Erstellt ein neues PrinterIO-Objekt.
        75      /// </summary>
   0    76      public PrinterIO ()
   0    77      {
   0    78        throw new System.NotImplementedException ();
        79      }
        80
        81      #endregion
        82
        83      #region Methods
        84
        85      /// <summary>
        86      /// Exportiert den Knoten in einem gltigen 3D-Drucker-Format.
        87      /// </summary>
        88      public virtual void Save (Knot knot)
   0    89      {
   0    90        throw new System.NotImplementedException ();
        91      }
        92
        93      /// <summary>
        94      /// Gibt eine IOException zurck.
```

```
        95      /// </summary>
        96      public virtual Knot Load (string filename)
0       97      {
0       98        throw new System.NotImplementedException ();
        99      }
       100
       101      /// <summary>
       102      /// Gibt eine IOException zurck.
       103      /// </summary>
       104      public virtual KnotMetaData LoadMetaData (string filename)
0      105      {
0      106        throw new System.NotImplementedException ();
       107      }
       108
       109      #endregion
       110    }
       111  }
```

Knot3.Data.RectangleMap

Summary

Class:	Knot3.Data.RectangleMap
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\RectangleMap.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	53
Coverable lines:	53
Total lines:	173

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
AddEdge(...)	1	0	0
AddEdge(...)	3	0	0
ContainsEdge(...)	7	0	0
MoveNext()	13	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\RectangleMap.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;

```
      33  using System.Diagnostics.CodeAnalysis;
      34  using System.Linq;
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Core;
      47  using Knot3.GameObjects;
      48  using Knot3.Input;
      49  using Knot3.RenderEffects;
      50  using Knot3.Screens;
      51  using Knot3.Utilities;
      52  using Knot3.Widgets;
      53
      54  #endregion
      55
      56  namespace Knot3.Data
      57  {
      58    public sealed class RectangleMap
      59    {
      60      #region Properties
      61
      62      private INodeMap NodeMap;
   0  63      private Dictionary<Vector3, List<PossibleRectanglePosition>> positions
      64          = new Dictionary<Vector3, List<PossibleRectanglePosition>> ();
      65
      66      #endregion
      67
      68      #region Constructors
      69
   0  70      public RectangleMap (INodeMap nodeMap)
   0  71      {
   0  72        NodeMap = nodeMap;
   0  73      }
      74
      75      #endregion
      76
      77      #region Methods
      78
      79      public void AddEdge (Edge edge, bool isVirtual)
   0  80      {
   0  81        Node a = NodeMap.NodeBeforeEdge (edge);
   0  82        Node b = NodeMap.NodeAfterEdge (edge);
   0  83        AddEdge (edge, a, b, isVirtual);
   0  84      }
      85
      86      public void AddEdge (Edge edge, Node nodeA, Node nodeB, bool isVirtual)
   0  87      {
   0  88        Vector3 edgeCenter = nodeA.CenterBetween (nodeB);
   0  89        foreach (Direction direction in Direction.Values) {
   0  90          if (direction.Axis != edge.Direction.Axis) {
   0  91            Vector3 rectangleCenter = edgeCenter + direction * Node.Scale / 2;
   0  92            PossibleRectanglePosition rectanglePosition = new PossibleRectangleP
      93              Edge = edge,
```

```
      94              NodeA = nodeA,
      95              NodeB = nodeB,
      96              Position = rectangleCenter,
      97              IsVirtual = isVirtual
      98            };
0     99            positions.Add (rectangleCenter, rectanglePosition);
0    100          }
0    101        }
0    102      }
     103
     104      public bool ContainsEdge (Node a, Node b)
0    105      {
0    106        foreach (List<PossibleRectanglePosition> many in positions.Values) {
0    107          foreach (PossibleRectanglePosition position in many) {
0    108            if ((position.NodeA == a && position.NodeB == b) || (position.NodeA
0    109              return true;
     110            }
0    111          }
0    112        }
0    113        return false;
0    114      }
     115
     116      public IEnumerable<ValidRectanglePosition> ValidPositions ()
0    117      {
0    118        foreach (List<PossibleRectanglePosition> many in positions.Values) {
0    119          foreach (var pair in many.SelectMany ((value, index) => many.Skip (ind
0    120          (first, second) => new { first, second })) {
0    121            List<PossibleRectanglePosition> pos
     122                = new PossibleRectanglePosition[] { pair.first, pair.second } .T
0    123            if (pos.Count == 2) {
0    124              for (int i = 0; i <= 1; ++i) {
0    125                int first = i % 2;
0    126                int second = (i + 1) % 2;
0    127                Edge edgeAB = pos [first].Edge;
0    128                Edge edgeCD = pos [second].Edge;
0    129                Node nodeA = pos [first].NodeA;
0    130                Node nodeB = pos [first].NodeB;
0    131                Node nodeC = pos [second].NodeA;
0    132                Node nodeD = pos [second].NodeB;
0    133                if (nodeB == nodeC || (nodeA-nodeB) == (nodeC-nodeD)) {
0    134                  var valid = new ValidRectanglePosition {
     135                    EdgeAB = edgeAB,
     136                    EdgeCD = edgeCD,
     137                    NodeA = nodeA,
     138                    NodeB = nodeB,
     139                    NodeC = nodeC,
     140                    NodeD = nodeD,
     141                    Position = pos[first].Position,
     142                    IsVirtual = pos[first].IsVirtual || pos[second].IsVirtual
     143                  };
0    144                  yield return valid;
0    145                }
0    146              }
0    147            }
0    148          }
0    149        }
0    150      }
     151
     152      #endregion
     153    }
     154
```

```
155     public struct PossibleRectanglePosition {
156       public Edge Edge;
157       public Node NodeA;
158       public Node NodeB;
159       public Vector3 Position;
160       public bool IsVirtual;
161     }
162 
163     public struct ValidRectanglePosition {
164       public Edge EdgeAB;
165       public Edge EdgeCD;
166       public Node NodeA;
167       public Node NodeB;
168       public Node NodeC;
169       public Node NodeD;
170       public Vector3 Position;
171       public bool IsVirtual;
172     }
173   }
```

Knot3.Data.ZipHelper

Summary

Class:	Knot3.Data.ZipHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeFileIO.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	7
Coverable lines:	7
Total lines:	262

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
ReadContent(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\ChallengeFileIO.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.IO;
	35	using System.Linq;
	36	using System.Text;
	37	

```
38  using Microsoft.Xna.Framework;
39  using Microsoft.Xna.Framework.Audio;
40  using Microsoft.Xna.Framework.Content;
41  using Microsoft.Xna.Framework.GamerServices;
42  using Microsoft.Xna.Framework.Graphics;
43  using Microsoft.Xna.Framework.Input;
44  using Microsoft.Xna.Framework.Media;
45  using Microsoft.Xna.Framework.Net;
46  using Microsoft.Xna.Framework.Storage;
47
48  using Ionic.Zip;
49
50  using Knot3.Core;
51  using Knot3.Development;
52  using Knot3.GameObjects;
53  using Knot3.Input;
54  using Knot3.RenderEffects;
55  using Knot3.Screens;
56  using Knot3.Widgets;
57
58  #endregion
59
60  namespace Knot3.Data
61  {
62    /// <summary>
63    /// Implementiert das Speicherformat fr Challenges.
64    /// </summary>
65    public sealed class ChallengeFileIO : IChallengeIO
66    {
67      #region Properties
68
69      /// <summary>
70      /// Die fr eine Knoten-Datei gltigen Dateiendungen.
71      /// </summary>
72      public IEnumerable<string> FileExtensions
73      {
74        get {
75          yield return ".challenge";
76          yield return ".chl";
77          yield return ".chn";
78          yield return ".chg";
79          yield return ".chlng";
80        }
81      }
82
83      #endregion
84
85      #region Constructors
86
87      /// <summary>
88      /// Erstellt ein ChallengeFileIO-Objekt.
89      /// </summary>
90      public ChallengeFileIO ()
91      {
92      }
93
94      #endregion
95
96      #region Methods
97
98      /// <summary>
```

```
 99      /// Speichert eine Challenge in dem Dateinamen, der in dem Challenge-Objek
100      /// </summary>
101      public void Save (Challenge challenge)
102      {
103        using (ZipFile zip = new ZipFile ()) {
104          // Namen
105          zip.AddEntry ("name.txt", challenge.Name);
106          // Startknoten
107          KnotStringIO parser = new KnotStringIO (challenge.Start);
108          zip.AddEntry ("start.knot", parser.Content);
109          // Zielknoten
110          parser = new KnotStringIO (challenge.Target);
111          zip.AddEntry ("target.knot", parser.Content);
112          // Highscore
113          zip.AddEntry ("highscore.txt", string.Join ("\n", printHighscore (chal
114          // ZIP-Datei speichern
115          zip.Save (challenge.MetaData.Filename);
116        }
117      }
118
119      /// <summary>
120      /// Ldt eine Challenge aus einer angegebenen Datei.
121      /// </summary>
122      public Challenge Load (string filename)
123      {
124        ChallengeMetaData meta = LoadMetaData (filename: filename);
125        Knot start = null;
126        Knot target = null;
127
128        using (ZipFile zip = ZipFile.Read (filename)) {
129          foreach (ZipEntry entry in zip) {
130            string content = entry.ReadContent ();
131
132            // fr die Datei mit dem Startknoten
133            if (entry.FileName.ToLower ().Contains ("start")) {
134              KnotStringIO parser = new KnotStringIO (content: content);
135              start = new Knot (
136                  new KnotMetaData (parser.Name, () => parser.CountEdges, null,
137                  parser.Edges
138              );
139            }
140
141            // fr die Datei mit dem Zielknoten
142            else if (entry.FileName.ToLower ().Contains ("target")) {
143              KnotStringIO parser = new KnotStringIO (content: content);
144              target = new Knot (
145                  new KnotMetaData (parser.Name, () => parser.CountEdges, null,
146                  parser.Edges
147              );
148            }
149          }
150        }
151
152        if (meta != null && start != null && target != null) {
153          return new Challenge (meta, start, target);
154        }
155        else {
156          throw new IOException (
157              "Error! Invalid challenge file: " + filename
158              + " (meta=" + meta + ",start=" + start + ",target=" + target + ")"
159          );
```

```
160       }
161     }
162
163     /// <summary>
164     /// Ldt die Metadaten einer Challenge aus einer angegebenen Datei.
165     /// </summary>
166     public ChallengeMetaData LoadMetaData (string filename)
167     {
168       string name = null;
169       KnotMetaData start = null;
170       KnotMetaData target = null;
171       IEnumerable<KeyValuePair<string, int>> highscore = null;
172       using (ZipFile zip = ZipFile.Read (filename)) {
173         foreach (ZipEntry entry in zip) {
174           string content = entry.ReadContent ();
175
176           // fr die Datei mit dem Startknoten
177           if (entry.FileName.ToLower ().Contains ("start")) {
178             KnotStringIO parser = new KnotStringIO (content: content);
179             start = new KnotMetaData (parser.Name, () => parser.CountEdges, nu
180           }
181
182           // fr die Datei mit dem Zielknoten
183           else if (entry.FileName.ToLower ().Contains ("target")) {
184             KnotStringIO parser = new KnotStringIO (content: content);
185             target = new KnotMetaData (parser.Name, () => parser.CountEdges, n
186           }
187
188           // fr die Datei mit dem Namen
189           else if (entry.FileName.ToLower ().Contains ("name")) {
190             name = content.Trim ();
191           }
192
193           // fr die Datei mit den Highscores
194           else if (entry.FileName.ToLower ().Contains ("highscore")) {
195             highscore = parseHighscore (content.Split (new char[] {'\r','\n'},
196           }
197         }
198       }
199       if (name != null && start != null && target != null) {
200         Log.Debug ("Load challenge file: ", filename, " (name=", name, ",start
201         return new ChallengeMetaData (
202                     name: name,
203                     start: start,
204                     target: target,
205                     filename: filename,
206                     format: this,
207                     highscore: highscore
208                 );
209       }
210       else {
211         throw new IOException (
212             "Error! Invalid challenge file: " + filename
213             + " (name=" + name + ",start=" + start + ",target=" + target + ",h
214         );
215       }
216     }
217
218     IEnumerable<string> printHighscore (IEnumerable<KeyValuePair<string, int>>
219     {
220       foreach (KeyValuePair<string, int> entry in highscore) {
```

```
     221          Log.Debug (
     222              "Save Highscore: "
     223              + entry.Value.ToString ()
     224              + ":"
     225              + entry.Key.ToString ()
     226          );
     227
     228          yield return entry.Value + ":" + entry.Key;
     229        }
     230      }
     231
     232      IEnumerable<KeyValuePair<string, int>> parseHighscore (IEnumerable<string>
     233      {
     234        foreach (string line in highscore) {
     235          Log.Debug ("Load Highscore: ",line);
     236          if (line.Contains (":")) {
     237            string[] entry = line.Split (new char[] {':'}, 2, StringSplitOptions
     238            string name = entry [1].Trim ();
     239            int time;
     240            if (Int32.TryParse (entry [0], out time)) {
     241              Log.Debug ("=> ", name, ":", time);
     242              yield return new KeyValuePair<string, int> (name, time);
     243            }
     244          }
     245        }
     246      }
     247
     248      #endregion
     249    }
     250
     251    static class ZipHelper
     252    {
     253      public static string ReadContent (this ZipEntry entry)
   0 254      {
   0 255        MemoryStream memory = new MemoryStream ();
   0 256        entry.Extract (memory);
   0 257        memory.Position = 0;
   0 258        var sr = new StreamReader (memory);
   0 259        return sr.ReadToEnd ();
   0 260      }
     261    }
     262  }
```

Knot3.Platform.SystemInfo

Summary

Class:	Knot3.Platform.SystemInfo
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Platform\SystemInfo.cs \Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Platform\SystemInfo-XNA.cs
Coverage:	88.6%
Covered lines:	70
Uncovered lines:	9
Coverable lines:	79
Total lines:	261

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
IsRunningOnMono()	1	100	100
IsRunningOnMonogame(	1	100	100
IsRunningOnLinux()	1	100	100
IsRunningOnWindows()	1	100	100
findBaseDirectory()	3	100	100
.cctor()	1	100	100

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Platform\SystemInfo.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
```

```
      31  using System.Collections.Generic;
      32  using System.Diagnostics.CodeAnalysis;
      33  using System.IO;
      34  using System.Linq;
      35  using System.Security.Cryptography;
      36  using System.Text;
      37
      38  using Microsoft.Xna.Framework;
      39  using Microsoft.Xna.Framework.Audio;
      40  using Microsoft.Xna.Framework.Content;
      41  using Microsoft.Xna.Framework.GamerServices;
      42  using Microsoft.Xna.Framework.Graphics;
      43  using Microsoft.Xna.Framework.Input;
      44  using Microsoft.Xna.Framework.Media;
      45  using Microsoft.Xna.Framework.Net;
      46  using Microsoft.Xna.Framework.Storage;
      47
      48  using Knot3.Data;
      49  using Knot3.Development;
      50  using Knot3.GameObjects;
      51  using Knot3.RenderEffects;
      52  using Knot3.Screens;
      53  using Knot3.Utilities;
      54  using Knot3.Widgets;
      55
      56  #endregion
      57
      58  namespace Knot3.Platform
      59  {
      60    public static partial class SystemInfo
      61    {
      62      #region Properties
      63
      64      /// <summary>
      65      /// Das Einstellungsverzeichnis.
      66      /// </summary>
      67      public static string SettingsDirectory
      68      {
 35   69        get {
      70          string directory;
 35   71          if (SystemInfo.IsRunningOnLinux ()) {
  0   72            directory = Environment.GetEnvironmentVariable ("HOME") + "/.knot3/"
  0   73          }
 35   74          else {
 35   75            directory = Environment.GetFolderPath (System.Environment.SpecialFol
 35   76          }
 35   77          Directory.CreateDirectory (directory);
 35   78          return directory;
 35   79        }
      80      }
      81
      82      /// <summary>
      83      /// Das Spielstandverzeichnis.
      84      /// </summary>
      85      public static string SavegameDirectory
      86      {
 33   87        get {
 33   88          string directory = SettingsDirectory + "Savegames";
 33   89          Directory.CreateDirectory (directory);
 33   90          return directory;
 33   91        }
```

```
        92      }
        93
        94      /// <summary>
        95      /// Das Bildschirmfotoverzeichnis.
        96      /// </summary>
        97      public static string ScreenshotDirectory
        98      {
1       99        get {
       100          string directory;
1      101          if (SystemInfo.IsRunningOnLinux ()) {
0      102            directory = Environment.GetEnvironmentVariable ("HOME");
0      103          }
1      104          else {
1      105            directory = Environment.GetFolderPath (System.Environment.SpecialFol
1      106          }
1      107          Directory.CreateDirectory (directory);
1      108          return directory;
1      109        }
       110      }
       111
       112      public static string DecodedMusicCache
       113      {
7      114        get {
       115          string directory;
7      116          if (SystemInfo.IsRunningOnLinux ()) {
0      117            directory = "/var/tmp/knot3/";
0      118          }
7      119          else {
7      120            directory = Environment.GetFolderPath (System.Environment.SpecialFol
7      121          }
7      122          Directory.CreateDirectory (directory);
7      123          return directory;
7      124        }
       125      }
       126
       127      public static string BaseDirectory
       128      {
2      129        get {
4      130          if (baseDirectory != null) {
2      131            return baseDirectory;
       132          }
0      133          else {
0      134            findBaseDirectory ();
0      135            return baseDirectory;
       136          }
2      137        }
       138      }
       139
       140      public static string RelativeBaseDirectory
       141      {
3      142        get {
5      143          if (relativeBaseDirectory != null) {
2      144            return relativeBaseDirectory;
       145          }
1      146          else {
1      147            findBaseDirectory ();
1      148            return relativeBaseDirectory;
       149          }
3      150        }
       151      }
       152
```

```
      153      private static void findBaseDirectory ()
    1 154      {
    1 155        string baseDir = Directory.GetCurrentDirectory ();
    1 156        string relBaseDir = "." + PathSeparator;
    1 157        string[] binDirectories = new string[] {
      158          "Debug",
      159          "Release",
      160          "x86",
      161          "bin"
      162        };
   15 163        foreach (string dir in binDirectories) {
    6 164          if (baseDir.ToLower ().EndsWith (dir.ToLower ())) {
    2 165            baseDir = baseDir.Substring (0, baseDir.Length - dir.Length - 1);
    2 166            relBaseDir += ".." + PathSeparator;
    2 167          }
    4 168        }
    1 169        Log.Debug ("Base directory: ", baseDir);
    1 170        baseDirectory = baseDir;
    1 171        Log.Debug ("Base directory (relative): ", relBaseDir);
    1 172        relativeBaseDirectory = relBaseDir;
    1 173      }
      174
    1 175      private static string relativeBaseDirectory = null;
    1 176      private static string baseDirectory = null;
    1 177      public readonly static char PathSeparator = Path.DirectorySeparatorChar;
      178
      179      #endregion
      180    }
      181  }
```

\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Platform\SystemInfo-XNA.cs

```
#  Line  Coverage
      1  #region Copyright
      2
      3  /*
      4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
      5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
      6   *
      7   * Permission is hereby granted, free of charge, to any person obtaining a cop
      8   * of this software and associated documentation files (the "Software"), to de
      9   * in the Software without restriction, including without limitation the right
     10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
     11   * copies of the Software, and to permit persons to whom the Software is
     12   * furnished to do so, subject to the following conditions:
     13   *
     14   * The above copyright notice and this permission notice shall be included in
     15   * copies or substantial portions of the Software.
     16   *
     17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
     18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
     19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
     20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
     22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
     23   * SOFTWARE.
     24   */
     25
     26  #endregion
     27
     28  #region Using
```

```
        29
        30  using System;
        31  using System.Collections;
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.Linq;
        35
        36  using Microsoft.Xna.Framework;
        37  using Microsoft.Xna.Framework.Audio;
        38  using Microsoft.Xna.Framework.Content;
        39  using Microsoft.Xna.Framework.GamerServices;
        40  using Microsoft.Xna.Framework.Graphics;
        41  using Microsoft.Xna.Framework.Input;
        42  using Microsoft.Xna.Framework.Media;
        43  using Microsoft.Xna.Framework.Net;
        44  using Microsoft.Xna.Framework.Storage;
        45
        46  using Knot3.Core;
        47  using Knot3.Data;
        48  using Knot3.GameObjects;
        49  using Knot3.Input;
        50  using Knot3.RenderEffects;
        51  using Knot3.Screens;
        52  using Knot3.Widgets;
        53
        54  #endregion
        55
        56  namespace Knot3.Platform
        57  {
        58    public static partial class SystemInfo
        59    {
        60      public static bool IsRunningOnMono ()
   1    61      {
   1    62        return false;
   1    63      }
        64
        65      public static bool IsRunningOnMonogame ()
   1    66      {
   1    67        return false;
   1    68      }
        69
        70      public static bool IsRunningOnLinux ()
 104    71      {
 104    72        return false;
 104    73      }
        74
        75      public static bool IsRunningOnWindows ()
   1    76      {
   1    77        return true;
   1    78      }
        79    }
        80  }
```

Knot3.Utilities.BoundingCylinder

Summary

Class:	Knot3.Utilities.BoundingCylinder
Assembly:	Knot3
File(s):	ers\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\BoundingCylinder.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	21
Coverable lines:	21
Total lines:	87

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
op_Equality(...)	4	0	0
op_Inequality(...)	1	0	0
Equals(...)	3	0	0
Equals(...)	2	0	0

File(s)

ers\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\BoundingCylinder.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections.Generic;`
	32	`using System.Diagnostics.CodeAnalysis;`

```
      33  using System.Linq;
      34  using System.Text;
      35
      36  using Microsoft.Xna.Framework;
      37
      38  #endregion
      39
      40  namespace Knot3.Utilities
      41  {
      42    public struct BoundingCylinder : IEquatable<BoundingCylinder> {
      43      public Vector3 SideA;
      44      public Vector3 SideB;
      45      public float Radius;
      46
      47      public BoundingCylinder (Vector3 sideA, Vector3 sideB, float radius)
  0   48      {
  0   49        this.SideA = sideA;
  0   50        this.SideB = sideB;
  0   51        this.Radius = radius;
  0   52      }
      53
      54      public static bool operator == (BoundingCylinder a, BoundingCylinder b)
  0   55      {
  0   56        if (System.Object.ReferenceEquals (a, b)) {
  0   57          return true;
      58        }
  0   59        if (((object)a == null) || ((object)b == null)) {
  0   60          return false;
      61        }
  0   62        return a.Equals (b);
  0   63      }
      64
      65      public static bool operator != (BoundingCylinder a, BoundingCylinder b)
  0   66      {
  0   67        return !(a == b);
  0   68      }
      69
      70      public bool Equals (BoundingCylinder other)
  0   71      {
  0   72        return SideA == other.SideA && SideB == other.SideB && Radius == other.R
  0   73      }
      74
      75      public override bool Equals (object other)
  0   76      {
  0   77        return other != null && Equals ((BoundingCylinder)other);
  0   78      }
      79
      80      [ExcludeFromCodeCoverageAttribute]
      81      public override int GetHashCode ()
      82      {
      83        // irgendwas mglichst eindeutiges
      84        return (Radius * (SideA + SideB)).GetHashCode ();
      85      }
      86    }
      87  }
```

Knot3.Utilities.FileIndex

Summary

Class:	Knot3.Utilities.FileIndex
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FileIndex.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	24
Coverable lines:	24
Total lines:	100

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
Add(...)	1	0	0
Remove(...)	1	0	0
Contains(...)	1	0	0
Save()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FileIndex.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
      22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
      23   * SOFTWARE.
      24   */
      25
      26  #endregion
      27
      28  #region Using
      29
      30  using System;
      31  using System.Collections;
      32  using System.Collections.Generic;
```

```
        33  using System.ComponentModel;
        34  using System.Diagnostics.CodeAnalysis;
        35  using System.IO;
        36  using System.Linq;
        37  using System.Reflection;
        38  using System.Text;
        39
        40  using Microsoft.Xna.Framework;
        41  using Microsoft.Xna.Framework.Audio;
        42  using Microsoft.Xna.Framework.Content;
        43  using Microsoft.Xna.Framework.GamerServices;
        44  using Microsoft.Xna.Framework.Graphics;
        45  using Microsoft.Xna.Framework.Input;
        46  using Microsoft.Xna.Framework.Media;
        47  using Microsoft.Xna.Framework.Net;
        48  using Microsoft.Xna.Framework.Storage;
        49
        50  using Knot3.Core;
        51  using Knot3.Data;
        52  using Knot3.GameObjects;
        53  using Knot3.Input;
        54  using Knot3.RenderEffects;
        55  using Knot3.Screens;
        56  using Knot3.Widgets;
        57
        58  #endregion
        59
        60  namespace Knot3.Utilities
        61  {
        62    public class FileIndex
        63    {
        64      private HashSet<string> hashes;
        65      private string filename;
        66
    0   67      public FileIndex (string filename)
    0   68      {
    0   69        this.filename = filename;
    0   70        try {
    0   71          hashes = new HashSet<string> (FileUtility.ReadFrom (filename));
    0   72        }
    0   73        catch (IOException) {
    0   74          hashes = new HashSet<string> ();
    0   75        }
    0   76      }
        77
        78      public void Add (string hash)
    0   79      {
    0   80        hashes.Add (hash);
    0   81        Save ();
    0   82      }
        83
        84      public void Remove (string hash)
    0   85      {
    0   86        hashes.Remove (hash);
    0   87        Save ();
    0   88      }
        89
        90      public bool Contains (string hash)
    0   91      {
    0   92        return hashes.Contains (hash);
    0   93      }
```

```
        94
        95      private void Save ()
 0      96      {
 0      97        File.WriteAllText (filename, string.Join ("\n", hashes));
 0      98      }
        99    }
       100  }
```

Knot3.Utilities.FileUtility

Summary

Class:	Knot3.Utilities.FileUtility
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FileUtility.cs
Coverage:	36.1%
Covered lines:	13
Uncovered lines:	23
Coverable lines:	36
Total lines:	130

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
ConvertToFileName(..	2	100	100
GetHash(...)	1	0	0
ToMD5Hash(...)	2	0	0
SearchFiles(...)	3	0	0
SearchFiles(...)	3	100	80
MoveNext()	5	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FileUtility.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections;`

```
        32  using System.Collections.Generic;
        33  using System.Diagnostics.CodeAnalysis;
        34  using System.IO;
        35  using System.Linq;
        36  using System.Security.Cryptography;
        37  using System.Text;
        38
        39  using Microsoft.Xna.Framework;
        40  using Microsoft.Xna.Framework.Audio;
        41  using Microsoft.Xna.Framework.Content;
        42  using Microsoft.Xna.Framework.GamerServices;
        43  using Microsoft.Xna.Framework.Graphics;
        44  using Microsoft.Xna.Framework.Input;
        45  using Microsoft.Xna.Framework.Media;
        46  using Microsoft.Xna.Framework.Net;
        47  using Microsoft.Xna.Framework.Storage;
        48
        49  using Knot3.Data;
        50  using Knot3.Development;
        51  using Knot3.GameObjects;
        52  using Knot3.RenderEffects;
        53  using Knot3.Screens;
        54  using Knot3.Widgets;
        55
        56  #endregion
        57
        58  namespace Knot3.Utilities
        59  {
        60    /// <summary>
        61    /// Eine Hilfsklasse fr Dateioperationen.
        62    /// </summary>
        63    public static class FileUtility
        64    {
        65      #region Methods
        66
        67      /// <summary>
        68      /// Konvertiert einen Namen eines Knotens oder einer Challenge in einen g
        69      /// </summary>
        70      public static string ConvertToFileName (string name)
32      71      {
32      72        char[] arr = name.ToCharArray ();
182     73        arr = Array.FindAll<char> (arr, (c => (char.IsLetterOrDigit (c)
        74                                               || char.IsWhiteSpace (c)
        75                                               || c == '-'))
        76                                   );
32      77        return new string (arr);
32      78      }
        79
        80      /// <summary>
        81      /// Liefert einen Hash-Wert zu der durch filename spezifizierten Datei.
        82      /// </summary>
        83      public static string GetHash (string filename)
0       84      {
0       85        return string.Join ("\n", FileUtility.ReadFrom (filename)).ToMD5Hash ();
0       86      }
        87
        88      public static string ToMD5Hash (this string TextToHash)
0       89      {
0       90        if (string.IsNullOrEmpty (TextToHash)) {
0       91          return string.Empty;
        92        }
```

```
        93
  0     94        MD5 md5 = new MD5CryptoServiceProvider ();
  0     95        byte[] textToHash = Encoding.Default.GetBytes (TextToHash);
  0     96        byte[] result = md5.ComputeHash (textToHash);
        97
  0     98        return System.BitConverter.ToString (result);
  0     99      }
       100
       101      public static IEnumerable<string> ReadFrom (string file)
  0    102      {
       103        string line;
  0    104        using (var reader = File.OpenText (file)) {
  0    105          while ((line = reader.ReadLine ()) != null) {
  0    106            yield return line;
  0    107          }
  0    108        }
  0    109      }
       110
       111      public static void SearchFiles (IEnumerable<string> directories, IEnumerab
  0    112      {
  0    113        foreach (string directory in directories) {
  0    114          SearchFiles (directory, extensions, add);
  0    115        }
  0    116      }
       117
       118      public static void SearchFiles (string directory, IEnumerable<string> exte
  6    119      {
  6    120        Directory.CreateDirectory (directory);
  6    121        var files = Directory.GetFiles (directory, "*.*", SearchOption.AllDirect
 36    122                    .Where (s => extensions.Any (e => s.EndsWith (e)));
 36    123        foreach (string file in files) {
  6    124          add (file);
  6    125        }
  6    126      }
       127
       128      #endregion
       129    }
       130  }
```

Knot3.Utilities.IniFile

Summary

Class:	Knot3.Utilities.IniFile
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\IniFile.cs
Coverage:	81.1%
Covered lines:	56
Uncovered lines:	13
Coverable lines:	69
Total lines:	130

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	9	100	70.59
Dispose()	1	0	0
Dispose(...)	2	0	0
Save()	8	100	80
StripComments(...)	3	66.67	60

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\IniFile.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections.Generic;`
	32	`using System.Diagnostics.CodeAnalysis;`

```
                33  using System.Globalization;
                34  using System.IO;
                35  using System.Linq;
                36
                37  #endregion
                38
                39  namespace Knot3.Utilities
                40  {
                41    public sealed class IniFile : IDisposable
                42    {
                43      private string Filename;
                44      public Dictionary<string, Dictionary<string, string>> Data;
                45
           3    46      public IniFile (string filename)
           3    47      {
           3    48        Data = new Dictionary<string, Dictionary<string, string>> ();
           3    49        Filename = filename;
           6    50        if (File.Exists (filename)) {
           6    51          using (StreamReader reader = new StreamReader (filename)) {
           3    52            string section = null;
        2705    53            while (reader.Peek () != -1) {
        1351    54              string line = StripComments (reader.ReadLine ().Trim ());
        1702    55              if (line.StartsWith ("[") && line.EndsWith ("]")) {
         351    56                section = line.Substring (1, line.Length - 2);
         702    57                if (!Data.ContainsKey (section)) {
         351    58                  Data [section] = new Dictionary<string,string> ();
         351    59                }
         351    60              }
        2000    61              else if (line.Contains ("=")) {
        1000    62                string[] parts = line.Split ('=');
        2000    63                if (section != null) {
        1000    64                  Data [section] [parts [0].Trim ()] = parts [1].Trim ();
        1000    65                }
        1000    66              }
        1351    67            }
           3    68          }
           3    69        }
           3    70      }
                71
                72      public void Dispose ()
           0    73      {
           0    74        Dispose (true);
           0    75        GC.SuppressFinalize (this);
           0    76      }
                77
                78      private void Dispose (bool disposing)
           0    79      {
           0    80        if (disposing) {
           0    81          Save ();
           0    82        }
           0    83      }
                84
                85      public void Save ()
         404    86      {
         808    87        using (StreamWriter writer = new StreamWriter (Filename)) {
       13708    88          foreach (string section in Data.Keys.OrderBy (x => x)) {
        3124    89            writer.WriteLine ("[" + section + "]");
      132308    90            foreach (string key  in Data[section].Keys.OrderBy (x => x)) {
       30734    91              writer.WriteLine (key + "=" + Data [section] [key]);
       30734    92            }
        3124    93          }
```

```
404   94        }
404   95      }
      96
      97      private static string StripComments (string line)
1351  98      {
2702  99        if (line != null) {
1351 100          if (line.IndexOf (';') != -1) {
   0 101            return line.Remove (line.IndexOf (';')).Trim ();
     102          }
1351 103          return line.Trim ();
     104        }
   0 105        return string.Empty;
1351 106      }
     107
     108      public string this [string section, string key, string defaultValue = null
     109      {
 525 110        get {
 535 111          if (!Data.ContainsKey (section)) {
  10 112            Data [section] = new Dictionary<string,string> ();
  10 113          }
 625 114          if (!Data [section].ContainsKey (key)) {
 100 115            Data [section] [key] = defaultValue;
 100 116            Save ();
 100 117          }
 525 118          string value = Data [section] [key];
 525 119          return value;
 525 120        }
 304 121        set {
 304 122          if (!Data.ContainsKey (section)) {
   0 123            Data [section] = new Dictionary<string,string> ();
   0 124          }
 304 125          Data [section] [key] = value;
 304 126          Save ();
 304 127        }
     128      }
     129    }
     130  }
```

Knot3.Utilities.RayExtensions

Summary

Class:	Knot3.Utilities.RayExtensions
Assembly:	Knot3
File(s):	\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\RayExtensions.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	50
Coverable lines:	50
Total lines:	147

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
Intersects(...)	21	0	0

File(s)

\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\RayExtensions.cs

#	Line	Coverage
	1	`#region Copyright`
	2	
	3	`/*`
	4	` * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,`
	5	` *                         Gerd Augsburg, Christina Erler, Daniel Warzel`
	6	` *`
	7	` * Permission is hereby granted, free of charge, to any person obtaining a cop`
	8	` * of this software and associated documentation files (the "Software"), to de`
	9	` * in the Software without restriction, including without limitation the right`
	10	` * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
	11	` * copies of the Software, and to permit persons to whom the Software is`
	12	` * furnished to do so, subject to the following conditions:`
	13	` *`
	14	` * The above copyright notice and this permission notice shall be included in`
	15	` * copies or substantial portions of the Software.`
	16	` *`
	17	` * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
	18	` * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
	19	` * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
	20	` * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
	21	` * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO`
	22	` * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T`
	23	` * SOFTWARE.`
	24	` */`
	25	
	26	`#endregion`
	27	
	28	`#region Using`
	29	
	30	`using System;`
	31	`using System.Collections.Generic;`
	32	`using System.Diagnostics.CodeAnalysis;`
	33	`using System.Linq;`
	34	`using System.Text;`
	35	
	36	`using Microsoft.Xna.Framework;`
	37	

```
        38  #endregion
        39
        40  namespace Knot3.Utilities
        41  {
        42    public static class RayExtensions
        43    {
        44      public static float? Intersects (this Ray ray, BoundingCylinder cylinder)
 0      45      {
 0      46        Vector3 dirAB = cylinder.SideB - cylinder.SideA;
        47        // Raystart innerhalb des Zylinders
 0      48        if (Vector3.Cross ((ray.Position - cylinder.SideA), ray.Direction).Lengt
 0      49          return 0.0f;
        50        }
 0      51        Vector3 perpendicular = Vector3.Cross (dirAB, ray.Direction);
        52        // if !(Ray Parallel zum Zylinder)
 0      53        if (perpendicular.Length () > 0.0000001f) {
 0      54          perpendicular.Normalize ();
 0      55          if (Vector3.Dot (perpendicular, ray.Direction) > 0) {
 0      56            perpendicular = -perpendicular;
 0      57          }
 0      58          Vector3 perpendicular2 = Vector3.Cross (dirAB, perpendicular);
        59          // If (Ray Senkrecht zum Zylinder)
 0      60          if (perpendicular2.Length () < 0.0000001f) {
 0      61            if (Vector3.Dot (dirAB, ray.Position - cylinder.SideA) < 0 || Vector
 0      62              return null;
        63            }
 0      64            float? result = Vector3.Cross ((ray.Position - cylinder.SideA), ray.
 0      65            if (result < 0) {
 0      66              result = 0.0f;
 0      67            }
 0      68            return result;
        69          }
 0      70          if (Vector3.Dot (perpendicular2, ray.Direction) > 0) {
 0      71            perpendicular2 = -perpendicular2;
 0      72          }
 0      73          perpendicular2.Normalize ();
 0      74          float minDist = Math.Abs (Vector3.Dot (cylinder.SideA - ray.Position,
 0      75          if (minDist > cylinder.Radius) {
 0      76            return null;
        77          }
 0      78          Vector3 plainNorm = perpendicular * minDist + (float)Math.Sqrt (cylind
 0      79          plainNorm.Normalize ();
 0      80          float? other_result = ray.Intersects (new Plane (plainNorm, Vector3.Do
 0      81          if (other_result == null) {
 0      82            return null;
        83          }
 0      84          Vector3 cutA = ray.Position + ray.Direction * (float)other_result - cy
 0      85          Vector3 cutB = ray.Position + ray.Direction * (float)other_result - cy
 0      86          if (Vector3.Dot (dirAB, cutA) > 0 && Vector3.Dot (-dirAB, cutB) > 0) {
 0      87            return other_result;
        88          }
 0      89        }
 0      90        if (Vector3.Distance (ray.Position, cylinder.SideA) < Vector3.Distance (
 0      91          dirAB.Normalize ();
 0      92          float? result = ray.Intersects (new Plane (dirAB, Vector3.Dot (dirAB,
 0      93          if (result == null || Vector3.Distance (ray.Position + ray.Direction *
 0      94            return null;
        95          }
 0      96          return result;
        97        }
 0      98        else {
```

```
0    99          dirAB.Normalize ();
0   100          dirAB = -dirAB;
0   101          float? result = ray.Intersects (new Plane (dirAB, Vector3.Dot (dirAB,
0   102          if (result == null || Vector3.Distance (ray.Position + ray.Direction *
0   103            return null;
    104          }
0   105          return result;
    106        }
    107        /*
    108        Vector3 diffA = capsule.CornerA - ray.Position;
    109        Vector3 diffB = capsule.CornerB - ray.Position;
    110        float diffASquared = diffA.LengthSquared ();
    111        float diffBSquared = diffB.LengthSquared ();
    112        float radiusSquared = capsule.Radius * capsule.Radius;
    113        // Startpunkt innerhalb der Eckkugeln
    114        if (diffASquared < radiusSquared || diffBSquared < radiusSquared)
    115        {
    116            return 0.0f;
    117        }
    118        Vector3 dirBA = (capsule.CornerA - capsule.CornerB);
    119        float distAlongAB = Vector3.Dot (diffA, dirBA) / dirBA.Length ();
    120        // Startpunkt innerhalb des Zylinders
    121        if (distAlongAB > 0 && distAlongAB < dirBA.Length () && (distAlongAB * d
    122        {
    123            return 0.0f;
    124        }
    125        float distAlongRayA = Vector3.Dot (ray.Direction, diffA);
    126        float distAlongRayB = Vector3.Dot (ray.Direction, diffB);
    127        // Richtung geht weg von der Kapsel
    128        if (distAlongRayA < 0 && distAlongRayB < 0)
    129            return null;
    130        Vector3 perpendicular = Vector3.Cross (ray.Direction, dirBA);
    131        perpendicular.Normalize ();
    132        float minDistance = Math.Abs (Vector3.Dot (diffA, perpendicular));
    133        // Kommt selbst der Geraden nie nahe genug.
    134        if (minDistance > capsule.Radius)
    135        {
    136            return null;
    137        }
    138        Vector3 normDirAB = -dirBA;
    139        normDirAB.Normalize ();
    140        Vector3 extensionToBase = Vector3.Cross (normDirAB, perpendicular);
    141        extensionToBase.Normalize ();
    142        Matrix transformation = new Matrix (normDirAB.X, normDirAB.Y, normDirAB.
    143        transformation = Matrix.Invert (transformation);
    144         */
0   145      }
    146    }
    147  }
```

Knot3.Utilities.SavegameLoader‘2

Summary

Class:	Knot3.Utilities.SavegameLoader‘2
Assembly:	Knot3
File(s):	Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\SavegameLoader.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	31
Coverable lines:	31
Total lines:	130

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
FindSavegames(...)	1	0	0
AddFileToList(...)	3	0	0

File(s)

Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\SavegameLoader.cs

#	Line	Coverage
	1	#region Copyright
	2	
	3	/*
	4	* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5	* Gerd Augsburg, Christina Erler, Daniel Warzel
	6	*
	7	* Permission is hereby granted, free of charge, to any person obtaining a cop
	8	* of this software and associated documentation files (the "Software"), to de
	9	* in the Software without restriction, including without limitation the right
	10	* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11	* copies of the Software, and to permit persons to whom the Software is
	12	* furnished to do so, subject to the following conditions:
	13	*
	14	* The above copyright notice and this permission notice shall be included in
	15	* copies or substantial portions of the Software.
	16	*
	17	* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
	18	* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
	19	* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
	20	* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
	21	* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
	22	* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
	23	* SOFTWARE.
	24	*/
	25	
	26	#endregion
	27	
	28	#region Using
	29	
	30	using System;
	31	using System.Collections;
	32	using System.Collections.Generic;
	33	using System.Diagnostics.CodeAnalysis;
	34	using System.Linq;

```
      35
      36  using Microsoft.Xna.Framework;
      37  using Microsoft.Xna.Framework.Audio;
      38  using Microsoft.Xna.Framework.Content;
      39  using Microsoft.Xna.Framework.GamerServices;
      40  using Microsoft.Xna.Framework.Graphics;
      41  using Microsoft.Xna.Framework.Input;
      42  using Microsoft.Xna.Framework.Media;
      43  using Microsoft.Xna.Framework.Net;
      44  using Microsoft.Xna.Framework.Storage;
      45
      46  using Knot3.Core;
      47  using Knot3.Data;
      48  using Knot3.Development;
      49  using Knot3.GameObjects;
      50  using Knot3.Input;
      51  using Knot3.Platform;
      52  using Knot3.RenderEffects;
      53  using Knot3.Screens;
      54  using Knot3.Widgets;
      55
      56  #endregion
      57
      58  namespace Knot3.Utilities
      59  {
      60    public class SavegameLoader<Savegame, SavegameMetaData>
      61    {
      62      public ISavegameIO<Savegame, SavegameMetaData> FileFormat { get; set; }
      63
      64      public FileIndex fileIndex { get; private set; }
      65
      66      public string IndexName;
      67      private Action<string, SavegameMetaData> OnSavegameFound;
      68
  0   69      public SavegameLoader (ISavegameIO<Savegame, SavegameMetaData> fileFormat,
  0   70      {
  0   71        FileFormat = fileFormat;
  0   72        IndexName = indexName;
  0   73      }
      74
      75      public void FindSavegames (Action<string, SavegameMetaData> onSavegameFoun
  0   76      {
      77        // Erstelle einen neuen Index, der eine Datei mit dem angegeben Indexnam
  0   78        fileIndex = new FileIndex (SystemInfo.SavegameDirectory + SystemInfo.Pat
      79
      80        // Diese Verzeichnisse werden nach Spielstnden durchsucht
  0   81        string[] searchDirectories = new string[] {
      82          SystemInfo.BaseDirectory,
      83          SystemInfo.SavegameDirectory
      84        };
  0   85        Log.Debug ("Search for Savegames: ", string.Join (", ", searchDirectorie
      86
      87        // Suche nach Spielstanddateien und flle das Men auf
  0   88        OnSavegameFound = onSavegameFound;
  0   89        FileUtility.SearchFiles (searchDirectories, FileFormat.FileExtensions, A
  0   90      }
      91
      92      /// <summary>
      93      /// Diese Methode wird fr jede gefundene Spielstanddatei aufgerufen
      94      /// </summary>
      95      private void AddFileToList (string filename)
```

```
0     96      {
      97        // Lese die Datei ein und erstelle einen Hashcode
0     98        string hashcode = FileUtility.GetHash (filename);
      99
     100        // Ist dieser Hashcode im Index enthalten?
     101        // Dann wre der Spielstand gltig, sonst ungltig oder unbekannt.
0    102        bool isValid = fileIndex.Contains (hashcode);
     103
     104        // Wenn der Spielstand ungltig oder unbekannt ist...
0    105        if (!isValid) {
0    106          try {
     107            // Lade den Knoten und prfe, ob Exceptions auftreten
0    108            FileFormat.Load (filename);
     109            // Keine Exceptions? Dann ist enthlt die Datei einen gltigen Knote
0    110            isValid = true;
0    111            fileIndex.Add (hashcode);
0    112          }
0    113          catch (Exception ex) {
     114            // Es ist eine Exception aufgetreten, der Knoten ist offenbar unglt
0    115            Log.Debug (ex);
0    116            isValid = false;
0    117          }
0    118        }
     119
     120        // Falls der Knoten gltig ist, entweder laut Index oder nach berprfun
0    121        if (isValid) {
     122          // Lade die Metadaten
0    123          SavegameMetaData meta = FileFormat.LoadMetaData (filename);
     124
     125          // Rufe die Callback-Funktion auf
0    126          OnSavegameFound (filename, meta);
0    127        }
0    128      }
     129    }
     130  }
```

Knot3.Widgets.Bounds

Summary

Class:	Knot3.Widgets.Bounds
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Bounds.cs
Coverage:	83.5%
Covered lines:	66
Uncovered lines:	13
Coverable lines:	79
Total lines:	234

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	100	100
.ctor(...)	1	100	100
Contains(...)	1	100	100
Contains(...)	1	100	100
Zero(...)	1	100	100
FromLeft(...)	1	100	100
FromRight(...)	2	100	66.67
FromTop(...)	1	100	100
FromBottom(...)	2	100	66.67
FromLeft(...)	1	100	100
FromRight(...)	1	100	100
FromTop(...)	1	100	100
FromBottom(...)	1	100	100
In(...)	1	0	0
op_Implicit(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\Bounds.cs

```
#  Line  Coverage
       1  #region Copyright
       2
       3  /*
       4   * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
       5   *                         Gerd Augsburg, Christina Erler, Daniel Warzel
       6   *
       7   * Permission is hereby granted, free of charge, to any person obtaining a cop
       8   * of this software and associated documentation files (the "Software"), to de
       9   * in the Software without restriction, including without limitation the right
      10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
      11   * copies of the Software, and to permit persons to whom the Software is
      12   * furnished to do so, subject to the following conditions:
      13   *
      14   * The above copyright notice and this permission notice shall be included in
      15   * copies or substantial portions of the Software.
      16   *
      17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
      18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
      19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
      20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
```

```
22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
23   * SOFTWARE.
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37
38  using Knot3.Core;
39  using Knot3.Input;
40  using Knot3.Utilities;
41
42  #endregion
43
44  namespace Knot3.Widgets
45  {
46    public class Bounds
47    {
48      #region Properties
49
50      /// <summary>
51      /// Die von der Auflsung unabhngige Position in Prozent.
52      /// </summary>
53      public ScreenPoint Position
54      {
66    55        get { return _position; }
 0    56        set { _position.Assign (value); }
57      }
58
59      private ScreenPoint _position;
60
61      /// <summary>
62      /// Die von der Auflsung unabhngige Gre in Prozent.
63      /// </summary>
64      public ScreenPoint Size
65      {
126   66        get { return _size; }
 3    67        set { _size.Assign (value); }
68      }
69
70      private ScreenPoint _size;
71
72      /// <summary>
73      /// Der von der Auflsung unabhngige Abstand in Prozent.
74      /// </summary>
75      public ScreenPoint Padding
76      {
24    77        get { return _padding; }
 0    78        set { _padding.Assign (value); }
79      }
80
81      private ScreenPoint _padding;
82
```

```
        83      /// <summary>
        84      /// Gibt ein auf die Auflsujng skaliertes Rechteck zurck, das in den XNA
        85      /// </summary>
        86      public Rectangle Rectangle
        87      {
 2      88        get {
 2      89          Point pos = Position.Absolute;
 2      90          Point size = Size.Absolute;
 2      91          return new Rectangle (pos.X, pos.Y, size.X, size.Y);
 2      92        }
        93      }
        94
        95      public Vector4 Vector4
        96      {
 0      97        get {
 0      98          Point pos = Position.Absolute;
 0      99          Point size = Size.Absolute;
 0     100          return new Vector4 (pos.X, pos.Y, size.X, size.Y);
 0     101        }
       102      }
       103
       104      #endregion
       105
       106      #region Constructors
       107
89     108      public Bounds (ScreenPoint position, ScreenPoint size, ScreenPoint padding
89     109      {
89     110        _position = position;
89     111        _size = size;
89     112        _padding = padding;
89     113      }
       114
 8     115      public Bounds (ScreenPoint position, ScreenPoint size)
 8     116      {
 8     117        _position = position;
 8     118        _size = size;
 8     119        _padding = new ScreenPoint (position.Screen, Vector2.Zero);
 8     120      }
       121
 6     122      public Bounds (IGameScreen screen, float relX, float relY, float relWidth,
 6     123      {
 6     124        _position = new ScreenPoint (screen, relX, relY);
 6     125        _size = new ScreenPoint (screen, relWidth, relHeight);
 6     126        _padding = new ScreenPoint (screen, Vector2.Zero);
 6     127      }
       128
       129      #endregion
       130
       131      #region Methods and Operators
       132
       133      public bool Contains (Point point)
 1     134      {
 1     135        return Rectangle.Contains (point);
 1     136      }
       137
       138      public bool Contains (ScreenPoint point)
 1     139      {
 1     140        return Rectangle.Contains ((Point)point);
 1     141      }
       142
       143      public static Bounds Zero (IGameScreen screen)
```

```
 81   144      {
 81   145        return new Bounds (
      146                    position: ScreenPoint.Zero (screen),
      147                    size: ScreenPoint.Zero (screen),
      148                    padding: ScreenPoint.Zero (screen)
      149                );
 81   150      }
      151
      152      public Bounds FromLeft (Func<float> percent)
  3   153      {
  3   154        return new Bounds (
      155                    position: Position,
  4   156                    size: new ScreenPoint (Size.Screen, () => Size.Relative.X * p
      157                    padding: Padding
      158                );
  3   159      }
      160
      161      public Bounds FromRight (Func<float> percent)
  1   162      {
  1   163        return new Bounds (
  4   164                    position: Position + new ScreenPoint (Size.Screen, () => Size
  4   165                    size: new ScreenPoint (Size.Screen, () => Size.Relative.X * p
      166                    padding: Padding
      167                );
  1   168      }
      169
      170      public Bounds FromTop (Func<float> percent)
  3   171      {
  3   172        return new Bounds (
      173                    position: Position,
  4   174                    size: new ScreenPoint (Size.Screen, () => Size.Relative.X, ()
      175                    padding: Padding
      176                );
  3   177      }
      178
      179      public Bounds FromBottom (Func<float> percent)
  1   180      {
  1   181        return new Bounds (
  4   182                    position: Position + new ScreenPoint (Size.Screen, () => 0, (
  4   183                    size: new ScreenPoint (Size.Screen, () => Size.Relative.X, ()
      184                    padding: Padding
      185                );
  1   186      }
      187
      188      public Bounds FromLeft (float percent)
  3   189      {
  5   190        return FromLeft (() => percent);
  3   191      }
      192
      193      public Bounds FromRight (float percent)
  1   194      {
  5   195        return FromRight (() => percent);
  1   196      }
      197
      198      public Bounds FromTop (float percent)
  3   199      {
  5   200        return FromTop (() => percent);
  3   201      }
      202
      203      public Bounds FromBottom (float percent)
  1   204      {
```

```
5   205         return FromBottom (() => percent);
1   206       }
    207
    208       public Bounds In (Bounds container)
0   209       {
0   210         return new Bounds (Position + container.Position, Size, Padding);
0   211       }
    212
    213       public static implicit operator Rectangle (Bounds bounds)
0   214       {
0   215         return bounds.Rectangle;
0   216       }
    217
    218       [ExcludeFromCodeCoverageAttribute]
    219       public override string ToString ()
    220       {
    221         return "("
    222                + Position.Relative.X.ToString ()
    223                + "x"
    224                + Position.Relative.Y.ToString ()
    225                + ","
    226                + Size.Relative.X.ToString ()
    227                + "x"
    228                + Size.Relative.Y.ToString ()
    229                + ")";
    230       }
    231
    232       #endregion
    233     }
    234   }
```

Knot3.Widgets.ScreenPoint

Summary

Class:	Knot3.Widgets.ScreenPoint
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\ScreenPoint.cs
Coverage:	50.8%
Covered lines:	59
Uncovered lines:	57
Coverable lines:	116
Total lines:	267

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	100	100
.ctor(...)	1	100	100
.ctor(...)	2	100	66.67
Assign(...)	1	100	100
Zero(...)	1	100	100
TopLeft(...)	1	0	0
BottomRight(...)	1	0	0
Centered(...)	1	0	0
op_Implicit(...)	1	100	100
op_Implicit(...)	1	0	0
op_Implicit(...)	1	100	100
op_Implicit(...)	1	0	0
op_Multiply(...)	1	0	0
op_Multiply(...)	1	0	0
op_Division(...)	1	0	0
op_Addition(...)	1	100	100
op_Subtraction(...)	1	0	0
ScaleX(...)	1	0	0
ScaleY(...)	1	0	0
op_Equality(...)	4	66.67	57.14
op_Inequality(...)	1	100	100
Equals(...)	3	100	60
Equals(...)	5	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\ScreenPoint.cs

#	Line	Coverage	
	1		#region Copyright
	2		
	3		/*
	4		* Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
	5		* Gerd Augsburg, Christina Erler, Daniel Warzel
	6		*
	7		* Permission is hereby granted, free of charge, to any person obtaining a cop
	8		* of this software and associated documentation files (the "Software"), to de
	9		* in the Software without restriction, including without limitation the right
	10		* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
	11		* copies of the Software, and to permit persons to whom the Software is
	12		* furnished to do so, subject to the following conditions:
	13		*

```
14   * The above copyright notice and this permission notice shall be included in
15   * copies or substantial portions of the Software.
16   *
17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FRO
22   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
23   * SOFTWARE.
24   */
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.Diagnostics.CodeAnalysis;
34  using System.Linq;
35
36  using Microsoft.Xna.Framework;
37
38  using Knot3.Core;
39  using Knot3.Input;
40  using Knot3.Utilities;
41
42  #endregion
43
44  namespace Knot3.Widgets
45  {
46    public class ScreenPoint : IEquatable<ScreenPoint>
47    {
48      #region Properties
49
325   50      public IGameScreen Screen { get; private set; }
51
52      public Vector2 Relative
53      {
76    54        get {
76    55          return RelativeFunc ();
76    56        }
285   57        set {
345   58          RelativeFunc = () => value;
285   59        }
60      }
61
375   62      public Func<Vector2> RelativeFunc
63      {
64        set;
65        private get;
66      }
67
68      public Point Absolute
69      {
6     70        get {
6     71          return Relative.Scale (Screen.Viewport).ToPoint ();
6     72        }
73      }
74
```

```
        75      public ScreenPoint OnlyX
        76      {
   0    77        get {
   0    78          return new ScreenPoint (Screen, () => new Vector2 (RelativeFunc ().X,
   0    79        }
        80      }
        81
        82      public ScreenPoint OnlyY
        83      {
   0    84        get {
   0    85          return new ScreenPoint (Screen, () => new Vector2 (0, RelativeFunc ().
   0    86        }
        87      }
        88
        89      public ScreenPoint Const
        90      {
   0    91        get {
   0    92          return new ScreenPoint (Screen, Relative.X, Relative.Y);
   0    93        }
        94      }
        95
   0    96      public bool IsEmpty { get { return Relative.Length () == 0; } }
        97
        98      #endregion
        99
       100      #region Constructors
       101
   2   102      public ScreenPoint (IGameScreen screen, Func<Vector2> func)
   2   103      {
   2   104        Screen = screen;
   2   105        RelativeFunc = func;
   2   106      }
       107
 257   108      public ScreenPoint (IGameScreen screen, Vector2 vector)
 257   109      {
 257   110        Screen = screen;
 257   111        Relative = vector;
 257   112      }
       113
  28   114      public ScreenPoint (IGameScreen screen, float x, float y)
  28   115      {
  28   116        Screen = screen;
  28   117        Relative = new Vector2 (x, y);
  28   118      }
       119
  10   120      public ScreenPoint (IGameScreen screen, Func<float> x, Func<float> y)
  10   121      {
  10   122        Screen = screen;
  22   123        RelativeFunc = () => new Vector2 (x (), y ());
  10   124      }
       125
       126      #endregion
       127
       128      #region Methods and Operators
       129
       130      public void Assign (ScreenPoint other)
   1   131      {
   1   132        Screen = other.Screen;
   1   133        RelativeFunc = other.RelativeFunc;
   1   134      }
       135
```

```
      136       public static ScreenPoint Zero (IGameScreen screen)
243   137       {
243   138         return new ScreenPoint (screen, Vector2.Zero);
243   139       }
      140
      141       public static ScreenPoint TopLeft (IGameScreen screen)
  0   142       {
  0   143         return new ScreenPoint (screen, Vector2.Zero);
  0   144       }
      145
      146       public static ScreenPoint BottomRight (IGameScreen screen)
  0   147       {
  0   148         return new ScreenPoint (screen, Vector2.One);
  0   149       }
      150
      151       public static ScreenPoint Centered (IGameScreen screen, Bounds sizeOf)
  0   152       {
  0   153         return new ScreenPoint (screen, () => (ScreenPoint.BottomRight (screen)
  0   154       }
      155
      156       public static implicit operator Vector2 (ScreenPoint point)
  2   157       {
  2   158         return point.Relative;
  2   159       }
      160
      161       public static implicit operator Func<Vector2> (ScreenPoint point)
  0   162       {
  0   163         return point.RelativeFunc;
  0   164       }
      165
      166       public static implicit operator Point (ScreenPoint point)
  2   167       {
  2   168         return point.Absolute;
  2   169       }
      170
      171       public static implicit operator bool (ScreenPoint point)
  0   172       {
  0   173         return !point.IsEmpty;
  0   174       }
      175
      176       [ExcludeFromCodeCoverageAttribute]
      177       public override string ToString ()
      178       {
      179         return "(" + Relative.X + "x" + Relative.Y + ")";
      180       }
      181
      182       public static ScreenPoint operator * (ScreenPoint a, float b)
  0   183       {
  0   184         return new ScreenPoint (a.Screen, () => a.Relative * b);
  0   185       }
      186
      187       public static ScreenPoint operator * (ScreenPoint a, ScreenPoint b)
  0   188       {
  0   189         return new ScreenPoint (a.Screen, () => new Vector2 (a.Relative.X * b.Re
  0   190       }
      191
      192       public static ScreenPoint operator / (ScreenPoint a, float b)
  0   193       {
  0   194         return new ScreenPoint (a.Screen, () => a.Relative / b);
  0   195       }
      196
```

```
      197       public static ScreenPoint operator + (ScreenPoint a, ScreenPoint b)
    2 198       {
    6 199         return new ScreenPoint (a.Screen, () => a.Relative + b.Relative);
    2 200       }
      201
      202       public static ScreenPoint operator - (ScreenPoint a, ScreenPoint b)
    0 203       {
    0 204         return new ScreenPoint (a.Screen, () => a.Relative - b.Relative);
    0 205       }
      206
      207       public ScreenPoint ScaleX (float percent)
    0 208       {
    0 209         return new ScreenPoint (Screen, () => new Vector2 (Relative.X * percent,
    0 210       }
      211
      212       public ScreenPoint ScaleY (float percent)
    0 213       {
    0 214         return new ScreenPoint (Screen, () => new Vector2 (Relative.X, Relative.
    0 215       }
      216
      217       public static bool operator == (ScreenPoint a, ScreenPoint b)
   10 218       {
   10 219         if (System.Object.ReferenceEquals (a, b)) {
    0 220           return true;
      221         }
   20 222         if (((object)a == null) || ((object)b == null)) {
   10 223           return false;
      224         }
    0 225         return a.Equals (b);
   10 226       }
      227
      228       public static bool operator != (ScreenPoint d1, ScreenPoint d2)
   10 229       {
   10 230         return !(d1 == d2);
   10 231       }
      232
      233       public bool Equals (ScreenPoint other)
   10 234       {
   10 235         float epsilon = 0.000001f;
      236
   10 237         return other != null && Math.Abs (Relative.X - other.Relative.X) < epsil
   10 238       }
      239
      240       public override bool Equals (object other)
    0 241       {
    0 242         if (other == null) {
    0 243           return false;
      244         }
    0 245         else if (other is Vector2) {
    0 246           return Relative.Equals ((Vector2)other);
      247         }
    0 248         else if (other is Point) {
    0 249           return Absolute.Equals ((Point)other);
      250         }
    0 251         else if ((other = other as string) != null) {
    0 252           return ToString ().Equals (other);
      253         }
    0 254         else {
    0 255           return false;
      256         }
    0 257       }
```

```
258
259       [ExcludeFromCodeCoverageAttribute]
260       public override int GetHashCode ()
261       {
262         return Relative.GetHashCode ();
263       }
264
265       #endregion
266     }
267   }
```