Summary

Generated on:	17.02.2014 - 12:27:12
Parser:	OpenCoverParser
Assemblies:	1
Classes:	50
Files:	49
Coverage:	35%
Covered lines:	1143
Uncovered lines:	2119
Coverable lines:	3262
Total lines:	9759

Assemblies

Knot3	**35%**
Knot3.Audio.AudioManager	0%
Knot3.Audio.LoopPlaylist	0%
Knot3.Audio.OggVorbisFile	0%
Knot3.Audio.SoundEffectFile	0%
Knot3.Core.Angles3	89.8%
Knot3.Core.BooleanOptionInfo	0%
Knot3.Core.Camera	55.9%
Knot3.Core.ConfigFile	23.6%
Knot3.Core.DisplayLayer	31.1%
Knot3.Core.DistinctOptionInfo	0%
Knot3.Core.DrawableGameScreenComponent	76.9%
Knot3.Core.FloatOptionInfo	0%
Knot3.Core.GameScreenComponent	77.7%
Knot3.Core.KeyOptionInfo	0%
Knot3.Core.Localizer	0%
Knot3.Core.OptionInfo	0%
Knot3.Core.Options	100%
Knot3.Core.World	13.5%
Knot3.Data.Challenge	0%
Knot3.Data.ChallengeFileIO	0%
Knot3.Data.ChallengeMetaData	0%
Knot3.Data.CircleEntry‘1	92.1%
Knot3.Data.CircleExtensions	0%
Knot3.Data.Direction	64.1%
Knot3.Data.DirectionHelper	0%
Knot3.Data.Edge	66.2%
Knot3.Data.Knot	84.9%
Knot3.Data.KnotFileIO	20.5%
Knot3.Data.KnotMetaData	60.3%
Knot3.Data.KnotStringIO	40.1%
Knot3.Data.Node	76.1%
Knot3.Data.NodeMap	88.5%
Knot3.Data.PrinterIO	0%
Knot3.Data.RectangleMap	0%
Knot3.Data.ZipHelper	0%
Knot3.Platform.SystemInfo	50%
Knot3.Utilities.BoundingCylinder	0%
Knot3.Utilities.ColorHelper	0%
Knot3.Utilities.DictionaryHelper	100%
Knot3.Utilities.EnumHelper	0%
Knot3.Utilities.FileIndex	0%
Knot3.Utilities.FileUtility	13.8%
Knot3.Utilities.FrustumHelper	0%

Knot3.Utilities.IniFile	52.1%
Knot3.Utilities.InputHelper	0%
Knot3.Utilities.RayExtensions	0%
Knot3.Utilities.SavegameLoader‘2	0%
Knot3.Utilities.TextHelper	0%
Knot3.Utilities.TextureHelper	0%
Knot3.Utilities.VectorHelper	19.6%

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:	0%
Covered lines:	0
Uncovered lines:	108
Coverable lines:	108
Total lines:	245

Metrics

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

File(s)

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

```
#region Copyright

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

```
25
26  #endregion
27
28  #region Using
29
30  using System;
31  using System.Collections;
32  using System.Collections.Generic;
33  using System.IO;
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.RenderEffects;
52  using Knot3.Screens;
53  using Knot3.Utilities;
54  using Knot3.Widgets;
55
56  #endregion
57
58  namespace Knot3.Audio
59  {
60    public class AudioManager : DrawableGameScreenComponent
61    {
62      /// <summary>
63      /// Eine Zuordnung zwischen dem Typ der Audiodateien und den Ordnern unter
64      /// in denen sich die Audiodateien befinden.
65      /// </summary>
0   66      private static readonly Dictionary<Sound, string> AudioDirectories
67      = new Dictionary<Sound, string> {
68        { Sound.CreativeMusic,      "Music/Creative" },
69        { Sound.ChallengeMusic,      "Music/Challenge" },
70        { Sound.MenuMusic,        "Music/Menu" },
71        { Sound.PipeMoveSound,      "Sound/Pipe/Move" },
72        { Sound.PipeInvalidMoveSound,  "Sound/Pipe/Invalid-Move" },
73      };
74
75      // Enthlt alle gefunden Audiodateien, sortiert nach ihrem Zweck
0   76      private static Dictionary<Sound, HashSet<IAudioFile>> AudioFiles
77          = new Dictionary<Sound, HashSet<IAudioFile>> ();
78
79      /// <summary>
80      /// Die aktuell verwendete Hintergrundmusik.
81      /// </summary>
82      public Sound BackgroundMusic
83      {
0   84        get {
0   85          return _backgroundMusic;
```

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

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

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

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:	35
Coverable lines:	35
Total lines:	130

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	0	0
Shuffle()	1	0	0
Start()	2	0	0
Stop()	2	0	0
Update(...)	4	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.IO;
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.RenderEffects;
52  using Knot3.Screens;
53  using Knot3.Utilities;
54  using Knot3.Widgets;
55
56  #endregion
57
58  namespace Knot3.Audio
59  {
60    /// <summary>
61    /// Diese Klasse reprsentiert eine Playlist, deren Audiodateien der reihe n
62    /// Endlosschleife abgespielt werden.
63    /// </summary>
64    public class LoopPlaylist : IPlaylist
65    {
66      private List<IAudioFile> Sounds;
67      private int index;
68
0   69      public SoundState State { get; private set; }
70
71      /// <summary>
72      /// Erstellt eine neue Playlist.
73      /// </summary>
74      /// <param name='sounds'>
75      /// Die abzuspielenden Audiodateien.
76      /// </param>
0   77      public LoopPlaylist (IEnumerable<IAudioFile> sounds)
0   78      {
0   79        Sounds = sounds.ToList ();
0   80        index = 0;
0   81        State = SoundState.Stopped;
82
0   83        Log.Debug ("Created new playlist (", Sounds.Count, " songs)");
0   84        foreach (IAudioFile sound in Sounds) {
0   85          Log.Debug ("  - ", sound.Name);
0   86        }
0   87      }
88
89      public void Shuffle ()
0   90      {
0   91        Sounds = Sounds.Shuffle ().ToList ();
0   92      }
93
```

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

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:	0%
Covered lines:	0
Uncovered lines:	54
Coverable lines:	54
Total lines:	142

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	7	0	0
Play()	1	0	0
Stop()	1	0	0
Update(...)	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.IO;
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 OggSharp;
47
48  using Knot3.Core;
49  using Knot3.Data;
50  using Knot3.Development;
51  using Knot3.GameObjects;
52  using Knot3.Input;
53  using Knot3.Platform;
54  using Knot3.RenderEffects;
55  using Knot3.Screens;
56  using Knot3.Utilities;
57  using Knot3.Widgets;
58
59  #endregion
60
61  namespace Knot3.Audio
62  {
63    public class OggVorbisFile : IAudioFile
64    {
0   65      public string Name { get; private set; }
66
0   67      public SoundState State { get { return internalFile.State; } }
68
69      private SoundEffectFile internalFile;
70
0   71      public OggVorbisFile (string name, string filepath, Sound soundType)
0   72      {
0   73        Name = name;
0   74        string cachefile = SystemInfo.DecodedMusicCache
75                             + SystemInfo.PathSeparator.ToString ()
76                             + soundType.ToString ()
77                             + "_"
78                             + name.GetHashCode ().ToString ()
79                             + ".wav";
80
81        byte[] data;
0   82        try {
0   83          Log.Debug ("Read from cache: ", cachefile);
0   84          data = File.ReadAllBytes (cachefile);
0   85        }
0   86        catch (Exception) {
0   87          Log.Debug ("Decode: ", name);
0   88          OggDecoder decoder = new OggDecoder ();
0   89          decoder.Initialize (TitleContainer.OpenStream (filepath));
0   90          data = decoder.SelectMany (chunk => chunk.Bytes.Take (chunk.Length)).T
0   91          using (MemoryStream stream = new MemoryStream ())
0   92          using (BinaryWriter writer = new BinaryWriter (stream)) {
0   93            WriteWave (writer, decoder.Stereo ? 2 : 1, decoder.SampleRate, data)
```

```
0    94            stream.Position = 0;
0    95            data = stream.ToArray ();
0    96          }
0    97          File.WriteAllBytes (cachefile, data);
0    98        }
     99
0   100        using (MemoryStream stream = new MemoryStream (data)) {
0   101          stream.Position = 0;
0   102          SoundEffect soundEffect = SoundEffect.FromStream (stream);
0   103          internalFile = new SoundEffectFile (name, soundEffect, soundType);
0   104        }
0   105      }
    106
    107      public void Play ()
0   108      {
0   109        internalFile.Play ();
0   110      }
    111
    112      public void Stop ()
0   113      {
0   114        internalFile.Stop ();
0   115      }
    116
    117      public void Update (GameTime time)
0   118      {
0   119        internalFile.Update (time);
0   120      }
    121
    122      private static void WriteWave (BinaryWriter writer, int channels, int rate
0   123      {
0   124        writer.Write (new char[4] { 'R', 'I', 'F', 'F' });
0   125        writer.Write ((int)(36 + data.Length));
0   126        writer.Write (new char[4] { 'W', 'A', 'V', 'E' });
    127
0   128        writer.Write (new char[4] { 'f', 'm', 't', ' ' });
0   129        writer.Write ((int)16);
0   130        writer.Write ((short)1);
0   131        writer.Write ((short)channels);
0   132        writer.Write ((int)rate);
0   133        writer.Write ((int)(rate * ((16 * channels) / 8)));
0   134        writer.Write ((short)((16 * channels) / 8));
0   135        writer.Write ((short)16);
    136
0   137        writer.Write (new char[4] { 'd', 'a', 't', 'a' });
0   138        writer.Write ((int)data.Length);
0   139        writer.Write (data);
0   140      }
    141    }
    142  }
```

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:	24
Coverable lines:	24
Total lines:	113

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	0	0
Play()	1	0	0
Stop()	1	0	0
Update(...)	2	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.IO;
```

```
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.Data;
using Knot3.Development;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Utilities;
using Knot3.Widgets;

#endregion

namespace Knot3.Audio
{
  /// <summary>
  /// Ein Wrapper um die SoundEffect-Klasse des XNA-Frameworks.
  /// </summary>
  public class SoundEffectFile : IAudioFile
  {
    /// <summary>
    /// Der Anzeigename des SoundEffects.
    /// </summary>
0   public string Name { get; private set; }

    /// <summary>
    /// Gibt an, ob die Wiedergabe luft oder gestoppt bzw. pausiert ist.
    /// </summary>
0   public SoundState State { get { return Instance.State; } }

0   public SoundEffect SoundEffect { get; private set; }

    private SoundEffectInstance Instance;

    private Sound SoundType;
    private float volume;

    /// <summary>
    /// Erstellt eine neue SoundEffect-Datei mit dem angegebenen Anzeigenamen
    /// </summary>
0   public SoundEffectFile (string name, SoundEffect soundEffect, Sound soundT
0   {
0     Name = name;
0     SoundEffect = soundEffect;
0     Instance = soundEffect.CreateInstance ();
0     SoundType = soundType;
0   }

    public void Play ()
0   {
```

```
0    95        Log.Debug ("Play: ", Name);
0    96        Instance.Volume = volume = AudioManager.Volume (SoundType);
0    97        Instance.Play ();
0    98      }
     99
    100      public void Stop ()
0   101      {
0   102        Log.Debug ("Stop: ", Name);
0   103        Instance.Stop ();
0   104      }
    105
    106      public void Update (GameTime time)
0   107      {
0   108        if (volume != AudioManager.Volume (SoundType)) {
0   109          Instance.Volume = volume = AudioManager.Volume (SoundType);
0   110        }
0   111      }
    112    }
    113  }
```

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:	89.8%
Covered lines:	62
Uncovered lines:	7
Coverable lines:	69
Total lines:	223

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
GetHashCode()	1	0	0
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
ToString()	1	0	0

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.Linq;
34
35 using Microsoft.Xna.Framework;
36 using Microsoft.Xna.Framework.Audio;
37 using Microsoft.Xna.Framework.Content;
38 using Microsoft.Xna.Framework.GamerServices;
39 using Microsoft.Xna.Framework.Graphics;
40 using Microsoft.Xna.Framework.Input;
41 using Microsoft.Xna.Framework.Media;
42 using Microsoft.Xna.Framework.Net;
43 using Microsoft.Xna.Framework.Storage;
44
45 using Knot3.Data;
46 using Knot3.GameObjects;
47 using Knot3.RenderEffects;
48 using Knot3.Screens;
49 using Knot3.Widgets;
50
51 #endregion
52
53 namespace Knot3.Core
54 {
55   /// <summary>
56   /// Diese Klasse reprsentiert die Rollwinkel der drei Achsen X, Y und Z.
57   /// Sie bietet Mglichkeit vordefinierte Winkelwerte zu verwenden, z.B. stel
58   /// Die Umwandlung zwischen verschiedenen Winkelmaen wie Grad- und Bogenma
59   /// </summary>
60   public sealed class Angles3 : IEquatable<Angles3>
61   {
62     #region Properties
63
64     /// <summary>
65     /// Der Winkel im Bogenma fr das Rollen um die X-Achse. Siehe statische
66     /// </summary>
150 67     public float X { get; set; }
68
69     /// <summary>
70     /// Der Winkel im Bogenma fr das Rollen um die Y-Achse. Siehe statische
71     /// </summary>
146 72     public float Y { get; set; }
73
74     /// <summary>
75     /// Der Winkel im Bogenma fr das Rollen um die Z-Achse. Siehe statische
76     /// </summary>
146 77     public float Z { get; set; }
78
79     /// <summary>
80     /// Eine statische Eigenschaft mit dem Wert X = 0, Y = 0, Z = 0.
```

```
      81      /// </summary>
      82      public static Angles3 Zero
      83      {
54    84        get { return new Angles3 (0f, 0f, 0f); }
      85      }
      86
      87      #endregion
      88
      89      #region Constructors
      90
      91      /// <summary>
      92      /// Konstruiert ein neues Angles3-Objekt mit drei gegebenen Winkeln im Bog
      93      /// </summary>
89    94      public Angles3 (float x, float y, float z)
89    95      {
89    96        X = x;
89    97        Y = y;
89    98        Z = z;
89    99      }
     100
 4   101      public Angles3 (Vector3 v)
 4   102      {
 4   103        X = v.X;
 4   104        Y = v.Y;
 4   105        Z = v.Z;
 4   106      }
     107
     108      #endregion
     109
     110      #region Methods
     111
     112      /// <summary>
     113      /// Eine statische Methode, die Grad in Bogenma konvertiert.
     114      /// </summary>
     115      public static Angles3 FromDegrees (float x, float y, float z)
46   116      {
46   117        return new Angles3 (
     118                    MathHelper.ToRadians (x),
     119                    MathHelper.ToRadians (y),
     120                    MathHelper.ToRadians (z)
     121                );
46   122      }
     123
     124      /// <summary>
     125      /// Konvertiert Bogenma in Grad.
     126      /// </summary>
     127      public void ToDegrees (out float x, out float y, out float z)
 1   128      {
 1   129        x = (int)MathHelper.ToDegrees (X) % 360;
 1   130        y = (int)MathHelper.ToDegrees (Y) % 360;
 1   131        z = (int)MathHelper.ToDegrees (Z) % 360;
 1   132      }
     133
     134      public override bool Equals (object obj)
 1   135      {
 1   136        return (obj is Angles3) ? this == (Angles3)obj : false;
 1   137      }
     138
     139      public bool Equals (Angles3 other)
12   140      {
12   141        return this == other;
```

```
12  142     }
    143
    144     public override int GetHashCode ()
 0  145     {
 0  146       return (int)(this.X + this.Y + this.Z);
 0  147     }
    148
    149     #endregion
    150
    151     #region Operators
    152
    153     public static bool operator == (Angles3 value1, Angles3 value2)
14  154     {
14  155       return value1.X == value2.X
    156              && value1.Y == value2.Y
    157              && value1.Z == value2.Z;
14  158     }
    159
    160     public static bool operator != (Angles3 value1, Angles3 value2)
 1  161     {
 1  162       return !(value1 == value2);
 1  163     }
    164
    165     public static Angles3 operator + (Angles3 value1, Angles3 value2)
 5  166     {
 5  167       return new Angles3 (value1.X + value2.X, value1.Y + value2.Y, value1.Z +
 5  168     }
    169
    170     public static Angles3 operator - (Angles3 value)
 1  171     {
 1  172       value = new Angles3 (-value.X, -value.Y, -value.Z);
 1  173       return value;
 1  174     }
    175
    176     public static Angles3 operator - (Angles3 value1, Angles3 value2)
 1  177     {
 1  178       return new Angles3 (value1.X - value2.X, value1.Y - value2.Y, value1.Z -
 1  179     }
    180
    181     public static Angles3 operator * (Angles3 value1, Angles3 value2)
 1  182     {
 1  183       return new Angles3 (value1.X * value2.X, value1.Y * value2.Y, value1.Z *
 1  184     }
    185
    186     public static Angles3 operator * (Angles3 value, float scaleFactor)
 1  187     {
 1  188       return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
 1  189     }
    190
    191     public static Angles3 operator * (float scaleFactor, Angles3 value)
 1  192     {
 1  193       return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
 1  194     }
    195
    196     public static Angles3 operator / (Angles3 value1, Angles3 value2)
 1  197     {
 1  198       return new Angles3 (value1.X / value2.X, value1.Y / value2.Y, value1.Z /
 1  199     }
    200
    201     public static Angles3 operator / (Angles3 value, float divider)
 1  202     {
```

```
1   203         float scaleFactor = 1 / divider;
1   204         return new Angles3 (value.X * scaleFactor, value.Y * scaleFactor, value.
1   205       }
    206
    207       public override string ToString ()
0   208       {
    209         float x, y, z;
0   210         ToDegrees (out x, out y, out z);
    211
0   212         return   "Angles3 ("
    213                  + x.ToString ()
    214                  + ","
    215                  + y.ToString ()
    216                  + ","
    217                  + z.ToString ()
    218                  + ")";
0   219       }
    220
    221       #endregion
    222     }
    223   }
```

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:	96

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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`

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

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:	55.9%
Covered lines:	75
Uncovered lines:	59
Coverable lines:	134
Total lines:	379

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	100	100
Update(...)	1	0	0
UpdateMatrices(...)	1	100	100
GetMouseRay(...)	1	0	0
ResetCamera()	1	100	100
StartSmoothMove(...)	2	0	0
UpdateSmoothMove(...	2	0	0
To3D(...)	2	43.75	66.67
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.Linq;
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Data;
46  using Knot3.GameObjects;
47  using Knot3.RenderEffects;
48  using Knot3.Screens;
49  using Knot3.Utilities;
50  using Knot3.Widgets;
51
52  #endregion
53
54  namespace Knot3.Core
55  {
56    /// <summary>
57    /// Jede Instanz der World-Klasse hlt eine fr diese Spielwelt verwendete K
58    /// Die Hauptfunktion der Kamera-Klasse ist das Berechnen der drei Matrizen,
59    /// und Skalierung von 3D-Objekten in einer bestimmten Spielwelt bentigt we
60    /// Um diese Matrizen zu berechnen, bentigt die Kamera unter Anderem Inform
61    /// das aktuelle Kamera-Ziel und das Field of View.
62    /// </summary>
63    public sealed class Camera : GameScreenComponent
64    {
65      #region Properties
66
67      private Vector3 _position;
68
69      /// <summary>
70      /// Die Position der Kamera.
71      /// </summary>
72      public Vector3 Position
73      {
74        get { return _position; }
75        set {
76          OnViewChanged ();
77          if ((value.X.Abs () <= MaxPositionDistance && value.Y.Abs () <= MaxPos
78                  && value.Z.Abs () <= MaxPositionDistance) || MaxPositionDistan
79            _position = value;
80          }
81        }
82      }
83
84      private Vector3 _target;
85
86      /// <summary>
87      /// Das Ziel der Kamera.
88      /// </summary>
89      public Vector3 Target
```

75	74
16	75
16	76
16	77
16	78
16	79
16	80
16	81

```
      90      {
78    91        get { return _target; }
12    92        set {
12    93          OnViewChanged ();
12    94          _target = value;
12    95        }
      96      }
      97
      98      private float _foV;
      99
     100      /// <summary>
     101      /// Das Sichtfeld.
     102      /// </summary>
     103      public float FoV
     104      {
36   105        get { return _foV; }
11   106        set {
11   107          _foV = MathHelper.Clamp (value, 10, 70);
11   108          OnViewChanged ();
11   109        }
     110      }
     111
     112      /// <summary>
     113      /// Die View-Matrix wird ber die statische Methode CreateLookAt der Klass
     114      /// mit Matrix.CreateLookAt (Position, Target, Vector3.Up) berechnet.
     115      /// </summary>
20   116      public Matrix ViewMatrix { get; private set; }
     117
     118      /// <summary>
     119      /// Die World-Matrix wird mit Matrix.CreateFromYawPitchRoll und den drei R
     120      /// </summary>
11   121      public Matrix WorldMatrix { get; private set; }
     122
     123      /// <summary>
     124      /// Die Projektionsmatrix wird ber die statische XNA-Methode Matrix.Creat
     125      /// </summary>
20   126      public Matrix ProjectionMatrix { get; private set; }
     127
     128      /// <summary>
     129      /// Berechnet ein Bounding-Frustum, das bentigt wird, um festzustellen, o
     130      /// </summary>
 9   131      public BoundingFrustum ViewFrustum { get; private set; }
     132
     133      /// <summary>
     134      /// Eine Referenz auf die Spielwelt, fr welche die Kamera zustndig ist.
     135      /// </summary>
19   136      private World World { get; set; }
     137
     138      /// <summary>
     139      /// Die Rotationswinkel.
     140      /// </summary>
34   141      public Angles3 Rotation { get; set; }
     142
16   143      public Vector3 UpVector { get; private set; }
     144
56   145      public float MaxPositionDistance { get; set; }
     146
47   147      public Action OnViewChanged = () => {};
     148      private float aspectRatio;
     149      private float nearPlane;
     150      private float farPlane;
```

```
8   151     private Vector3 defaultPosition = new Vector3 (400, 400, 700);
    152
    153     #endregion
    154
    155     #region Constructors
    156
    157     /// <summary>
    158     /// Erstellt eine neue Kamera in einem bestimmten IGameScreen fr eine bes
    159     /// </summary>
8   160     public Camera (IGameScreen screen, World world)
    161     : base (screen, DisplayLayer.None)
8   162     {
8   163       World = world;
8   164       Position = defaultPosition;
8   165       Target = Vector3.Zero;
8   166       UpVector = Vector3.Up;
8   167       Rotation = Angles3.Zero;
8   168       MaxPositionDistance = 5000;
    169
8   170       FoV = 60;
8   171       nearPlane = 0.5f;
8   172       farPlane = 15000.0f;
    173
8   174       UpdateMatrices (null);
8   175     }
    176
    177     #endregion
    178
    179     #region Methods
    180
    181     /// <summary>
    182     /// Die Blickrichtung.
    183     /// </summary>
    184     public Vector3 PositionToTargetDirection
    185     {
0   186       get {
0   187         return Vector3.Normalize (Target - Position);
0   188       }
    189     }
    190
    191     public Vector3 PositionToArcballTargetDirection
    192     {
0   193       get {
0   194         return Vector3.Normalize (ArcballTarget - Position);
0   195       }
    196     }
    197
    198     /// <summary>
    199     /// Der Abstand zwischen der Kamera und dem Kamera-Ziel.
    200     /// </summary>
    201     public float PositionToTargetDistance
    202     {
4   203       get {
4   204         return Position.DistanceTo (Target);
4   205       }
4   206       set {
4   207         Position = Position.SetDistanceTo (Target, value);
4   208       }
    209     }
    210
    211     public float PositionToArcballTargetDistance
```

```
      212      {
0     213        get {
0     214          return Position.DistanceTo (ArcballTarget);
0     215        }
0     216        set {
0     217          Position = Position.SetDistanceTo (ArcballTarget, value);
0     218        }
      219      }
      220
      221      /// <summary>
      222      /// Wird fr jeden Frame aufgerufen.
      223      /// </summary>
      224      public override void Update (GameTime time)
0     225      {
      226        // Setze den Viewport auf den der aktuellen Spielwelt
0     227        Viewport original = Screen.Viewport;
0     228        Screen.Viewport = World.Viewport;
      229
0     230        UpdateMatrices (time);
0     231        UpdateSmoothMove (time);
      232
      233        // Setze den Viewport wieder auf den ganzen Screen
0     234        Screen.Viewport = original;
0     235      }
      236
      237      private void UpdateMatrices (GameTime time)
8     238      {
8     239        aspectRatio = Screen.Viewport.AspectRatio;
8     240        farPlane = MaxPositionDistance * 4;
8     241        ViewMatrix = Matrix.CreateLookAt (Position, Target, UpVector);
8     242        WorldMatrix = Matrix.CreateFromYawPitchRoll (Rotation.Y, Rotation.X, Rot
8     243        ProjectionMatrix = Matrix.CreatePerspectiveFieldOfView (MathHelper.ToRad
8     244        ViewFrustum = new BoundingFrustum (ViewMatrix * ProjectionMatrix);
8     245      }
      246
      247      /// <summary>
      248      /// Berechnet einen Strahl fr die angegebenene 2D-Mausposition.
      249      /// </summary>
      250      public Ray GetMouseRay (Vector2 mousePosition)
0     251      {
0     252        Viewport viewport = World.Viewport;
      253
0     254        Vector3 nearPoint = new Vector3 (mousePosition, 0);
0     255        Vector3 farPoint = new Vector3 (mousePosition, 1);
      256
0     257        nearPoint = viewport.Unproject (nearPoint, ProjectionMatrix, ViewMatrix,
0     258        farPoint = viewport.Unproject (farPoint, ProjectionMatrix, ViewMatrix, M
      259
0     260        Vector3 direction = farPoint - nearPoint;
0     261        direction.Normalize ();
      262
0     263        return new Ray (nearPoint, direction);
0     264      }
      265
      266      /// <summary>
      267      /// Eine Position, um die rotiert werden soll, wenn der User die rechte Ma
      268      /// </summary>
      269      public Vector3 ArcballTarget
      270      {
0     271        get {
0     272          if (World.SelectedObject != null) {
```

```
0   273            return World.SelectedObject.Center ();
    274          }
0   275          else {
0   276            return Vector3.Zero;
    277          }
0   278        }
    279      }
    280
    281      public void ResetCamera ()
2   282      {
2   283        Position = defaultPosition;
2   284        Target = new Vector3 (0, 0, 0);
2   285        Rotation = Angles3.Zero;
2   286        FoV = 45;
2   287      }
    288
8   289      private Vector3? smoothTarget = null;
8   290      private float smoothDistance = 0f;
8   291      private float smoothProgress = 0f;
    292
    293      public void StartSmoothMove (Vector3 target, GameTime time)
0   294      {
0   295        if (!InSmoothMove) {
0   296          smoothTarget = target;
0   297          smoothDistance = Math.Abs (Target.DistanceTo (target));
0   298          smoothProgress = 0f;
0   299        }
0   300      }
    301
0   302      public bool InSmoothMove { get { return smoothTarget.HasValue && smoothPro
    303
    304      private void UpdateSmoothMove (GameTime time)
0   305      {
0   306        if (InSmoothMove) {
0   307          float distance = MathHelper.SmoothStep (0, smoothDistance, smoothProgr
    308
0   309          smoothProgress += 0.05f;
    310
    311          //Log.Debug ("distance = ", distance);
0   312          Target = Target.SetDistanceTo (
    313                       target: smoothTarget.Value,
    314                       distance: Math.Max (0, smoothDistance - distance)
    315                   );
0   316          World.Redraw = true;
0   317        }
0   318      }
    319
    320      /// <summary>
    321      /// Berechne aus einer 2D-Positon (z.b. Mausposition) die entsprechende Po
    322      /// Fr die fehlende dritte Koordinate wird eine Angabe einer weiteren 3D-
    323      /// mit der die 3D-(Maus-)Position auf der selben Ebene liegen soll.
    324      /// </summary>
    325      public Vector3 To3D (Vector2 position, Vector3 nearTo)
1   326      {
1   327        if (Options.Default ["debug", "unproject", "SelectedObject"] == "NearFar
0   328          Vector3 nearScreenPoint = new Vector3 (position.X, position.Y, 0);
0   329          Vector3 farScreenPoint = new Vector3 (position.X, position.Y, 1);
0   330          Vector3 nearWorldPoint = World.Viewport.Unproject (
    331                                       source: nearScreenPoint,
    332                                       projection: World.Camera.ProjectionMatrix
    333                                       view: World.Camera.ViewMatrix,
```

```
      334                                     world: Matrix.Identity
      335                                 );
0     336          Vector3 farWorldPoint = World.Viewport.Unproject (
      337                                     source: farScreenPoint,
      338                                     projection: World.Camera.ProjectionMatrix,
      339                                     view: World.Camera.ViewMatrix,
      340                                     world: Matrix.Identity
      341                                 );
      342
0     343          Vector3 direction = farWorldPoint - nearWorldPoint;
      344
0     345          float zFactor = -nearWorldPoint.Y / direction.Y;
0     346          Vector3 zeroWorldPoint = nearWorldPoint + direction * zFactor;
0     347          return zeroWorldPoint;
      348        }
1     349        else {
1     350          Vector3 screenLocation = World.Viewport.Project (
      351                                     source: nearTo,
      352                                     projection: World.Camera.ProjectionMatrix
      353                                     view: World.Camera.ViewMatrix,
      354                                     world: World.Camera.WorldMatrix
      355                                 );
1     356          Vector3 currentMousePosition = World.Viewport.Unproject (
      357                                         source: new Vector3 (position, scre
      358                                         projection: World.Camera.Projection
      359                                         view: World.Camera.ViewMatrix,
      360                                         world: Matrix.Identity
      361                                     );
1     362          return currentMousePosition;
      363        }
1     364      }
      365
      366      public Vector2 To2D (Vector3 position)
1     367      {
1     368        Vector3 screenLocation = World.Viewport.Project (
      369                                     source: position,
      370                                     projection: World.Camera.ProjectionMatrix,
      371                                     view: World.Camera.ViewMatrix,
      372                                     world: World.Camera.WorldMatrix
      373                                 );
1     374        return new Vector2 (screenLocation.X, screenLocation.Y);
1     375      }
      376
      377      #endregion
      378    }
      379  }
```

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:	23.6%
Covered lines:	13
Uncovered lines:	42
Coverable lines:	55
Total lines:	187

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	2	100	66.67
SetOption(...)	1	0	0
GetOption(...)	1	100	100
SetOption(...)	2	0	0
GetOption(...)	4	0	0
SetOption(...)	1	0	0
GetOption(...)	1	0	0
floatToString(...)	1	0	0
stringToFloat(...)	2	0	0

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.IO;
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 Einstellungsdatei.
59    /// </summary>
60    public sealed class ConfigFile
61    {
62      #region Properties
63
64      /// <summary>
65      /// Die Reprsentation des Wahrheitswerts "wahr" als String in einer Einst
66      /// </summary>
0   67      public static string True { get { return "true"; } }
68
69      /// <summary>
70      /// Die Reprsentation des Wahrheitswerts "falsch" als String in einer Ein
71      /// </summary>
0   72      public static string False { get { return "false"; } }
73
74      private string Filename;
75      private IniFile ini;
76
77      #endregion
78
79      #region Constructors
80
2   81      public ConfigFile (string filename)
2   82      {
83        // load ini file
2   84        Filename = filename;
85
86        // create a new ini parser
4   87        using (StreamWriter w = File.AppendText (Filename)) {
2   88        }
2   89        ini = new IniFile (Filename);
```

```
2    90      }
     91
     92      #endregion
     93
     94      #region Methods
     95
     96      /// <summary>
     97      /// Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen
     98      /// </summary>
     99      public void SetOption (string section, string option, string _value)
0   100      {
0   101        ini [section, option] = _value;
0   102      }
    103
    104      /// <summary>
    105      /// Gibt den aktuell in der Datei vorhandenen Wert fr die angegebene Opti
    106      /// </summary>
    107      public string GetOption (string section, string option, string defaultValu
5   108      {
5   109        return ini [section, option, defaultValue];
5   110      }
    111
    112      /// <summary>
    113      /// Setzt den Wert der Option mit dem angegebenen Namen in den angegebenen
    114      /// </summary>
    115      public void SetOption (string section, string option, bool _value)
0   116      {
0   117        SetOption (section, option, _value ? True : False);
0   118      }
    119
    120      /// <summary>
    121      /// Gibt den aktuell in der Datei vorhandenen Wert fr die angegebene Opti
    122      /// </summary>
    123      public bool GetOption (string section, string option, bool defaultValue)
0   124      {
0   125        return GetOption (section, option, defaultValue ? True : False) == True
0   126      }
    127
    128      public void SetOption (string section, string option, float _value)
0   129      {
0   130        SetOption (section, option, floatToString (_value));
0   131      }
    132
    133      public float GetOption (string section, string option, float defaultValue)
0   134      {
0   135        return stringToFloat (GetOption (section, option, floatToString (default
0   136      }
    137
    138      private string floatToString (float f)
0   139      {
0   140        return String.Empty + ((int) (f * 1000)).ToString ();
0   141      }
    142
    143      private float stringToFloat (string s)
0   144      {
    145        int i;
0   146        bool result = Int32.TryParse (s, out i);
0   147        if (true == result) {
0   148          return ((float)i) / 1000f;
    149        }
0   150        else {
```

```
0   151          return 0;
    152        }
0   153      }
    154
    155      public bool this [string section, string option, bool defaultValue = false
    156      {
0   157        get {
0   158          return GetOption (section, option, defaultValue);
0   159        }
0   160        set {
0   161          SetOption (section, option, value);
0   162        }
    163      }
    164
    165      public float this [string section, string option, float defaultValue = 0f]
    166      {
0   167        get {
0   168          return GetOption (section, option, defaultValue);
0   169        }
0   170        set {
0   171          SetOption (section, option, value);
0   172        }
    173      }
    174
    175      public string this [string section, string option, string defaultValue = n
    176      {
5   177        get {
5   178          return GetOption (section, option, defaultValue);
5   179        }
0   180        set {
0   181          SetOption (section, option, value);
0   182        }
    183      }
    184
    185      #endregion
    186    }
    187  }
```

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:	31.1%
Covered lines:	19
Uncovered lines:	42
Coverable lines:	61
Total lines:	204

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
.ctor(...)	1	0	0
ToString()	1	0	0
op_Addition(...)	1	0	0
op_Addition(...)	1	0	0
op_Addition(...)	1	0	0
op_Multiply(...)	1	0	0
op_Equality(...)	4	0	0
op_Inequality(...)	1	0	0
Equals(...)	2	0	0
Equals(...)	2	0	0
op_Implicit(...)	1	0	0
op_Implicit(...)	1	100	100
GetHashCode()	1	0	0
.cctor()	1	100	100

File(s)

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

#	Line	Coverage

```
#region Copyright

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

```
 * SOFTWARE.
 */

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Data;
using Knot3.GameObjects;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Core
{
  /// <summary>
  /// Die Zeichenreihenfolge der Elemente der grafischen Benutzeroberflche.
  /// </summary>
  public class DisplayLayer : IEquatable<DisplayLayer>
  {
    #region Enumeration Values

    /// <summary>
    /// Steht fr die hinterste Ebene bei der Zeichenreihenfolge.
    /// </summary>
1   public static readonly DisplayLayer None = new DisplayLayer (0, "None");
    /// <summary>
    /// Steht fr eine Ebene hinter der Spielwelt, z.B. um
    /// Hintergrundbilder darzustellen.
    /// </summary>
1   public static readonly DisplayLayer Background = new DisplayLayer (10, "Ba
    /// <summary>
    /// Steht fr die Ebene in der die Spielwelt dargestellt wird.
    /// </summary>
1   public static readonly DisplayLayer GameWorld = new DisplayLayer (20, "Gam
1   public static readonly DisplayLayer ScreenUI = new DisplayLayer (30, "Scre
    /// <summary>
    /// Steht fr die Ebene in der die Dialoge dargestellt werden.
    /// Dialoge werden vor der Spielwelt gezeichnet, damit der Spieler damit i
    /// </summary>
1   public static readonly DisplayLayer Dialog = new DisplayLayer (50, "Dialog
    /// <summary>
    /// Steht fr die Ebene in der Mens gezeichnet werden. Mens werden inner
    /// </summary>
```

```
1   84      public static readonly DisplayLayer Menu = new DisplayLayer (10, "Menu");
        85      /// <summary>
        86      /// Steht fr die Ebene in der Meneintrge gezeichnet werden. Meneintrg
        87      /// </summary>
1   88      public static readonly DisplayLayer MenuItem = new DisplayLayer (20, "Menu
        89      /// <summary>
        90      /// Zum Anzeigen zustzlicher Informationen bei der (Weiter-)Entwicklung o
        91      /// </summary>
1   92      public static readonly DisplayLayer Overlay = new DisplayLayer (300, "Over
        93      /// <summary>
        94      /// Die Maus ist das Hauptinteraktionswerkzeug, welches der Spieler
        95      /// stndig verwendet. Daher muss die Maus bei der Interaktion immer
        96      /// im Vordergrund sein. Cursor steht fr die vorderste Ebene.
        97      /// </summary>
1   98      public static readonly DisplayLayer Cursor = new DisplayLayer (500, "Curso
        99
       100      public static readonly DisplayLayer[] Values = {
       101        None, Background, GameWorld, ScreenUI, Dialog, Menu, MenuItem, Overlay,
       102      };
       103
       104      #endregion
       105
       106      #region Static Attributes
       107
       108      #endregion
       109
       110      #region Properties
       111
13     112      public int Index { get; private set; }
       113
9      114      public string Description { get; private set; }
       115
9      116      #endregion
9      117
9      118      #region Constructors
9      119
9      120      private DisplayLayer (int index, string desciption)
       121      {
0      122        Index = index;
0      123        Description = desciption;
0      124      }
0      125
0      126      private DisplayLayer (DisplayLayer layer1, DisplayLayer layer2)
       127      {
       128        Index = layer1.Index + layer2.Index;
       129        Description = layer1.Description + "+" + layer2.Description;
       130      }
       131
       132      #endregion
0      133
0      134      #region Methods and Operators
0      135
       136      public override string ToString ()
       137      {
0      138        return Description;
0      139      }
0      140
       141      public static DisplayLayer operator + (DisplayLayer layer1, DisplayLayer l
       142      {
0      143        return new DisplayLayer (layer1, layer2);
0      144      }
```

```
0   145
    146     public static DisplayLayer operator + (DisplayLayer layer, Widget widget)
    147     {
0   148       return new DisplayLayer (widget.Index, layer);
0   149     }
0   150
    151     public static DisplayLayer operator * (DisplayLayer layer, int i)
    152     {
0   153       return new DisplayLayer (layer.Index * i, "(" + layer + "*" + i + ")");
0   154     }
0   155
    156     public static bool operator == (DisplayLayer a, DisplayLayer b)
    157     {
0   158       // If both are null, or both are same instance, return true.
    159       if (System.Object.ReferenceEquals (a, b)) {
0   160         return true;
0   161       }
    162
    163       // If one is null, but not both, return false.
    164       if (((object)a == null) || ((object)b == null)) {
0   165         return false;
0   166       }
    167
    168       // Return true if the fields match:
    169       return a.Index == b.Index;
0   170     }
0   171
    172     public static bool operator != (DisplayLayer d1, DisplayLayer d2)
    173     {
0   174       return !(d1 == d2);
0   175     }
0   176
    177     public bool Equals (DisplayLayer other)
    178     {
0   179       return other != null && Index == other.Index;
0   180     }
0   181
    182     public override bool Equals (object other)
    183     {
0   184       return other != null && Equals (other as DisplayLayer);
0   185     }
0   186
    187     public static implicit operator string (DisplayLayer layer)
    188     {
0   189       return layer.Description;
0   190     }
0   191
    192     public static implicit operator int (DisplayLayer layer)
    193     {
4   194       return layer.Index;
4   195     }
4   196
    197     public override int GetHashCode ()
    198     {
0   199       return Description.GetHashCode ();
0   200     }
0   201
    202     #endregion
    203   }
    204 }
```

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:	111

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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`
	36	`using Microsoft.Xna.Framework.Audio;`
	37	`using Microsoft.Xna.Framework.Content;`

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

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

Knot3.Core.DrawableGameScreenComponent

Summary

Class:	Knot3.Core.DrawableGameScreenComponent
Assembly:	Knot3
File(s):	scal\Documents\GitHub\knot3-code\src\Knot3\Core\DrawableGameScreenComponent.cs
Coverage:	76.9%
Covered lines:	10
Uncovered lines:	3
Coverable lines:	13
Total lines:	111

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
MoveNext()	2	0	0

File(s)

scal\Documents\GitHub\knot3-code\src\Knot3\Core\DrawableGameScreenComponent.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.Linq;
34
35 using Microsoft.Xna.Framework;
```

```
      36  using Microsoft.Xna.Framework.Audio;
      37  using Microsoft.Xna.Framework.Content;
      38  using Microsoft.Xna.Framework.GamerServices;
      39  using Microsoft.Xna.Framework.Graphics;
      40  using Microsoft.Xna.Framework.Input;
      41  using Microsoft.Xna.Framework.Media;
      42  using Microsoft.Xna.Framework.Net;
      43  using Microsoft.Xna.Framework.Storage;
      44
      45  using Knot3.Data;
      46  using Knot3.GameObjects;
      47  using Knot3.RenderEffects;
      48  using Knot3.Screens;
      49  using Knot3.Widgets;
      50
      51  #endregion
      52
      53  namespace Knot3.Core
      54  {
      55    /// <summary>
      56    /// Eine zeichenbare Spielkomponente, die in einem angegebenen Spielzustand
      57    /// </summary>
      58    public abstract class DrawableGameScreenComponent : DrawableGameComponent, I
      59    {
      60      #region Properties
      61
      62      /// <summary>
      63      /// Der zugewiesene Spielzustand.
      64      /// </summary>
 14   65      public IGameScreen Screen { get; set; }
      66
      67      private DisplayLayer _index;
      68
      69      /// <summary>
      70      /// Die Zeichen- und Eingabeprioritt.
      71      /// </summary>
      72      public DisplayLayer Index
      73      {
  0   74        get { return _index; }
  4   75        set {
  4   76          _index = value;
  4   77          DrawOrder = (int)value;
  4   78        }
      79      }
      80
      81      #endregion
      82
      83      #region Constructors
      84
      85      /// <summary>
      86      /// Erzeugt eine neue Instanz eines DrawableGameScreenComponent-Objekts un
      87      /// index bezeichnet die Zeichenebene, auf welche die Komponente zu zeichn
      88      /// </summary>
  4   89      public DrawableGameScreenComponent (IGameScreen screen, DisplayLayer index
      90      : base (screen.Game)
  4   91      {
  4   92        this.Screen = screen;
  4   93        this.Index = index;
  4   94      }
      95
      96      #endregion
```

```
      97
      98      #region Methods
      99
     100      /// <summary>
     101      /// Gibt Spielkomponenten zurck, die in dieser Spielkomponente enthalten
     102      /// [returntype=IEnumerable<IGameScreenComponent>]
     103      /// </summary>
     104      public virtual IEnumerable<IGameScreenComponent> SubComponents (GameTime G
0    105      {
0    106        yield break;
     107      }
     108
     109      #endregion
     110    }
     111  }
```

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:	120

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

```
#region Copyright

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

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

```

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

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

Knot3.Core.GameScreenComponent

Summary

Class:	Knot3.Core.GameScreenComponent
Assembly:	Knot3
File(s):	Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\GameScreenComponent.cs
Coverage:	77.7%
Covered lines:	7
Uncovered lines:	2
Coverable lines:	9
Total lines:	101

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
MoveNext()	2	0	0

File(s)

Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Core\GameScreenComponent.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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`

```
      36  using Microsoft.Xna.Framework.Audio;
      37  using Microsoft.Xna.Framework.Content;
      38  using Microsoft.Xna.Framework.GamerServices;
      39  using Microsoft.Xna.Framework.Graphics;
      40  using Microsoft.Xna.Framework.Input;
      41  using Microsoft.Xna.Framework.Media;
      42  using Microsoft.Xna.Framework.Net;
      43  using Microsoft.Xna.Framework.Storage;
      44
      45  using Knot3.Data;
      46  using Knot3.GameObjects;
      47  using Knot3.RenderEffects;
      48  using Knot3.Screens;
      49  using Knot3.Widgets;
      50
      51  #endregion
      52
      53  namespace Knot3.Core
      54  {
      55    /// <summary>
      56    /// Eine Spielkomponente, die in einem IGameScreen verwendet wird und eine b
      57    /// </summary>
      58    public abstract class GameScreenComponent : GameComponent, IGameScreenCompon
      59    {
      60      #region Properties
      61
      62      /// <summary>
      63      /// Die Zeichen- und Eingabeprioritt.
      64      /// </summary>
16    65      public DisplayLayer Index { get; set; }
      66
      67      /// <summary>
      68      /// Der zugewiesene Spielzustand.
      69      /// </summary>
24    70      public IGameScreen Screen { get; set; }
      71
      72      #endregion
      73
      74      #region Constructors
      75
      76      /// <summary>
      77      /// Erzeugt eine neue Instanz eines IGameScreenComponent-Objekts und initi
      78      /// </summary>
16    79      public GameScreenComponent (IGameScreen screen, DisplayLayer index)
      80      : base (screen.Game)
16    81      {
16    82        this.Screen = screen;
16    83        this.Index = index;
16    84      }
      85
      86      #endregion
      87
      88      #region Methods
      89
      90      /// <summary>
      91      /// Gibt Spielkomponenten zurck, die in dieser Spielkomponente enthalten
      92      /// [returntype=IEnumerable<IGameScreenComponent>]
      93      /// </summary>
      94      public virtual IEnumerable<IGameScreenComponent> SubComponents (GameTime G
0     95      {
0     96        yield break;
```

```
        }

        #endregion
      }
    }
```

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:	86

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.Linq;
34
35 using Microsoft.Xna.Framework;
```

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

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:	82

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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`
	36	`using Microsoft.Xna.Framework.Audio;`
	37	`using Microsoft.Xna.Framework.Content;`

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

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:	115

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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`
	36	`using Microsoft.Xna.Framework.Audio;`
	37	`using Microsoft.Xna.Framework.Content;`

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

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

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:	100%
Covered lines:	13
Uncovered lines:	0
Coverable lines:	13
Total lines:	98

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.Linq;
    34
    35  using Microsoft.Xna.Framework;
    36  using Microsoft.Xna.Framework.Audio;
    37  using Microsoft.Xna.Framework.Content;
    38  using Microsoft.Xna.Framework.GamerServices;
    39  using Microsoft.Xna.Framework.Graphics;
    40  using Microsoft.Xna.Framework.Input;
    41  using Microsoft.Xna.Framework.Media;
    42  using Microsoft.Xna.Framework.Net;
```

```
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Data;
46  using Knot3.GameObjects;
47  using Knot3.Platform;
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    /// Eine statische Klasse, die eine Referenz auf die zentrale Einstellungsda
59    /// </summary>
60    public static class Options
61    {
62      #region Properties
63
64      /// <summary>
65      /// Die zentrale Einstellungsdatei des Spiels.
66      /// </summary>
67      public static ConfigFile Default
68      {
1   69        get {
2   70          if (_default == null) {
1   71            _default = new ConfigFile (SystemInfo.SettingsDirectory + SystemInfo
1   72          }
1   73          return _default;
1   74        }
75        set {
76          _default = value;
77        }
78      }
79
80      private static ConfigFile _default;
4   81
5   82      public static ConfigFile Models
1   83      {
1   84        get {
85          if (_models == null) {
1   86            String seperatorString = SystemInfo.PathSeparator.ToString ();
4   87            _models = new ConfigFile (SystemInfo.BaseDirectory + seperatorString
4   88                                      + "Content" + seperatorString + "models.in
89          }
90          return _models;
91        }
92      }
93
94      private static ConfigFile _models;
95
96      #endregion
97    }
98  }
```

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:	13.5%
Covered lines:	23
Uncovered lines:	147
Coverable lines:	170
Total lines:	380

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
Update(...)	4	0	0
Draw(...)	3	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.Linq;
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Data;
46  using Knot3.GameObjects;
47  using Knot3.RenderEffects;
48  using Knot3.Screens;
49  using Knot3.Utilities;
50  using Knot3.Widgets;
51
52  #endregion
53
54  namespace Knot3.Core
55  {
56    /// <summary>
57    /// Reprsentiert eine Spielwelt, in der sich 3D-Modelle befinden und gezeic
58    /// </summary>
59    public sealed class World : DrawableGameScreenComponent, IEnumerable<IGameOb
60    {
61      #region Properties
62
63      /// <summary>
64      /// Die Kamera dieser Spielwelt.
65      /// </summary>
66      public Camera Camera
67      {
8   68        get {
8   69          return _camera;
8   70        }
0   71        set {
0   72          _camera = value;
0   73          useInternalCamera = false;
0   74        }
75      }
76
77      private Camera _camera;
4   78      private bool useInternalCamera = true;
79
80      /// <summary>
81      /// Die Liste von Spielobjekten.
82      /// </summary>
4   83      public HashSet<IGameObject> Objects { get; set; }
84
85      private IGameObject _selectedObject;
```

```
      86
      87      /// <summary>
      88      /// Das aktuell ausgewhlte Spielobjekt.
      89      /// </summary>
      90      public IGameObject SelectedObject
      91      {
0     92        get {
0     93          return _selectedObject;
0     94        }
0     95        set {
0     96          if (_selectedObject != value) {
0     97            _selectedObject = value;
0     98            SelectionChanged (_selectedObject);
0     99            Redraw = true;
0    100          }
0    101        }
     102      }
     103
     104      public float SelectedObjectDistance
     105      {
0    106        get {
0    107          if (SelectedObject != null) {
0    108            Vector3 toTarget = SelectedObject.Center () - Camera.Position;
0    109            return toTarget.Length ();
     110          }
0    111          else {
0    112            return 0;
     113          }
0    114        }
     115      }
     116
     117      /// <summary>
     118      /// Der aktuell angewendete Rendereffekt.
     119      /// </summary>
4    120      public IRenderEffect CurrentEffect { get; set; }
     121
     122      /// <summary>
     123      /// Wird ausgelst, wenn das selektierte Spielobjekt gendert wurde.
     124      /// </summary>
4    125      public Action<IGameObject> SelectionChanged = (o) => {};
     126
     127      /// <summary>
     128      /// Gibt an, ob die Spielwelt im folgenden Frame neugezeichnet wird
     129      /// oder nur der letzte Frame wiedergegeben wird.
     130      /// </summary>
0    131      public bool Redraw { get; set; }
     132
     133      /// <summary>
     134      /// Wird ausgelst, wenn die Spielwelt neu gezeichnet wird.
     135      /// </summary>
4    136      public Action OnRedraw = () => {};
     137
     138      /// <summary>
     139      /// Die Ausmae der Welt auf dem Screen.
     140      /// </summary>
4    141      public Bounds Bounds { get; private set; }
     142
     143      #endregion
     144
     145      #region Constructors
     146
```

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

```
      208     /// </summary>
      209     public override void Update (GameTime time)
0     210     {
0     211       if (!Options.Default ["video", "selectiveRendering", false]) {
0     212         Redraw = true;
0     213       }
0     214       if (Screen.PostProcessingEffect is FadeEffect) {
0     215         Redraw = true;
0     216       }
      217
      218       // run the update method on all game objects
0     219       foreach (IGameObject obj in Objects) {
0     220         obj.Update (time);
0     221       }
0     222     }
      223
4     224     private Dictionary<Point,Dictionary<Vector4, Viewport>> viewportCache
      225         = new Dictionary<Point,Dictionary<Vector4, Viewport>> ();
      226
      227     public Viewport Viewport
      228     {
3     229       get {
      230         // when we have a graphics device
3     231         if (Screen.Device != null) {
0     232           PresentationParameters pp = Screen.Device.PresentationParameters;
0     233           Point resolution = new Point (pp.BackBufferWidth, pp.BackBufferHeigh
0     234           Vector4 key = Bounds.Vector4;
0     235           if (!viewportCache.ContainsKey (resolution)) {
0     236             viewportCache [resolution] = new Dictionary<Vector4, Viewport> ();
0     237           }
0     238           if (!viewportCache [resolution].ContainsKey (key)) {
0     239             Rectangle subScreen = Bounds.Rectangle;
0     240             viewportCache [resolution] [key] = new Viewport (subScreen.X, subS
      241               MinDepth = 0,
      242               MaxDepth = 1
      243             };
0     244           }
0     245           return viewportCache [resolution] [key];
      246         }
      247         // for unit tests
3     248         else {
3     249           return Screen.Viewport;
      250         }
3     251       }
      252     }
      253
      254     /// <summary>
      255     /// Ruft auf allen Spielobjekten die Draw ()-Methode auf.
      256     /// </summary>
      257     public override void Draw (GameTime time)
0     258     {
0     259       if (Redraw) {
0     260         OnRedraw ();
0     261         Redraw = false;
      262
      263         //Screen.BackgroundColor = CurrentEffect is CelShadingEffect ? Color.C
      264
      265         // begin the knot render effect
0     266         CurrentEffect.Begin (time);
      267
0     268         foreach (IGameObject obj in Objects) {
```

```
0   269          obj.World = this;
0   270          obj.Draw (time);
0   271        }
    272
    273        // end of the knot render effect
0   274        CurrentEffect.End (time);
0   275      }
0   276      else {
0   277        CurrentEffect.DrawLastFrame (time);
0   278      }
0   279    }
    280
    281    /// <summary>
    282    /// Liefert einen Enumerator ber die Spielobjekte dieser Spielwelt.
    283    /// [returntype=IEnumerator<IGameObject>]
    284    /// </summary>
    285    public IEnumerator<IGameObject> GetEnumerator ()
0   286    {
0   287      foreach (IGameObject obj in flat (Objects)) {
0   288        yield return obj;
0   289      }
0   290    }
    291
    292    private IEnumerable<IGameObject> flat (IEnumerable<IGameObject> enumerable
0   293    {
0   294      foreach (IGameObject obj in enumerable) {
0   295        if (obj is IEnumerable<IGameObject>) {
0   296          foreach (IGameObject subobj in flat (obj as IEnumerable<IGameObject>
0   297            yield return subobj;
0   298          }
0   299        }
0   300        else {
0   301          yield return obj;
0   302        }
0   303      }
0   304    }
    305    // Explicit interface implementation for nongeneric interface
    306    IEnumerator IEnumerable.GetEnumerator ()
0   307    {
0   308      return GetEnumerator (); // Just return the generic version
0   309    }
    310
    311    public override IEnumerable<IGameScreenComponent> SubComponents (GameTime
0   312    {
0   313      foreach (DrawableGameScreenComponent component in base.SubComponents (ti
0   314        yield return component;
0   315      }
0   316      if (useInternalCamera) {
0   317        yield return Camera;
0   318      }
0   319    }
    320
    321    /// <summary>
    322    /// Gibt einen Iterator ber alle Spielobjekte zurck, der so sortiert ist
    323    /// Spielobjekte, die der angegebenen 2D-Position am nchsten sind, am Anf
    324    /// Dazu wird die 2D-Position in eine 3D-Position konvertiert.
    325    /// </summary>
    326    public IEnumerable<IGameObject> FindNearestObjects (Vector2 nearTo)
0   327    {
0   328      Dictionary<float, IGameObject> distances = new Dictionary<float, IGameOb
0   329      foreach (IGameObject obj in this) {
```

```
0   330          if (obj.Info.IsSelectable) {
    331            // Berechne aus der angegebenen 2D-Position eine 3D-Position
0   332            Vector3 position3D = Camera.To3D (
    333                                 position: nearTo,
    334                                 nearTo: obj.Center ()
    335                               );
    336            // Berechne die Distanz zwischen 3D-Mausposition und dem Spielobjekt
0   337            float distance = Math.Abs ((position3D - obj.Center ()).Length ());
0   338            distances [distance] = obj;
0   339          }
0   340        }
0   341        if (distances.Count > 0) {
0   342          IEnumerable<float> sorted = distances.Keys.OrderBy (k => k);
0   343          foreach (float where in sorted) {
0   344            yield return distances [where];
    345            // Log.Debug ("where=", where, " = ", distances [where].Center ());
0   346          }
0   347        }
0   348        else {
0   349          yield break;
    350        }
0   351      }
    352
    353      /// <summary>
    354      /// Gibt einen Iterator ber alle Spielobjekte zurck, der so sortiert ist
    355      /// Spielobjekte, die der angegebenen 3D-Position am nchsten sind, am Anf
    356      /// </summary>
    357      public IEnumerable<IGameObject> FindNearestObjects (Vector3 nearTo)
0   358      {
0   359        Dictionary<float, IGameObject> distances = new Dictionary<float, IGameOb
0   360        foreach (IGameObject obj in this) {
0   361          if (obj.Info.IsSelectable) {
    362            // Berechne die Distanz zwischen 3D-Mausposition und dem Spielobjekt
0   363            float distance = Math.Abs ((nearTo - obj.Center ()).Length ());
0   364            distances [distance] = obj;
0   365          }
0   366        }
0   367        if (distances.Count > 0) {
0   368          IEnumerable<float> sorted = distances.Keys.OrderBy (k => k);
0   369          foreach (float where in sorted) {
0   370            yield return distances [where];
0   371          }
0   372        }
0   373        else {
0   374          yield break;
    375        }
0   376      }
    377
    378      #endregion
    379    }
    380  }
```

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:	135

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.Linq;`
	34	

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Ein Objekt dieser Klasse reprsentiert eine Challenge.
  /// </summary>
  public sealed class Challenge
  {
    #region Properties

    /// <summary>
    /// Der Ausgangsknoten, den der Spieler in den Referenzknoten transformier
    /// </summary>
0   public Knot Start { get; private set; }

    /// <summary>
    /// Der Referenzknoten, in den der Spieler den Ausgangsknoten transformier
    /// </summary>
0   public Knot Target { get; private set; }

    /// <summary>
    /// Eine sortierte Bestenliste.
    /// </summary>
0   private SortedList<int, string> highscore { get; set; }

    /// <summary>
    /// Ein ffentlicher Enumerator, der die Bestenliste unabhngig von der da
    /// </summary>
0   public IEnumerable<KeyValuePair<string, int>> Highscore { get { return Met

    /// <summary>
    /// Die Metadaten der Challenge.
    /// </summary>
0   public ChallengeMetaData MetaData { get; private set; }

    /// <summary>
    /// Der Name der Challenge.
    /// </summary>
    public string Name
    {
0     get { return MetaData.Name; }
0     set { MetaData.Name = value; }
    }
```

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

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:	261

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.IO;
34  using System.Linq;
35  using System.Text;
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 Ionic.Zip;
48
49  using Knot3.Core;
50  using Knot3.Development;
51  using Knot3.GameObjects;
52  using Knot3.Input;
53  using Knot3.RenderEffects;
54  using Knot3.Screens;
55  using Knot3.Widgets;
56
57  #endregion
58
59  namespace Knot3.Data
60  {
61    /// <summary>
62    /// Implementiert das Speicherformat fr Challenges.
63    /// </summary>
64    public sealed class ChallengeFileIO : IChallengeIO
65    {
66      #region Properties
67
68      /// <summary>
69      /// Die fr eine Knoten-Datei gltigen Dateiendungen.
70      /// </summary>
0   71      public IEnumerable<string> FileExtensions
72      {
0   73        get {
0   74          yield return ".challenge";
0   75          yield return ".chl";
0   76          yield return ".chn";
0   77          yield return ".chg";
0   78          yield return ".chlng";
0   79        }
80      }
81
82      #endregion
83
84      #region Constructors
85
86      /// <summary>
87      /// Erstellt ein ChallengeFileIO-Objekt.
88      /// </summary>
0   89      public ChallengeFileIO ()
0   90      {
0   91      }
```

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

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

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

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:	85
Coverable lines:	85
Total lines:	240

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
GetHashCode()	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.Linq;
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Development;
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 zu einer Challenge.
61    /// </summary>
62    public class ChallengeMetaData
63    {
64      #region Properties
65
66      /// <summary>
67      /// Der Name der Challenge.
68      /// </summary>
69      public string Name
70      {
0   71        get {
0   72          return name;
0   73        }
0   74        set {
0   75          name = value;
0   76          if (Format == null) {
0   77            Format = new ChallengeFileIO ();
0   78          }
    79          string extension;
0   80          if (Format.FileExtensions.Any ()) {
0   81            extension = Format.FileExtensions.ElementAt (0);
0   82          }
0   83          else {
0   84            throw new ArgumentException ("Every implementation of IChallengeIO m
    85          }
0   86          Filename = SystemInfo.SavegameDirectory + SystemInfo.PathSeparator.ToS
0   87        }
    88      }
    89
    90      private string name;
```

```
0   91
    92      /// <summary>
    93      /// Der Ausgangsknoten, den der Spieler in den Referenzknoten transformier
    94      /// </summary>
0   95      public KnotMetaData Start { get; private set; }
    96
    97      /// <summary>
    98      /// Der Referenzknoten, in den der Spieler den Ausgangsknoten transformier
    99      /// </summary>
0   100     public KnotMetaData Target { get; private set; }
    101
    102     /// <summary>
    103     /// Das Format, aus dem die Metadaten der Challenge gelesen wurden oder nu
    104     /// </summary>
0   105     public IChallengeIO Format { get; private set; }
    106
    107     /// <summary>
    108     /// Der Dateiname, aus dem die Metadaten der Challenge gelesen wurden oder
    109     /// </summary>
0   110     public string Filename { get; private set; }
    111
    112     /// <summary>
    113     /// Ein ffentlicher Enumerator, der die Bestenliste unabhngig von der da
    114     /// </summary>
0   115     public IEnumerable<KeyValuePair<string, int>> Highscore { get { return hig
    116
    117     private List<KeyValuePair<string, int>> highscore;
    118
    119     public float AvgTime
    120     {
0   121       get {
0   122         if (   highscore != null
0   123                 && highscore.Any ()) {
0   124           float amount =0;
0   125           foreach (KeyValuePair<string, int> entry in highscore) {
0   126             amount += (float)entry.Value;
0   127           }
0   128           return amount/((float)highscore.Count);
    129         }
0   130         return 0f;
0   131       }
    132
0   133       private set {}
    134     }
    135
    136     public string FormatedAvgTime
    137     {
0   138       get {
0   139         float time = AvgTime;
0   140         Log.Debug (time);
0   141         if (time != 0f) {
0   142           return formatTime (time);
    143         }
0   144         return "Not yet set.";
0   145       }
0   146       private set {
0   147       }
    148     }
    149
    150     #endregion
    151
```

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

```
0   213      {
0   214        return (name ?? String.Empty).GetHashCode ();
0   215      }
    216
    217      public static bool operator == (ChallengeMetaData a, ChallengeMetaData b)
0   218      {
    219        // If both are null, or both are same instance, return true.
0   220        if (System.Object.ReferenceEquals (a, b)) {
0   221          return true;
    222        }
    223
    224        // If one is null, but not both, return false.
0   225        if (((object)a == null) || ((object)b == null)) {
0   226          return false;
    227        }
    228
    229        // Return true if the fields match:
0   230        return a.Equals (b);
0   231      }
    232
    233      public static bool operator != (ChallengeMetaData a, ChallengeMetaData b)
0   234      {
0   235        return !(a == b);
0   236      }
    237
    238      #endregion
    239    }
    240  }
```

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:	92.1%
Covered lines:	210
Uncovered lines:	18
Coverable lines:	228
Total lines:	400

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
ToString()	2	71.43	66.67
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	`/*`

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

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Eine doppelt verkettete Liste.
  /// </summary>
  public class CircleEntry<T> : IEnumerable<T>, ICollection<T>, IList<T>
  {
    public T Value { get; set; }

    public CircleEntry<T> Next { get; set; }

```

```
        65      public CircleEntry<T> Previous { get; set; }
        66
1077    67      public CircleEntry (T value)
1077    68      {
1077    69        Value = value;
1077    70        Previous = this;
1077    71        Next = this;
1077    72      }
        73
   7    74      private CircleEntry ()
   7    75      {
   7    76        Previous = this;
   7    77        Next = this;
   7    78      }
        79
  39    80      public CircleEntry (IEnumerable<T> list)
  39    81      {
  39    82        bool first = true;
  39    83        CircleEntry<T> inserted = this;
3318    84        foreach (T obj in list) {
1106    85          if (first) {
  39    86            Value = obj;
  39    87            Previous = this;
  39    88            Next = this;
  39    89          }
1028    90          else {
1028    91            inserted = inserted.InsertAfter (obj);
1028    92          }
1067    93          first = false;
1067    94        }
  39    95      }
        96
        97      public static CircleEntry<T> Empty
        98      {
   7    99        get {
   7   100          return new CircleEntry<T> ();
   7   101        }
       102      }
       103
       104      public CircleEntry<T> InsertBefore (T obj)
  47   105      {
  47   106        CircleEntry<T> insert = new CircleEntry<T> (obj);
  47   107        insert.Previous = this.Previous;
  47   108        insert.Next = this;
  47   109        this.Previous.Next = insert;
  47   110        this.Previous = insert;
  47   111        return insert;
  47   112      }
       113
       114      public CircleEntry<T> InsertAfter (T obj)
1030   115      {
       116        //Log.Debug (this, ".InsertAfter (", obj, ")");
1030   117        CircleEntry<T> insert = new CircleEntry<T> (obj);
1030   118        insert.Next = this.Next;
1030   119        insert.Previous = this;
1030   120        this.Next.Previous = insert;
1030   121        this.Next = insert;
1030   122        return insert;
1030   123      }
       124
       125      public void Remove ()
```

```
115    {
115      Previous.Next = Next;
115      Next.Previous = Previous;
115      Previous = null;
115      Next = null;
115    }

       private bool IsEmpty
       {
29       get {
29         return (Next == this || Next == null) && (Previous == this || Previous
29       }
       }

       public int Count
       {
27       get {
27         if (IsEmpty) {
0            return 0;
           }
27         else {
27           CircleEntry<T> current = this;
27           int count = 0;
244          do {
244            ++count;
244            current = current.Next;
244          }
244          while (current != this);
27           return count;
           }
27       }
       }

       public bool Contains (T obj, out IEnumerable<CircleEntry<T>> item)
199    {
199      item = Find (obj);
199      return item.Count () > 0;
199    }

       public bool Contains (Func<T, bool> func, out IEnumerable<CircleEntry<T>>
199    {
199      item = Find (func);
199      return item.Count () > 0;
199    }

       public bool Contains (T obj, out CircleEntry<T> item)
301    {
301      item = Find (obj).ElementAtOrDefault (0);
301      return item != null;
301    }

       public bool Contains (Func<T, bool> func, out CircleEntry<T> item)
100    {
100      item = Find (func).ElementAtOrDefault (0);
100      return item != null;
100    }

       public IEnumerable<CircleEntry<T>> Find (T obj)
707    {
27279    return Find ((t) => t.Equals (obj));
707    }
```

```
187
188     public IEnumerable<CircleEntry<T>> Find (Func<T, bool> func)
189     {
190       CircleEntry<T> current = this;
191       do {
192         if (func (current.Value)) {
193           yield return current;
194         }
195         current = current.Next;
196       }
197       while (current != this);
198       yield break;
199     }
200
201     public int IndexOf (T obj)
202     {
203       return IndexOf ((t) => t.Equals (obj));
204     }
205
206     public int IndexOf (Func<T, bool> func)
207     {
208       int i = 0;
209       CircleEntry<T> current = this;
210       do {
211         if (func (current.Value)) {
212           return i;
213         }
214         current = current.Next;
215         ++ i;
216       }
217       while (current != this);
218       return -1;
219     }
220
221     public IEnumerable<T> RangeTo (CircleEntry<T> other)
222     {
223       CircleEntry<T> current = this;
224       do {
225         yield return current.Value;
226         current = current.Next;
227       }
228       while (current != other.Next && current != this);
229     }
230
231     public IEnumerable<T> WayTo (T other)
232     {
233       CircleEntry<T> current = this;
234       while (!current.Value.Equals (other)) {
235         yield return current.Value;
236         current = current.Next;
237         if (current == this) {
238           break;
239         }
240       }
241     }
242
243     public IEnumerable<Tuple<T,T>> Pairs
244     {
245       get {
246         CircleEntry<T> current = this;
247         do {
```

1909	189
1909	190
52516	191
53929	192
1413	193
600	194
51703	195
51703	196
51703	197
1096	198
100	202
5150	203
100	204
200	207
200	208
200	209
10100	210
10300	211
200	212
9900	214
9900	215
9900	216
9900	217
0	218
200	219
4	222
4	223
16	224
16	225
16	226
16	227
16	228
4	229
1	232
1	233
3	234
1	235
1	236
1	237
0	238
1	240
1	241
0	245
0	246
0	247

```
     0          yield return Tuple.Create (current.Value, current.Next.Value);
     0          current = current.Next;
     0        }
     0        while (current != this);
     0      }
            }

            public IEnumerable<Tuple<T,T,T>> Triples
            {
    12        get {
    12          CircleEntry<T> current = this;
    52          do {
    52            yield return Tuple.Create (current.Previous.Value, current.Value, cu
    50            current = current.Next;
    50          }
    50          while (current != this);
    10        }
            }

            public IEnumerator<T> GetEnumerator ()
    91      {
    91        CircleEntry<T> current = this;
   826        do {
              //Log.Debug (this, " => ", current.Content);
   826        yield return current.Value;
   818        current = current.Next;
   818        }
   818        while (current != this);
    83      }

            // explicit interface implementation for nongeneric interface
            IEnumerator IEnumerable.GetEnumerator ()
     0      {
     0        return GetEnumerator (); // just return the generic version
     0      }

            public override string ToString ()
     2      {
     2        if (IsEmpty) {
     0          return "CircleEntry (" + Value.ToString () + ")";
              }
     2        else {
     2          return "CircleEntry.Empty";
              }
     2      }

            public static CircleEntry<T> operator + (CircleEntry<T> circle, int i)
   559      {
   559        CircleEntry<T> next = circle;
 40349        while (i > 0) {
 19895          next = next.Next;
 19895          i--;
 19895        }
   969        while (i < 0) {
   205          next = next.Previous;
   205          i++;
   205        }
   559        return next;
   559      }

            public T this [int index]
```

```
      {
218     get {
218       return (this + index).Value;
218     }
100     set {
100       (this + index).Value = value;
100     }
      }

      public static CircleEntry<T> operator - (CircleEntry<T> circle, int i)
30    {
30      return circle + (-i);
30    }

      public static CircleEntry<T> operator ++ (CircleEntry<T> circle)
189   {
189     return circle.Next;
189   }

      public static CircleEntry<T> operator -- (CircleEntry<T> circle)
11    {
11      return circle.Previous;
11    }

      public static implicit operator T (CircleEntry<T> circle)
700   {
700     return circle.Value;
700   }

3     public bool IsReadOnly { get { return false; } }

      public bool Contains (T obj)
102   {
102     CircleEntry<T> item = Find (obj).ElementAtOrDefault (0);
102     return item != null;
102   }

      public bool Remove (T value)
198   {
      CircleEntry<T> item;
297   if (Contains (value, out item)) {
99      item.Remove ();
99      return true;
      }
99    else {
99      return false;
      }
198   }

      public void RemoveAt (int i)
1     {
1       (this + i).Remove ();
1     }

      public void Insert (int i, T value)
0     {
0       (this + i).InsertBefore (value);
0     }

      public void Add (T value)
50    {
```

```
 56  370        if (Value == null) {
  6  371          Value = value;
  6  372        }
 44  373        else {
 44  374          InsertBefore (value);
 44  375        }
 50  376      }
     377
     378      public void Clear ()
  1  379      {
  1  380        Remove ();
  1  381        Next = Previous = this;
  1  382      }
     383
     384      public void CopyTo (T[] array, int start)
  1  385      {
303  386        foreach (T value in this) {
100  387          array.SetValue (value, start);
100  388          ++start;
100  389        }
  1  390      }
     391    }
     392
     393    public static class CircleExtensions
     394    {
     395      public static CircleEntry<T> ToCircle<T> (this IEnumerable<T> enumerable)
     396      {
     397        return new CircleEntry<T> (enumerable);
     398      }
     399    }
     400  }
```

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:	0%
Covered lines:	0
Uncovered lines:	3
Coverable lines:	3
Total lines:	400

Metrics

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

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.Linq;`
	34	
	35	`using Microsoft.Xna.Framework;`
	36	`using Microsoft.Xna.Framework.Audio;`
	37	`using Microsoft.Xna.Framework.Content;`

```
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Eine doppelt verkettete Liste.
  /// </summary>
  public class CircleEntry<T> : IEnumerable<T>, ICollection<T>, IList<T>
  {
    public T Value { get; set; }

    public CircleEntry<T> Next { get; set; }

    public CircleEntry<T> Previous { get; set; }

    public CircleEntry (T value)
    {
      Value = value;
      Previous = this;
      Next = this;
    }

    private CircleEntry ()
    {
      Previous = this;
      Next = this;
    }

    public CircleEntry (IEnumerable<T> list)
    {
      bool first = true;
      CircleEntry<T> inserted = this;
      foreach (T obj in list) {
        if (first) {
          Value = obj;
          Previous = this;
          Next = this;
        }
        else {
          inserted = inserted.InsertAfter (obj);
        }
        first = false;
      }
    }

    public static CircleEntry<T> Empty
    {
```

```
      get {
        return new CircleEntry<T> ();
      }
    }

    public CircleEntry<T> InsertBefore (T obj)
    {
      CircleEntry<T> insert = new CircleEntry<T> (obj);
      insert.Previous = this.Previous;
      insert.Next = this;
      this.Previous.Next = insert;
      this.Previous = insert;
      return insert;
    }

    public CircleEntry<T> InsertAfter (T obj)
    {
      //Log.Debug (this, ".InsertAfter (", obj, ")");
      CircleEntry<T> insert = new CircleEntry<T> (obj);
      insert.Next = this.Next;
      insert.Previous = this;
      this.Next.Previous = insert;
      this.Next = insert;
      return insert;
    }

    public void Remove ()
    {
      Previous.Next = Next;
      Next.Previous = Previous;
      Previous = null;
      Next = null;
    }

    private bool IsEmpty
    {
      get {
        return (Next == this || Next == null) && (Previous == this || Previous
      }
    }

    public int Count
    {
      get {
        if (IsEmpty) {
          return 0;
        }
        else {
          CircleEntry<T> current = this;
          int count = 0;
          do {
            ++count;
            current = current.Next;
          }
          while (current != this);
          return count;
        }
      }
    }

    public bool Contains (T obj, out IEnumerable<CircleEntry<T>> item)
```

```
    {
      item = Find (obj);
      return item.Count () > 0;
    }

    public bool Contains (Func<T, bool> func, out IEnumerable<CircleEntry<T>>
    {
      item = Find (func);
      return item.Count () > 0;
    }

    public bool Contains (T obj, out CircleEntry<T> item)
    {
      item = Find (obj).ElementAtOrDefault (0);
      return item != null;
    }

    public bool Contains (Func<T, bool> func, out CircleEntry<T> item)
    {
      item = Find (func).ElementAtOrDefault (0);
      return item != null;
    }

    public IEnumerable<CircleEntry<T>> Find (T obj)
    {
      return Find ((t) => t.Equals (obj));
    }

    public IEnumerable<CircleEntry<T>> Find (Func<T, bool> func)
    {
      CircleEntry<T> current = this;
      do {
        if (func (current.Value)) {
          yield return current;
        }
        current = current.Next;
      }
      while (current != this);
      yield break;
    }

    public int IndexOf (T obj)
    {
      return IndexOf ((t) => t.Equals (obj));
    }

    public int IndexOf (Func<T, bool> func)
    {
      int i = 0;
      CircleEntry<T> current = this;
      do {
        if (func (current.Value)) {
          return i;
        }
        current = current.Next;
        ++ i;
      }
      while (current != this);
      return -1;
    }

```

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

```
    }

    public override string ToString ()
    {
      if (IsEmpty) {
        return "CircleEntry (" + Value.ToString () + ")";
      }
      else {
        return "CircleEntry.Empty";
      }
    }

    public static CircleEntry<T> operator + (CircleEntry<T> circle, int i)
    {
      CircleEntry<T> next = circle;
      while (i > 0) {
        next = next.Next;
        i--;
      }
      while (i < 0) {
        next = next.Previous;
        i++;
      }
      return next;
    }

    public T this [int index]
    {
      get {
        return (this + index).Value;
      }
      set {
        (this + index).Value = value;
      }
    }

    public static CircleEntry<T> operator - (CircleEntry<T> circle, int i)
    {
      return circle + (-i);
    }

    public static CircleEntry<T> operator ++ (CircleEntry<T> circle)
    {
      return circle.Next;
    }

    public static CircleEntry<T> operator -- (CircleEntry<T> circle)
    {
      return circle.Previous;
    }

    public static implicit operator T (CircleEntry<T> circle)
    {
      return circle.Value;
    }

    public bool IsReadOnly { get { return false; } }

    public bool Contains (T obj)
    {
      CircleEntry<T> item = Find (obj).ElementAtOrDefault (0);
```

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

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:	64.1%
Covered lines:	50
Uncovered lines:	28
Coverable lines:	78
Total lines:	252

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	1	100	100
FromAxis(...)	5	0	0
FromString(...)	3	0	0
ToString()	1	100	100
op_Addition(...)	1	100	100
op_Subtraction(...)	1	0	0
op_Division(...)	1	100	100
op_Multiply(...)	1	0	0
op_Equality(...)	4	100	85.71
op_Inequality(...)	1	100	100
Equals(...)	2	100	66.67
Equals(...)	5	0	0
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
GetHashCode()	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
```

```
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
 * SOFTWARE.
 */

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Eine Wertesammlung der mglichen Richtungen in einem dreidimensionalen R
  /// Wird benutzt, damit keine ungltigen Kantenrichtungen angegeben werden k
  /// Dies ist eine Klasse und kein Enum, kann aber
  /// uneingeschrnkt wie eines benutzt werden (Typesafe Enum Pattern).
  /// </summary>
  public sealed class Direction : IEquatable<Direction>
  {
    #region Enumeration Values

    /// <summary>
    /// Links.
    /// </summary>
1   public static readonly Direction Left = new Direction (Vector3.Left, "Left
    /// <summary>
    /// Rechts.
    /// </summary>
1   public static readonly Direction Right = new Direction (Vector3.Right, "Ri
    /// <summary>
    /// Hoch.
    /// </summary>
1   public static readonly Direction Up = new Direction (Vector3.Up, "Up");
    /// <summary>
    /// Runter.
    /// </summary>
1   public static readonly Direction Down = new Direction (Vector3.Down, "Down
    /// <summary>
```

```
        /// Vorwrts.
        /// </summary>
1       public static readonly Direction Forward = new Direction (Vector3.Forward,
        /// <summary>
        /// Rckwrts.
        /// </summary>
1       public static readonly Direction Backward = new Direction (Vector3.Backwar
        /// <summary>
        /// Keine Richtung.
        /// </summary>
1       public static readonly Direction Zero = new Direction (Vector3.Zero, "Zero

        #endregion

        #region Static Attributes

1       public static readonly Direction[] Values = {
          Left, Right, Up, Down, Forward,  Backward
        };
1       private static readonly Dictionary<Direction, Direction> ReverseMap
            = new Dictionary<Direction, Direction> ()
        {
          { Left, Right }, { Right, Left },
          { Up, Down }, { Down, Up },
          { Forward, Backward }, { Backward, Forward },
          { Zero, Zero }
        };

1       private static readonly Dictionary<Direction, Axis> AxisMap
            = new Dictionary<Direction, Axis> ()
        {
          { Left, Axis.X }, { Right, Axis.X },
          { Up, Axis.Y }, { Down, Axis.Y },
          { Forward, Axis.Z }, { Backward, Axis.Z },
          { Zero, Axis.Zero }
        };

        #endregion

        #region Properties

1947    public Vector3 Vector { get; private set; }

356     public string Description { get; private set; }

183     public Direction Reverse { get { return ReverseMap [this]; } }

24      public Axis Axis { get { return AxisMap[this]; } }

        #endregion

        #region Constructors

7       private Direction (Vector3 vector, string desciption)
7       {
7         Vector = vector;
7         Description = desciption;
7       }

        #endregion

```

```
        144     #region Methods and Operators
        145
        146     public static Direction FromAxis (Axis axis)
   0    147     {
   0    148       return axis == Axis.X ? Right : axis == Axis.Y ? Up : axis == Axis.Z ? B
   0    149     }
        150
        151     public static Direction FromString (string str)
   0    152     {
   0    153       foreach (Direction direction in Values) {
   0    154         if (str.ToLower () == direction.Description.ToLower ()) {
   0    155           return direction;
        156         }
   0    157       }
   0    158       return null;
   0    159     }
        160
        161     public override string ToString ()
 108    162     {
 108    163       return Description;
 108    164     }
        165
        166     public static Vector3 operator + (Vector3 v, Direction d)
 316    167     {
 316    168       return v + d.Vector;
 316    169     }
        170
        171     public static Vector3 operator - (Vector3 v, Direction d)
   0    172     {
   0    173       return v - d.Vector;
   0    174     }
        175
        176     public static Vector3 operator / (Direction d, float i)
 599    177     {
 599    178       return d.Vector / i;
 599    179     }
        180
        181     public static Vector3 operator * (Direction d, float i)
   0    182     {
   0    183       return d.Vector * i;
   0    184     }
        185
        186     public static bool operator == (Direction a, Direction b)
 788    187     {
        188       // If both are null, or both are same instance, return true.
1078    189       if (System.Object.ReferenceEquals (a, b)) {
 290    190         return true;
        191       }
        192
        193       // If one is null, but not both, return false.
 571    194       if (((object)a == null) || ((object)b == null)) {
  73    195         return false;
        196       }
        197
        198       // Return true if the fields match:
 425    199       return a.Vector == b.Vector;
 788    200     }
        201
        202     public static bool operator != (Direction d1, Direction d2)
 105    203     {
 105    204       return !(d1 == d2);
```

```
105   205      }
      206
      207      public bool Equals (Direction other)
 73   208      {
 73   209        return other != null && Vector == other.Vector;
 73   210      }
      211
      212      public override bool Equals (object other)
  0   213      {
  0   214        if (other == null) {
  0   215          return false;
      216        }
  0   217        else if (other is Direction) {
  0   218          return Equals (other as Direction);
      219        }
  0   220        else if (other is Vector3) {
  0   221          return Vector.Equals ((Vector3)other);
      222        }
  0   223        else if (other is string) {
  0   224          return Description.Equals ((string)other);
      225        }
  0   226        else {
  0   227          return false;
      228        }
  0   229      }
      230
      231      public static implicit operator string (Direction direction)
  8   232      {
  8   233        return direction.Description;
  8   234      }
      235
      236      public static implicit operator Vector3 (Direction direction)
 21   237      {
 21   238        return direction.Vector;
 21   239      }
      240
      241      public override int GetHashCode ()
233   242      {
233   243        return Description.GetHashCode ();
233   244      }
      245
      246      #endregion
      247    }
      248
      249    public enum Axis {
      250      X, Y, Z, Zero
      251    }
      252  }
```

Knot3.Data.DirectionHelper

Summary

Class:	Knot3.Data.DirectionHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\DirectionHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	8
Coverable lines:	8
Total lines:	70

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
ToDirection(...)	3	0	0
.cctor()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Data\DirectionHelper.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.Linq;
    34
    35  using Microsoft.Xna.Framework;
```

```
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  public static class DirectionHelper
  {
    public static Direction ToDirection (this Vector3 vector)
    {
      foreach (Direction direction in Direction.Values) {
        if (direction.Vector == vector) {
          return direction;
        }
      }
      return Direction.Zero;
    }

    public static Axis[] Axes = new Axis[] { Axis.X, Axis.Y, Axis.Z };
  }
}
```

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:	66.2%
Covered lines:	57
Uncovered lines:	29
Coverable lines:	86
Total lines:	243

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	66.67
Equals(...)	6	31.58	27.27
GetHashCode()	1	100	100
ToString()	1	0	0
op_Implicit(...)	1	0	0
op_Implicit(...)	1	100	100
op_Implicit(...)	1	100	100
RandomColor()	1	100	100
RandomColor(...)	1	0	0
RandomEdge()	6	0	0
Clone()	1	0	0
.cctor()	1	100	100

File(s)

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

```
#region Copyright

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

```
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN T
 * SOFTWARE.
 */

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Eine Kante eines Knotens, die aus einer Richtung und einer Farbe, sowie
  /// </summary>
  public sealed class Edge : IEquatable<Edge>, ICloneable
  {
    #region Properties

    /// <summary>
    /// Die Farbe der Kante.
    /// </summary>
    public Color Color { get; set; }

    /// <summary>
    /// Die Richtung der Kante.
    /// </summary>
    public Direction Direction { get; private set; }

    /// <summary>
    /// Die Liste der Flchennummern, die an die Kante angrenzen.
    /// </summary>
    public HashSet<int> Rectangles { get; private set; }

    private int id;
    private static int previousId = 0;

    #endregion

```

```
        83      #region Constructors
        84
        85      /// <summary>
        86      /// Erstellt eine neue Kante mit der angegebenen Richtung.
        87      /// </summary>
128     88      public Edge (Direction direction)
128     89      {
128     90        Direction = direction;
128     91        Color = DefaultColor;
128     92        id = ++previousId;
128     93        Rectangles = new HashSet<int> ();
128     94      }
        95
        96      /// <summary>
        97      /// Erstellt eine neue Kante mit der angegebenen Richtung und Farbe.
        98      /// </summary>
18      99      public Edge (Direction direction, Color color)
18     100      {
18     101        Direction = direction;
18     102        Color = color;
18     103        id = ++previousId;
18     104        Rectangles = new HashSet<int>();
18     105      }
       106
       107      #endregion
       108
       109      #region Methods
       110
       111      public static bool operator == (Edge a, Edge b)
67     112      {
       113        // If both are null, or both are same instance, return true.
68     114        if (System.Object.ReferenceEquals (a, b)) {
1      115          return true;
       116        }
       117
       118        // If one is null, but not both, return false.
129    119        if (((object)a == null) || ((object)b == null)) {
63     120          return false;
       121        }
       122
       123        // Return true if the fields match:
3      124        return a.id == b.id;
67     125      }
       126
       127      public static bool operator != (Edge a, Edge b)
65     128      {
65     129        return !(a == b);
65     130      }
       131
       132      public bool Equals (Edge other)
63     133      {
63     134        return other != null && this.id == other.id;
63     135      }
       136
       137      public override bool Equals (object other)
29     138      {
29     139        if (other == null) {
0      140          return false;
       141        }
58     142        else if (other is Edge) {
29     143          return Equals (other as Edge);
```

```
      }
      else if (other is Direction) {
        return Direction.Equals (other as Direction);
      }
      else if (other is Vector3) {
        return Direction.Vector.Equals ((Vector3)other);
      }
      else if (other is Color) {
        return Color.Equals ((Color)other);
      }
      else {
        return false;
      }
    }

    public override int GetHashCode ()
    {
      return id;
    }

    public override string ToString ()
    {
      return Direction + "/" + id.ToString ();
    }

    public static implicit operator Direction (Edge edge)
    {
      return edge.Direction;
    }

    public static implicit operator Vector3 (Edge edge)
    {
      return edge.Direction;
    }

    public static implicit operator Color (Edge edge)
    {
      return edge.Color;
    }

    #endregion

    #region Helper Methods

    private static Random r = new Random ();

    public static Color RandomColor ()
    {
      return Colors [r.Next () % Colors.Count];
    }

    public static Color RandomColor (GameTime time)
    {
      return Colors [(int)time.TotalGameTime.TotalSeconds % Colors.Count];
    }

    public static Edge RandomEdge ()
    {
      int i = r.Next () % 6;
      return i == 0 ? Left : i == 1 ? Right : i == 2 ? Up : i == 3 ? Down : i
    }
```

```
      205
      206     public object Clone ()
  0   207     {
  0   208       return new Edge (Direction, Color);
  0   209     }
      210
      211     #endregion
      212
      213     #region Static Properties
      214
  1   215     public static List<Color> Colors = new List<Color> ()
      216     {
      217       Color.Red, Color.Green, Color.Blue, Color.Yellow, Color.Orange
      218     };
  1   219     public static Color DefaultColor = RandomColor ();
      220
  0   221     public static Edge Zero { get { return new Edge (Direction.Zero); } }
      222
  0   223     public static Edge UnitX { get { return new Edge (Direction.Right); } }
      224
  0   225     public static Edge UnitY { get { return new Edge (Direction.Up); } }
      226
  0   227     public static Edge UnitZ { get { return new Edge (Direction.Backward); } }
      228
 96   229     public static Edge Up { get { return new Edge (Direction.Up); } }
      230
 84   231     public static Edge Down { get { return new Edge (Direction.Down); } }
      232
 96   233     public static Edge Right { get { return new Edge (Direction.Right); } }
      234
 96   235     public static Edge Left { get { return new Edge (Direction.Left); } }
      236
  6   237     public static Edge Forward { get { return new Edge (Direction.Forward); }
      238
  6   239     public static Edge Backward { get { return new Edge (Direction.Backward);
      240
      241     #endregion
      242   }
      243 }
```

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:	84.9%
Covered lines:	266
Uncovered lines:	47
Coverable lines:	313
Total lines:	643

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
ToString()	3	100	66.67
.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.IO;
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.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    /// Diese Klasse reprsentiert einen Knoten, bestehend aus einem Knoten-Meta
61    /// </summary>
62    public sealed class Knot : ICloneable, IEnumerable<Edge>, IEquatable<Knot>
63    {
64      #region Properties
65
66      /// <summary>
67      /// Der Name des Knotens, welcher auch leer sein kann.
68      /// Beim Speichern muss der Nutzer in diesem Fall zwingend einen nichtleer
69      /// Der Wert dieser Eigenschaft wird aus der \glqq Name\grqq -Eigenschaft
70      /// und bei nderungen wieder in diesem gespeichert.
71      /// Beim ndern dieser Eigenschaft wird automatisch auch der im Metadaten-
72      /// </summary>
73      public string Name
```

```
        {
30        get { return MetaData.Name; }
 3        set { MetaData.Name = value; }
        }

        /// <summary>
        /// Das Startelement der doppelt-verketteten Liste, in der die Kanten gesp
        /// </summary>
        private CircleEntry<Edge> startElement;

        /// <summary>
        /// Die Metadaten des Knotens.
        /// </summary>
51      public KnotMetaData MetaData { get; private set; }

        /// <summary>
        /// Ein Ereignis, das in der Move-Methode ausgelst wird, wenn sich die St
        /// </summary>
28      public Action EdgesChanged = () => {};

        /// <summary>
        /// Enthlt die aktuell vom Spieler selektierten Kanten in der Reihenfolge
        /// </summary>
24      public IEnumerable<Edge> SelectedEdges { get { return selectedEdges; } }

        /// <summary>
        /// Enthlt die selektierten Kanten.
        /// </summary>
        private HashSet<Edge> selectedEdges;

        /// <summary>
        /// WTF?!
        /// </summary>
        public int debugId;

        /// <summary>
        /// Wird aufgerufen, wenn sich die Selektion gendert hat.
        /// </summary>
39      public Action SelectionChanged = () => {};

        /// <summary>
        /// Enthlt die zuletzt selektierte Kante.
        /// </summary>
        private CircleEntry<Edge> lastSelected;

        /// <summary>
        /// Wird aufgerufen, wenn sich die Startkante gendert hat.
        /// </summary>
28      public Action<Vector3> StartEdgeChanged = (v) => {};

        /// <summary>
        /// Der Cache fr die Knotencharakteristik.
        /// </summary>
28      private KnotCharakteristic? CharakteristicCache = null;

35      public Vector3 OffSet { get; private set;}

        #endregion

        #region Constructors

```

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

```
      196
      197     /// <summary>
      198     /// Prft ob die gegeben Struktur einen gltigen Knoten darstellt.
      199     /// </summary>
      200     public bool IsValidStructure (IEnumerable<Direction> knot)
   27 201     {
   27 202       Vector3 position3D = Vector3.Zero;
   27 203       HashSet<Vector3> occupancy = new HashSet<Vector3> ();
   27 204       if (knot.Count () < 4) {
    0 205         return false;
      206       }
  795 207       foreach (Direction peek in knot) {
  238 208         if (occupancy.Contains (position3D + (peek / 2))) {
    0 209           return false;
      210         }
  238 211         else {
  238 212           occupancy.Add (position3D + (peek / 2));
  238 213           position3D += peek;
  238 214         }
  238 215       }
   28 216       if (position3D.DistanceTo (Vector3.Zero) > 0.00001f) {
    1 217         return false;
      218       }
   26 219       return true;
   27 220     }
      221
      222     private bool IsValidStructure (IEnumerable<Edge> edges)
    3 223     {
   43 224       return IsValidStructure (from e in edges select e.Direction);
    3 225     }
      226
      227     /// <summary>
      228     /// Verschiebt die aktuelle Kantenauswahl in die angegebene Richtung um di
      229     /// </summary>
      230     public bool TryMove (Direction direction, int distance, out Knot newknot)
    9 231     {
   12 232       if (direction == Direction.Zero || distance == 0) {
    3 233         newknot = this;
    3 234         return true;
      235       }
      236
    6 237       Log.Debug ("TryMove: direction = ", direction, ", distance = ", distance
   38 238       Log.Debug ("Current Knot #", startElement.Count, " = ", string.Join (",
      239
    6 240       HashSet<Edge> selected = new HashSet<Edge> (selectedEdges);
    6 241       CircleEntry<Edge> newCircle = CircleEntry<Edge>.Empty;
      242
  114 243       foreach (Tuple<Edge, Edge, Edge> triple in startElement.Triples) {
   32 244         Edge previousEdge = triple.Item1;
   32 245         Edge currentEdge = triple.Item2;
   32 246         Edge nextEdge = triple.Item3;
      247
   38 248         if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (p
   15 249           distance.Repeat (i => newCircle.Add (new Edge (direction: direction,
    6 250         }
      251
   32 252         newCircle.Add (currentEdge);
      253
   38 254         if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (n
   15 255           distance.Repeat (i => newCircle.Add (new Edge (direction: direction.
    6 256         }
```

```
32  257         }
    258
56  259         Log.Debug ("New Knot #", newCircle.Count, " = ", string.Join (", ", from
    260
 6  261         Vector3 localOffset = OffSet;
 6  262         CircleEntry<Edge> current = newCircle;
45  263         do {
55  264           if (current [- 1].Direction == current [- 2].Direction.Reverse) {
    265             // Selektierte nicht lschen
13  266             if (selected.Contains (current [- 1]) || selected.Contains (current
 3  267               Log.Debug ("Error: Selektierte nicht lschen");
 3  268               newknot = null;
 3  269               return false;
    270             }
 7  271             if (newCircle == current - 1) {
 0  272               localOffset += (current - 1).Value;
 0  273               newCircle = current;
 0  274             }
 8  275             else if (newCircle == current - 2) {
 1  276               localOffset += (current - 1).Value.Direction + (current - 1).Value
 1  277               newCircle = current;
 1  278             }
 7  279             (current - 2).Remove ();
 7  280             (current - 1).Remove ();
 7  281           }
42  282           ++ current;
42  283         }
42  284         while (current != newCircle);
    285
23  286         Log.Debug ("New Knot after Remove #", newCircle.Count, " = ", string.Joi
    287
 3  288         if (!IsValidStructure (newCircle)) {
 0  289           Log.Debug ("Error: newCircle ist keine valide Struktur");
 0  290           newknot = null;
 0  291           return false;
    292         }
 3  293         newknot = new Knot (MetaData, newCircle, selected, localOffset);
 3  294         return true;
 9  295       }
    296
    297       public Vector3 MoveCenterToZero ()
 1  298       {
 1  299         Vector3 position3D = Vector3.Zero;
 1  300         Dictionary<Vector3, Edge> occupancy = new Dictionary<Vector3, Edge>();
21  301         foreach (Edge edge in startElement) {
 6  302           occupancy.Add (position3D + (edge.Direction / 2), edge);
 6  303           position3D += edge;
 6  304         }
 1  305         Vector3 mid = Vector3.Zero;
21  306         foreach (KeyValuePair<Vector3,Edge> pos in occupancy) {
 6  307           mid += pos.Key;
 6  308         }
 1  309         mid /= startElement.Count;
 1  310         float minDistance = mid.Length ();
 1  311         Edge newStart = startElement.Value;
21  312         foreach (KeyValuePair<Vector3,Edge> pos in occupancy) {
 6  313           float testDistance = pos.Key.DistanceTo (mid);
 8  314           if (testDistance < minDistance) {
 2  315             newStart = pos.Value;
 2  316             minDistance = testDistance;
 2  317           }
```

```
6   318        }
1   319        Vector3 offset = Vector3.Zero;
6   320        foreach (Edge edge in startElement.WayTo (newStart)) {
1   321          offset += edge;
1   322        }
1   323        startElement.Contains (newStart, out startElement);
1   324        offset += OffSet;
1   325        OffSet = Vector3.Zero;
1   326        return offset;
1   327      }
    328
    329      /// <summary>
    330      /// Verschiebt die aktuelle Kantenauswahl in die angegebene Richtung um di
    331      /// </summary>
    332      public bool Move (Direction direction, int distance)
9   333      {
    334        Knot newKnot;
15  335        if (TryMove (direction, distance, out newKnot)) {
6   336          startElement = newKnot.startElement;
6   337          selectedEdges = newKnot.selectedEdges;
6   338          return true;
    339        }
3   340        else {
3   341          return false;
    342        }
9   343      }
    344
    345      /// <summary>
    346      /// Gibt an ob ein Move in diese Richtung berhaupt mglich ist.
    347      /// </summary>
    348      public bool IsValidDirection (Direction direction)
6   349      {
    350        // Nichts selektiert
6   351        if (selectedEdges.Count == 0) {
0   352          return false;
    353        }
    354        // Alles selektiert
6   355        if (selectedEdges.Count == startElement.Count) {
0   356          return true;
    357        }
    358
6   359        HashSet<Axis> axes = new HashSet<Axis> ();
76  360        foreach (Tuple<Edge, Edge, Edge> triple in startElement.Triples) {
20  361          Edge previousEdge = triple.Item1;
20  362          Edge currentEdge = triple.Item2;
20  363          Edge nextEdge = triple.Item3;
    364
    365          // Wenn Kante nach der Bewegung gelscht werden msste ist ein Zug nic
20  366          if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (p
1   367                  && currentEdge.Direction == direction.Reverse && previousEdge.
1   368            return false;
    369          }
    370          // Wenn Kante nach der Bewegung gelscht werden msste ist ein Zug nic
19  371          if (selectedEdges.Contains (currentEdge) && !selectedEdges.Contains (n
1   372                  && currentEdge.Direction == direction && nextEdge.Direction !=
1   373            return false;
    374          }
    375
22  376          if (selectedEdges.Contains (currentEdge)) {
4   377            axes.Add (currentEdge.Direction.Axis);
4   378          }
```

```
18  379        }
    380        // Wenn alle Kanten entlang einer Achse angeordnet sind und die Verschie
 4  381        if (axes.Count == 1 && axes.Contains (direction.Axis)) {
 0  382          return false;
    383        }
 4  384        return true;
 6  385      }
    386
    387      private void onEdgesChanged ()
 0  388      {
 0  389        CharakteristicCache = null;
 0  390        EdgesChanged ();
 0  391      }
    392
    393      /// <summary>
    394      /// Gibt die doppelt-verkettete Kantenliste als Enumerator zurck.
    395      /// </summary>
    396      public IEnumerator<Edge> GetEnumerator ()
36  397      {
36  398        return startElement.GetEnumerator ();
36  399      }
    400
    401      /// <summary>
    402      /// Speichert den Knoten unter dem Dateinamen in dem Dateiformat, das in d
    403      /// Enthalten entweder die Dateiname-Eigenschaft, die Dateiformat-Eigensch
    404      /// oder beide den Wert \glqq null\grqq, dann wird eine IOException geworf
    405      /// </summary>
    406      public void Save ()
 0  407      {
 0  408        if (MetaData.Format == null) {
 0  409          throw new IOException ("Error: Knot: MetaData.Format is null!");
    410        }
 0  411        else if (MetaData.Filename == null) {
 0  412          throw new IOException ("Error: Knot: MetaData.Filename is null!");
    413        }
 0  414        else {
 0  415          MetaData.Format.Save (this);
 0  416        }
 0  417      }
    418
    419      /// <summary>
    420      /// Erstellt eine vollstndige Kopie des Knotens, inklusive der Kanten-Dat
    421      /// </summary>
    422      public object Clone ()
 2  423      {
 2  424        CircleEntry<Edge> newCircle = new CircleEntry<Edge> (startElement as IEn
 2  425        KnotMetaData metaData = new KnotMetaData (
    426            name: MetaData.Name,
 0  427            countEdges: () => 0,
    428            format: MetaData.Format,
    429            filename: MetaData.Filename
    430        );
 2  431        return new Knot (metaData: metaData, edges: newCircle) {
    432          selectedEdges = new HashSet<Edge> (selectedEdges),
    433          EdgesChanged = null,
    434          SelectionChanged = null,
    435        };
 2  436      }
    437
    438      private void OnSelectionChanged ()
11  439      {
```

```
11  440         SelectionChanged ();
11  441       }
    442
    443       /// <summary>
    444       /// Fgt die angegebene Kante zur aktuellen Kantenauswahl hinzu.
    445       /// </summary>
    446       public void AddToSelection (Edge edge)
 5  447       {
 5  448         IEnumerable<CircleEntry<Edge>> found = startElement.Find (edge);
10  449         if (found.Any ()) {
10  450           if (!selectedEdges.Contains (edge)) {
 5  451             selectedEdges.Add (edge);
 5  452           }
 5  453           lastSelected = found.ElementAt (0);
 5  454         }
 5  455         OnSelectionChanged ();
 5  456       }
    457
    458       /// <summary>
    459       /// Entfernt die angegebene Kante von der aktuellen Kantenauswahl.
    460       /// </summary>
    461       public void RemoveFromSelection (Edge edge)
 2  462       {
 2  463         selectedEdges.Remove (edge);
 3  464         if (lastSelected.Value == edge) {
 1  465           lastSelected = null;
 1  466         }
 2  467         OnSelectionChanged ();
 2  468       }
    469
    470       /// <summary>
    471       /// Hebt die aktuelle Kantenauswahl auf.
    472       /// </summary>
    473       public void ClearSelection ()
 2  474       {
 2  475         selectedEdges.Clear ();
 2  476         lastSelected = null;
 2  477         OnSelectionChanged ();
 2  478       }
    479
    480       /// <summary>
    481       /// Fgt alle Kanten auf dem krzesten Weg zwischen der zuletzt ausgewhlt
    482       /// zur aktuellen Kantenauswahl hinzu. Sind beide Wege gleich lang,
    483       /// wird der Weg in Richtung der ersten Kante ausgewhlt.
    484       /// </summary>
    485       public void AddRangeToSelection (Edge selectedEdge)
 3  486       {
 4  487         if (lastSelected == null) {
 1  488           AddToSelection (selectedEdge);
 1  489           return;
    490         }
 2  491         CircleEntry<Edge> selectedCircle = null;
 4  492         if (startElement.Contains (selectedEdge, out selectedCircle) && selected
 2  493           List<Edge> forward = new List<Edge> (lastSelected.RangeTo (selectedCir
 2  494           List<Edge> backward = new List<Edge> (selectedCircle.RangeTo (lastSele
    495
 3  496           if (forward.Count < backward.Count) {
12  497             foreach (Edge e in forward) {
 5  498               if (!selectedEdges.Contains (e)) {
 2  499                 selectedEdges.Add (e);
 2  500               }
```

```
3   501            }
1   502          }
1   503          else {
9   504            foreach (Edge e in backward) {
3   505              if (!selectedEdges.Contains (e)) {
1   506                selectedEdges.Add (e);
1   507              }
2   508            }
1   509          }
2   510          lastSelected = selectedCircle;
2   511        }
2   512        OnSelectionChanged ();
3   513      }
    514
    515      /// <summary>
    516      /// Prft, ob die angegebene Kante in der aktuellen Kantenauswahl enthalte
    517      /// </summary>
    518      public Boolean IsSelected (Edge edge)
0   519      {
0   520        return selectedEdges.Contains (edge);
0   521      }
    522
    523      /// <summary>
    524      /// Gibt die doppelt-verkettete Kantenliste als Enumerator zurck.
    525      /// [name=IEnumerable.GetEnumerator]
    526      /// [keywords= ]
    527      /// </summary>
    528      IEnumerator IEnumerable.GetEnumerator ()
2   529      {
2   530        return GetEnumerator (); // just return the generic version
2   531      }
    532
    533      /// <summary>
    534      /// Speichert den Knoten unter dem angegebenen Dateinamen in dem angegeben
    535      /// </summary>
    536      public void Save (IKnotIO format, string filename)
0   537      {
0   538        KnotMetaData metaData = new KnotMetaData (MetaData.Name, () => MetaData.
0   539        Knot knotToSave = new Knot (metaData, startElement);
0   540        format.Save (knotToSave);
0   541      }
    542
    543      /// <summary>
    544      /// Prft, ob die rumliche Struktur identisch ist, unabhngig von dem Sta
    545      /// [parameters=Knot other]
    546      /// </summary>
    547      public bool Equals (Knot other)
12  548      {
12  549        KnotCharakteristic thisCharakteristik = Charakteristic ();
12  550        KnotCharakteristic otherCharakteristik = other.Charakteristic ();
18  551        if (thisCharakteristik.CountEdges != otherCharakteristik.CountEdges) {
6   552          return false;
    553        }
    554        // Bei Struktur im gleicher Richtung
12  555        if (thisCharakteristik.CharacteristicalEdge.Value.Direction == otherChar
6   556          CircleEntry<Edge> currentThisElement = thisCharakteristik.Characterist
6   557          CircleEntry<Edge> currentOtherElement = otherCharakteristik.Characteri
66  558          while (currentThisElement != thisCharakteristik.CharacteristicalEdge)
30  559            if (currentThisElement.Value.Direction != currentOtherElement.Value.
0   560              return false;
    561            }
```

```
30  562            currentThisElement++;
30  563            currentOtherElement++;
30  564          }
 6  565          return true;
    566        }
    567        // Bei Struktur in entgegengesetzter Richtung
 0  568        else if (thisCharakteristik.CharacteristicalEdge.Value.Direction == othe
 0  569          CircleEntry<Edge> currentThisElement = thisCharakteristik.Characterist
 0  570          CircleEntry<Edge> currentOtherElement = otherCharakteristik.Characteri
 0  571          while (currentThisElement != thisCharakteristik.CharacteristicalEdge)
 0  572            if (currentThisElement.Value.Direction != currentOtherElement.Value.
 0  573              return false;
    574            }
 0  575            currentThisElement++;
 0  576            currentOtherElement++;
 0  577          }
 0  578          return true;
    579        }
 0  580        else {
 0  581          return false;
    582        }
12  583      }
    584
    585      /// <summary>
    586      /// Gibt chrakteristische Werte zurck, die bei gleichen Knoten gleich sin
    587      /// Einmal als Key ein eindeutiges Circle\<Edge\> Element und als Value
    588      /// einen Charakteristischen Integer. Momentan die Anzahl der Kanten.
    589      /// </summary>
    590      private KnotCharakteristic Charakteristic ()
24  591      {
35  592        if (CharakteristicCache.HasValue) {
11  593          return CharakteristicCache.Value;
    594        }
    595
13  596        CircleEntry<Edge> charakteristikElement = startElement;
13  597        Vector3 position3D = startElement.Value.Direction;
13  598        Vector3 bestPosition3D = startElement.Value.Direction / 2;
13  599        CircleEntry<Edge> edgePointer = startElement.Next;
    600
13  601        int edgeCount = 1;
257 602        for (edgeCount = 1; edgePointer != startElement; edgePointer++, edgeCoun
77  603          Vector3 nextPosition3D = position3D + edgePointer.Value.Direction / 2;
77  604          if ((nextPosition3D.X < bestPosition3D.X)
    605                  || (nextPosition3D.X == bestPosition3D.X && nextPosition3D.Y <
27  606                  || (nextPosition3D.X == bestPosition3D.X && nextPosition3D.Y =
27  607            bestPosition3D = position3D + edgePointer.Value.Direction / 2;
27  608            charakteristikElement = edgePointer;
27  609          }
77  610          position3D += edgePointer.Value.Direction;
77  611        }
    612
13  613        CharakteristicCache = new KnotCharakteristic (charakteristikElement, edg
13  614        return CharakteristicCache.Value;
24  615      }
    616
    617      public override string ToString ()
 1  618      {
 1  619        return "Knot (name=" + Name + ",#edgecount=" + startElement.Count.ToStri
    620               + ",format=" + (MetaData.Format != null ? MetaData.ToString () :
    621               + ")";
 1  622      }
```

```

    #endregion

    #region Classes and Structs

    private struct KnotCharakteristic {
      public CircleEntry<Edge> CharacteristicalEdge { get; private set; }

      public int CountEdges { get; private set; }

      public KnotCharakteristic (CircleEntry<Edge> characteristicalEdge, int c
      : this ()
      {
        CharacteristicalEdge = characteristicalEdge;
        CountEdges = countEdges;
      }
    }

    #endregion
  }
}
```

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:	20.5%
Covered lines:	8
Uncovered lines:	31
Coverable lines:	39
Total lines:	154

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor()	1	100	100
Save(...)	2	0	0
Load(...)	2	0	0
LoadMetaData(...)	2	0	0
ToString()	1	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.IO;
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.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    /// Implementiert das Speicherformat fr Knoten.
61    /// </summary>
62    public sealed class KnotFileIO : IKnotIO
63    {
64      #region Properties
65
66      /// <summary>
67      /// Die fr eine Knoten-Datei gltigen Dateiendungen.
68      /// </summary>
64  69      public IEnumerable<string> FileExtensions
    70      {
64  71        get {
64  72          yield return ".knot";
 0  73          yield return ".knt";
 0  74        }
    75      }
    76
32  77      private Dictionary<string, Knot> KnotCache = new Dictionary<string, Knot>
32  78      private Dictionary<string, KnotMetaData> KnotMetaDataCache = new Dictionar
    79
    80      #endregion
    81
    82      #region Constructors
    83
    84      /// <summary>
    85      /// Erstellt ein KnotFileIO-Objekt.
    86      /// </summary>
32  87      public KnotFileIO ()
32  88      {
32  89      }
    90
    91      #endregion
    92
```

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

```
}
```

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:	60.3%
Covered lines:	32
Uncovered lines:	21
Coverable lines:	53
Total lines:	194

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	3	100	100
.ctor(...)	1	100	100
Equals(...)	3	0	0
Equals(...)	2	0	0
GetHashCode()	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.Linq;
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.GameObjects;
47  using Knot3.Input;
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.Data
57  {
58    /// <summary>
59    /// Enthlt Metadaten eines Knotens, die aus einer Spielstand-Datei schnelle
60    /// als der vollstndige Knoten. Dieses Objekt enthlt keine Datenstruktur z
61    /// sondern nur Informationen ber den Namen des Knoten und die Anzahl seine
62    /// dazugehriges Knoten-Objekt existieren, aber jedes Knoten-Objekt enthlt
63    /// </summary>
64    public class KnotMetaData : IEquatable<KnotMetaData>
65    {
66      #region Properties
67
68      /// <summary>
69      /// Der Anzeigename des Knotens, welcher auch leer sein kann.
70      /// Beim Speichern muss der Spieler in diesem Fall zwingend
71      /// einen nichtleeren Namen whlen. Wird ein neuer Anzeigename festgelegt,
72      /// dann wird der Dateiname ebenfalls auf einen neuen Wert gesetzt, unabh
73      /// ob er bereits einen Wert enthlt oder \glqq null\grqq ist.
74      /// Diese Eigenschaft kann ffentlich gelesen und gesetzt werden.
75      /// </summary>
76      public string Name
77      {
78        get {
79          return name;
80        }
81        set {
82          name = value;
83          if (Format == null) {
84            Format = new KnotFileIO ();
85          }
86          if (name != null && name.Length > 0) {
87            string extension;
88            if (Format.FileExtensions.Any ()) {
89              extension = Format.FileExtensions.ElementAt (0);
90            }
91            else {
```

39	78
39	79
39	80
33	81
33	82
65	83
32	84
32	85
65	86
	87
64	88
32	89
32	90
0	91

```
0    92              throw new ArgumentException ("Every implementation of IKnotIO must
     93            }
32   94            Filename = SystemInfo.SavegameDirectory + SystemInfo.PathSeparator.T
32   95          }
33   96        }
     97      }
     98
     99      private string name;
    100
    101      /// <summary>
    102      /// Das Format, aus dem die Metadaten geladen wurden.
    103      /// Es ist genau dann \glqq null\grqq, wenn die Metadaten nicht aus einer
    104      /// </summary>
220 105      public IKnotIO Format { get; private set; }
    106
    107      /// <summary>
    108      /// Ein Delegate, das die Anzahl der Kanten zurckliefert.
    109      /// Falls dieses Metadaten-Objekt Teil eines Knotens ist, gibt es dynamisc
    110      /// Kanten des Knoten-Objektes zurck. Anderenfalls gibt es eine statische
    111      /// die beim Einlesen der Metadaten vor dem Erstellen dieses Objektes gele
    112      /// </summary>
3   113      public int CountEdges { get { return countEdges (); } }
    114
    115      private Func<int> countEdges;
    116
    117      /// <summary>
    118      /// Falls die Metadaten aus einer Datei eingelesen wurden, enthlt dieses
    119      /// sonst \glqq null\grqq.
    120      /// </summary>
122 121      public string Filename { get; private set; }
    122
    123      #endregion
    124
    125      #region Constructors
    126
    127      /// <summary>
    128      /// Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knote
    129      /// und einer angegebenen Funktion, welche eine Kantenanzahl zurck gibt.
    130      /// Zustzlich wird der Dateiname oder das Speicherformat angegeben, aus d
    131      /// </summary>
32  132      public KnotMetaData (string name, Func<int> countEdges, IKnotIO format, st
32  133      {
32  134        Name = name;
32  135        this.countEdges = countEdges;
32  136        Format = format ?? Format;
32  137        Filename = filename ?? Filename;
32  138      }
    139
    140      /// <summary>
    141      /// Erstellt ein neues Knoten-Metadaten-Objekt mit einem angegebenen Knote
    142      /// und einer angegebenen Funktion, welche eine Kantenanzahl zurck gibt.
    143      /// </summary>
7   144      public KnotMetaData (string name, Func<int> countEdges)
7   145      {
7   146        this.name = name;
7   147        this.countEdges = countEdges;
7   148        Format = null;
7   149        Filename = null;
7   150      }
    151
    152      #endregion
```

```
153
154       #region Methods
155
156       public bool Equals (KnotMetaData other)
157       {
158         return other != null && name == other.name && countEdges () == other.cou
159       }
160
161       public override bool Equals (object other)
162       {
163         return other != null && Equals (other as KnotMetaData);
164       }
165
166       public override int GetHashCode ()
167       {
168         return (countEdges ().ToString () + (name ?? String.Empty)).GetHashCode
169       }
170
171       public static bool operator == (KnotMetaData a, KnotMetaData b)
172       {
173         // If both are null, or both are same instance, return true.
174         if (System.Object.ReferenceEquals (a, b)) {
175           return true;
176         }
177
178         // If one is null, but not both, return false.
179         if (((object)a == null) || ((object)b == null)) {
180           return false;
181         }
182
183         // Return true if the fields match:
184         return a.Equals (b);
185       }
186
187       public static bool operator != (KnotMetaData a, KnotMetaData b)
188       {
189         return !(a == b);
190       }
191
192       #endregion
193     }
194   }
```

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:	40.1%
Covered lines:	49
Uncovered lines:	73
Coverable lines:	122
Total lines:	264

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
ToString()	1	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.IO;
      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.RenderEffects;
      51  using Knot3.Screens;
      52  using Knot3.Widgets;
      53
      54  #endregion
      55
      56  namespace Knot3.Data
      57  {
      58    /// <summary>
      59    /// Diese Klasse reprsentiert einen Parser fr das Knoten-Austauschformat u
      60    /// eingelesenen Informationen wie den Namen des Knotens und die Kantenliste
      61    /// </summary>
      62    public sealed class KnotStringIO
      63    {
      64      #region Properties
      65
      66      /// <summary>
      67      /// Der Name der eingelesenen Knotendatei oder des zugewiesenen Knotenobje
      68      /// </summary>
 16   69      public string Name { get; set; }
      70
      71      private IEnumerable<string> edgeLines;
      72
      73      /// <summary>
      74      /// Die Kanten der eingelesenen Knotendatei oder des zugewiesenen Knotenob
      75      /// </summary>
  2   76      public IEnumerable<Edge> Edges
      77      {
  2   78        get {
  2   79          Log.Debug ("KnotStringIO.Edges[get] = ", edgeLines.Count ());
 10   80          foreach (string _line in edgeLines) {
  2   81            string line = _line;
  2   82            Edge edge = DecodeEdge (line [0]);
  0   83            line = line.Substring (1);
  0   84            if (line.StartsWith ("#")) {
  0   85              line = line.Substring (1);
  0   86            }
  0   87            edge.Color = DecodeColor (line.Substring (0, 8));
  0   88            line = line.Substring (8);
  0   89            if (line.StartsWith ("#")) {
```

```
0   90            line = line.Substring (1);
0   91          }
0   92          if (line.Length > 0) {
0   93            foreach (int rect in line.Split (',').Select (int.Parse).ToList ()
0   94              edge.Rectangles.Add (rect);
0   95            }
0   96          }
0   97          yield return edge;
0   98        }
0   99      }
0  100      set {
0  101        Log.Debug ("KnotStringIO.Edges[set] = #", value.Count ());
0  102        try {
0  103          edgeLines = ToLines (value);
0  104        }
0  105        catch (Exception ex) {
0  106          Log.Debug (ex);
0  107        }
0  108      }
   109    }
   110
   111    /// <summary>
   112    /// Die Anzahl der Kanten der eingelesenen Knotendatei oder des zugewiesen
   113    /// </summary>
   114    public int CountEdges
   115    {
0  116      get {
0  117        return edgeLines.Where ((l) => l.Trim ().Length > 0).Count ();
0  118      }
   119    }
   120
   121    /// <summary>
   122    /// Erstellt aus den \glqq Name\grqq - und \glqq Edges\grqq -Eigenschaften
   123    /// die als Dateiinhalt in einer Datei eines Spielstandes einen gltigen K
   124    /// </summary>
   125    public string Content
   126    {
10 127      get {
10 128        return Name + "\n" + string.Join ("\n", edgeLines);
10 129      }
 4 130      set {
 8 131        if (value.Trim ().Contains ("\n")) {
 4 132          string[] parts = value.Split (new char[] {'\r','\n'}, StringSplitOpt
 4 133          Name = parts [0];
 4 134          edgeLines = parts.Skip (1);
 4 135        }
0  136        else {
0  137          Name = value;
0  138          edgeLines = new string[] {};
0  139        }
 4 140      }
   141    }
   142
   143    #endregion
   144
   145    #region Constructors
   146
   147    /// <summary>
   148    /// Liest das in der angegebenen Zeichenkette enthaltene Dateiformat ein.
   149    /// so werden die \glqq Name\grqq - und \glqq Edges\grqq -Eigenschaften au
   150    /// Enthlt es einen ungltigen Knoten, so wird eine IOException geworfen
```

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

```
60   212          return  'Y';
     213        }
120  214        else if (edge.Direction == Direction.Down) {
60   215          return  'y';
     216        }
0    217        else if (edge.Direction == Direction.Backward) {
0    218          return  'Z';
     219        }
0    220        else if (edge.Direction == Direction.Forward) {
0    221          return  'z';
     222        }
0    223        else {
0    224          throw new IOException ("Failed to encode Edge: '" + edge + "'!");
     225        }
240  226      }
     227
     228      private static String EncodeColor (Color c)
240  229      {
240  230        return c.R.ToString ("X2") + c.G.ToString ("X2") + c.B.ToString ("X2") +
240  231      }
     232
     233      private static Color DecodeColor (string hexString)
0    234      {
0    235        if (hexString.StartsWith ("#")) {
0    236          hexString = hexString.Substring (1);
0    237        }
0    238        uint hex = uint.Parse (hexString, System.Globalization.NumberStyles.HexN
0    239        Color color = Color.White;
0    240        if (hexString.Length == 8) {
0    241          color.R = (byte)(hex >> 24);
0    242          color.G = (byte)(hex >> 16);
0    243          color.B = (byte)(hex >> 8);
0    244          color.A = (byte)(hex);
0    245        }
0    246        else if (hexString.Length == 6) {
0    247          color.R = (byte)(hex >> 16);
0    248          color.G = (byte)(hex >> 8);
0    249          color.B = (byte)(hex);
0    250        }
0    251        else {
0    252          throw new IOException ("Invald hex representation of an ARGB or RGB co
     253        }
0    254        return color;
0    255      }
     256
     257      public override string ToString ()
0    258      {
0    259        return "KnotStringIO (length=" + Content.Length + ")";
0    260      }
     261
     262      #endregion
     263    }
     264  }
```

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:	76.1%
Covered lines:	51
Uncovered lines:	16
Coverable lines:	67
Total lines:	208

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
GetHashCode()	1	100	100
ToString()	1	100	100
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

```
#region Copyright

/*
 * Copyright (c) 2013-2014 Tobias Schulz, Maximilian Reuter, Pascal Knodel,
 *                         Gerd Augsburg, Christina Erler, Daniel Warzel
 *
 * Permission is hereby granted, free of charge, to any person obtaining a cop
 * of this software and associated documentation files (the "Software"), to de
 * in the Software without restriction, including without limitation the right
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * 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.Linq;
34
35 using Microsoft.Xna.Framework;
36 using Microsoft.Xna.Framework.Audio;
37 using Microsoft.Xna.Framework.Content;
38 using Microsoft.Xna.Framework.GamerServices;
39 using Microsoft.Xna.Framework.Graphics;
40 using Microsoft.Xna.Framework.Input;
41 using Microsoft.Xna.Framework.Media;
42 using Microsoft.Xna.Framework.Net;
43 using Microsoft.Xna.Framework.Storage;
44
45 using Knot3.Core;
46 using Knot3.GameObjects;
47 using Knot3.Input;
48 using Knot3.RenderEffects;
49 using Knot3.Screens;
50 using Knot3.Widgets;
51
52 #endregion
53
54 namespace Knot3.Data
55 {
56   /// <summary>
57   /// Eine Position im 3D-Raster. Die Werte fr alle drei Koordinaten sind Int
58   /// Eine Skalierung auf Koordinaten im 3D-Raum und damit einhergehend eine K
59   /// </summary>
60   public sealed class Node : IEquatable<Node>, ICloneable
61   {
62     #region Properties
63
64     /// <summary>
65     /// X steht fr eine x-Koordinate im dreidimensionalen Raster.
66     /// </summary>
152 67     public int X { get; private set; }
68
69     /// <summary>
70     /// Y steht fr eine y-Koordinate im dreidimensionalen Raster.
71     /// </summary>
152 72     public int Y { get; private set; }
73
74     /// <summary>
75     /// Z steht fr eine z-Koordinate im dreidimensionalen Raster.
76     /// </summary>
152 77     public int Z { get; private set; }
78
79     /// <summary>
80     /// Ein Skalierungswert.
81     /// </summary>
```

```
1   82      public static readonly int Scale = 100;
    83
    84      #endregion
    85
    86      #region Constructors
    87
    88      /// <summary>
    89      /// Erzeugt eine neue Instanz eines Node-Objekts und initialisiert diese m
    90      /// fr die x-, y- und z-Koordinate.
    91      /// </summary>
22  92      public Node (int x, int y, int z)
22  93      {
22  94        X = x;
22  95        Y = y;
22  96        Z = z;
22  97      }
    98
    99      #endregion
   100
   101      #region Methods
   102
   103      /// <summary>
   104      /// Liefert die x-, y- und z-Koordinaten im 3D-Raum als ein Vektor3 der Fo
   105      /// </summary>
   106      public Vector3 Vector
   107      {
22 108        get {
22 109          return new Vector3 (X * Scale, Y * Scale, Z * Scale);
22 110        }
   111      }
   112
   113      public static implicit operator Vector3 (Node node)
14 114      {
14 115        return node.Vector;
14 116      }
   117
   118      public Vector3 CenterBetween (Node other)
 4 119      {
 4 120        Vector3 positionFrom = this.Vector;
 4 121        Vector3 positionTo = other.Vector;
 4 122        return positionFrom + (positionTo - positionFrom) / 2;
 4 123      }
   124
   125      public static Node operator + (Node a, Vector3 b)
 1 126      {
 1 127        return new Node (a.X + (int)b.X, a.Y + (int)b.Y, a.Z + (int)b.Z);
 1 128      }
   129
   130      public static Vector3 operator - (Node a, Node b)
 1 131      {
 1 132        return new Vector3 (a.X - b.X, a.Y - b.Y, a.Z - b.Z);
 1 133      }
   134
   135      public static Node operator + (Node a, Direction b)
 0 136      {
 0 137        return new Node (a.X + (int)b.Vector.X, a.Y + (int)b.Vector.Y, a.Z + (in
 0 138      }
   139
   140      public static Node operator - (Node a, Direction b)
 0 141      {
 0 142        return new Node (a.X - (int)b.Vector.X, a.Y - (int)b.Vector.Y, a.Z - (in
```

```
0   143     }
        144
        145     public static Node operator + (Direction a, Node b)
0   146     {
0   147       return b+a;
0   148     }
        149
        150     public static Node operator - (Direction a, Node b)
0   151     {
0   152       return b-a;
0   153     }
        154
        155     public override int GetHashCode ()
35  156     {
35  157       return X * 10000 + Y * 100 + Z;
35  158     }
        159
        160     public override string ToString ()
1   161     {
1   162       return "(" + X.ToString () + "," + Y.ToString () + "," + Z.ToString () +
1   163     }
        164
        165     public object Clone ()
1   166     {
1   167       return new Node (X, Y, Z);
1   168     }
        169
        170     public static bool operator == (Node a, Node b)
2   171     {
        172       // If both are null, or both are same instance, return true.
2   173       if (System.Object.ReferenceEquals (a, b)) {
0   174         return true;
        175       }
        176
        177       // If one is null, but not both, return false.
2   178       if (((object)a == null) || ((object)b == null)) {
0   179         return false;
        180       }
        181
        182       // Return true if the fields match:
2   183       return a.X == b.X && a.Y == b.Y && a.Z == b.Z;
2   184     }
        185
        186     public static bool operator != (Node a, Node b)
1   187     {
1   188       return !(a == b);
1   189     }
        190
        191     public bool Equals (Node other)
32  192     {
32  193       return this.X == other.X && this.Y == other.Y && this.Z == other.Z;
32  194     }
        195
        196     public override bool Equals (object obj)
2   197     {
4   198       if (obj is Node) {
2   199         return Equals ((Node)obj);
        200       }
0   201       else {
0   202         return false;
        203       }
```

```
2    204        }
     205
     206        #endregion
     207      }
     208  }
```

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:	175

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

```
1   90      {
1   91        Edges = edges;
1   92        BuildIndex ();
1   93      }
    94
    95      #endregion
    96
    97      #region Methods
    98
    99      /// <summary>
   100      /// Gibt die Rasterposition des bergangs am Anfang der Kante zurck.
   101      /// </summary>
   102      public Node NodeBeforeEdge (Edge edge)
8  103      {
8  104        return (Node)fromMap [edge];
8  105      }
   106
   107      /// <summary>
   108      /// Gibt die Rasterposition des bergangs am Ende der Kante zurck.
   109      /// </summary>
   110      public Node NodeAfterEdge (Edge edge)
32 111      {
32 112        return (Node)toMap [edge];
32 113      }
   114
   115      public List<IJunction> JunctionsAtNode (Node node)
16 116      {
16 117        return junctionMap [node];
16 118      }
   119
   120      public List<IJunction> JunctionsBeforeEdge (Edge edge)
4  121      {
4  122        return junctionMap [NodeBeforeEdge (edge)];
4  123      }
   124
   125      public List<IJunction> JunctionsAfterEdge (Edge edge)
4  126      {
4  127        return junctionMap [NodeAfterEdge (edge)];
4  128      }
   129
   130      public IEnumerable<Node> Nodes
   131      {
0  132        get {
0  133          return junctionMap.Keys;
0  134        }
   135      }
   136
   137      /// <summary>
   138      /// Aktualisiert die Zuordnung, wenn sich die Kanten gendert haben.
   139      /// </summary>
   140      public void OnEdgesChanged ()
0  141      {
0  142        BuildIndex ();
0  143      }
   144
   145      private void BuildIndex ()
1  146      {
1  147        fromMap.Clear ();
1  148        toMap.Clear ();
3  149        float x = Offset.X, y = Offset.Y, z = Offset.Z;
15 150        foreach (Edge edge in Edges) {
```

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

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:	110

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.Linq;
```

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

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

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:	172

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

```
#region Copyright

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

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
```

```
      using System.Linq;

      using Microsoft.Xna.Framework;
      using Microsoft.Xna.Framework.Audio;
      using Microsoft.Xna.Framework.Content;
      using Microsoft.Xna.Framework.GamerServices;
      using Microsoft.Xna.Framework.Graphics;
      using Microsoft.Xna.Framework.Input;
      using Microsoft.Xna.Framework.Media;
      using Microsoft.Xna.Framework.Net;
      using Microsoft.Xna.Framework.Storage;

      using Knot3.Core;
      using Knot3.GameObjects;
      using Knot3.Input;
      using Knot3.RenderEffects;
      using Knot3.Screens;
      using Knot3.Utilities;
      using Knot3.Widgets;

      #endregion

      namespace Knot3.Data
      {
        public sealed class RectangleMap
        {
          #region Properties

          private INodeMap NodeMap;
0         private Dictionary<Vector3, List<PossibleRectanglePosition>> positions
              = new Dictionary<Vector3, List<PossibleRectanglePosition>> ();

          #endregion

          #region Constructors

0         public RectangleMap (INodeMap nodeMap)
0         {
0           NodeMap = nodeMap;
0         }

          #endregion

          #region Methods

          public void AddEdge (Edge edge, bool isVirtual)
0         {
0           Node a = NodeMap.NodeBeforeEdge (edge);
0           Node b = NodeMap.NodeAfterEdge (edge);
0           AddEdge (edge, a, b, isVirtual);
0         }

          public void AddEdge (Edge edge, Node nodeA, Node nodeB, bool isVirtual)
0         {
0           Vector3 edgeCenter = nodeA.CenterBetween (nodeB);
0           foreach (Direction direction in Direction.Values) {
0             if (direction.Axis != edge.Direction.Axis) {
0               Vector3 rectangleCenter = edgeCenter + direction * Node.Scale / 2;
0               PossibleRectanglePosition rectanglePosition = new PossibleRectangleP
                  Edge = edge,
                  NodeA = nodeA,
```

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

```
      public Edge Edge;
      public Node NodeA;
      public Node NodeB;
      public Vector3 Position;
      public bool IsVirtual;
    }

    public struct ValidRectanglePosition {
      public Edge EdgeAB;
      public Edge EdgeCD;
      public Node NodeA;
      public Node NodeB;
      public Node NodeC;
      public Node NodeD;
      public Vector3 Position;
      public bool IsVirtual;
    }
  }
```

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:	261

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.IO;`
	34	`using System.Linq;`
	35	`using System.Text;`
	36	
	37	`using Microsoft.Xna.Framework;`

```
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Ionic.Zip;

using Knot3.Core;
using Knot3.Development;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Data
{
  /// <summary>
  /// Implementiert das Speicherformat fr Challenges.
  /// </summary>
  public sealed class ChallengeFileIO : IChallengeIO
  {
    #region Properties

    /// <summary>
    /// Die fr eine Knoten-Datei gltigen Dateiendungen.
    /// </summary>
    public IEnumerable<string> FileExtensions
    {
      get {
        yield return ".challenge";
        yield return ".chl";
        yield return ".chn";
        yield return ".chg";
        yield return ".chlng";
      }
    }

    #endregion

    #region Constructors

    /// <summary>
    /// Erstellt ein ChallengeFileIO-Objekt.
    /// </summary>
    public ChallengeFileIO ()
    {
    }

    #endregion

    #region Methods

    /// <summary>
    /// Speichert eine Challenge in dem Dateinamen, der in dem Challenge-Objek
```

```
    /// </summary>
    public void Save (Challenge challenge)
    {
      using (ZipFile zip = new ZipFile ()) {
        // Namen
        zip.AddEntry ("name.txt", challenge.Name);
        // Startknoten
        KnotStringIO parser = new KnotStringIO (challenge.Start);
        zip.AddEntry ("start.knot", parser.Content);
        // Zielknoten
        parser = new KnotStringIO (challenge.Target);
        zip.AddEntry ("target.knot", parser.Content);
        // Highscore
        zip.AddEntry ("highscore.txt", string.Join ("\n", printHighscore (chal
        // ZIP-Datei speichern
        zip.Save (challenge.MetaData.Filename);
      }
    }

    /// <summary>
    /// Ldt eine Challenge aus einer angegebenen Datei.
    /// </summary>
    public Challenge Load (string filename)
    {
      ChallengeMetaData meta = LoadMetaData (filename: filename);
      Knot start = null;
      Knot target = null;

      using (ZipFile zip = ZipFile.Read (filename)) {
        foreach (ZipEntry entry in zip) {
          string content = entry.ReadContent ();

          // fr die Datei mit dem Startknoten
          if (entry.FileName.ToLower ().Contains ("start")) {
            KnotStringIO parser = new KnotStringIO (content: content);
            start = new Knot (
                new KnotMetaData (parser.Name, () => parser.CountEdges, null,
                parser.Edges
            );
          }

          // fr die Datei mit dem Zielknoten
          else if (entry.FileName.ToLower ().Contains ("target")) {
            KnotStringIO parser = new KnotStringIO (content: content);
            target = new Knot (
                new KnotMetaData (parser.Name, () => parser.CountEdges, null,
                parser.Edges
            );
          }
        }
      }

      if (meta != null && start != null && target != null) {
        return new Challenge (meta, start, target);
      }
      else {
        throw new IOException (
            "Error! Invalid challenge file: " + filename
            + " (meta=" + meta + ",start=" + start + ",target=" + target + ")"
        );
      }
```

```
    }

    /// <summary>
    /// Ldt die Metadaten einer Challenge aus einer angegebenen Datei.
    /// </summary>
    public ChallengeMetaData LoadMetaData (string filename)
    {
      string name = null;
      KnotMetaData start = null;
      KnotMetaData target = null;
      IEnumerable<KeyValuePair<string, int>> highscore = null;
      using (ZipFile zip = ZipFile.Read (filename)) {
        foreach (ZipEntry entry in zip) {
          string content = entry.ReadContent ();

          // fr die Datei mit dem Startknoten
          if (entry.FileName.ToLower ().Contains ("start")) {
            KnotStringIO parser = new KnotStringIO (content: content);
            start = new KnotMetaData (parser.Name, () => parser.CountEdges, nu
          }

          // fr die Datei mit dem Zielknoten
          else if (entry.FileName.ToLower ().Contains ("target")) {
            KnotStringIO parser = new KnotStringIO (content: content);
            target = new KnotMetaData (parser.Name, () => parser.CountEdges, n
          }

          // fr die Datei mit dem Namen
          else if (entry.FileName.ToLower ().Contains ("name")) {
            name = content.Trim ();
          }

          // fr die Datei mit den Highscores
          else if (entry.FileName.ToLower ().Contains ("highscore")) {
            highscore = parseHighscore (content.Split (new char[] {'\r','\n'},
          }
        }
      }
      if (name != null && start != null && target != null) {
        Log.Debug ("Load challenge file: ", filename, " (name=", name, ",start
        return new ChallengeMetaData (
                   name: name,
                   start: start,
                   target: target,
                   filename: filename,
                   format: this,
                   highscore: highscore
               );
      }
      else {
        throw new IOException (
            "Error! Invalid challenge file: " + filename
            + " (name=" + name + ",start=" + start + ",target=" + target + ",h
        );
      }
    }

    IEnumerable<string> printHighscore (IEnumerable<KeyValuePair<string, int>>
    {
      foreach (KeyValuePair<string, int> entry in highscore) {
        Log.Debug (
```

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

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:	50%
Covered lines:	32
Uncovered lines:	32
Coverable lines:	64
Total lines:	238

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
IsRunningOnMono()	1	0	0
IsRunningOnMonogame(	1	0	0
IsRunningOnLinux()	1	100	100
IsRunningOnWindows()	1	0	0
.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.IO;
33  using System.Linq;
34  using System.Security.Cryptography;
35  using System.Text;
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.Development;
49  using Knot3.GameObjects;
50  using Knot3.RenderEffects;
51  using Knot3.Screens;
52  using Knot3.Utilities;
53  using Knot3.Widgets;
54
55  #endregion
56
57  namespace Knot3.Platform
58  {
59    public static partial class SystemInfo
60    {
61      #region Properties
62
63      /// <summary>
64      /// Das Einstellungsverzeichnis.
65      /// </summary>
66      public static string SettingsDirectory
67      {
68        get {
69          string directory;
70          if (SystemInfo.IsRunningOnLinux ()) {
71            directory = Environment.GetEnvironmentVariable ("HOME") + "/.knot3/"
72          }
73          else {
74            directory = Environment.GetFolderPath (System.Environment.SpecialFol
75          }
76          Directory.CreateDirectory (directory);
77          return directory;
78        }
79      }
80
81      /// <summary>
82      /// Das Spielstandverzeichnis.
83      /// </summary>
84      public static string SavegameDirectory
85      {
86        get {
87          string directory = SettingsDirectory + PathSeparator.ToString () + "Sa
88          Directory.CreateDirectory (directory);
89          return directory;
90        }
91      }
92
```

33	68
33	70
0	71
0	72
33	73
33	74
33	75
33	76
33	77
33	78
32	86
32	87
32	88
32	89
32	90

```
      93      /// <summary>
      94      /// Das Bildschirmfotoverzeichnis.
      95      /// </summary>
      96      public static string ScreenshotDirectory
      97      {
0     98        get {
      99          string directory;
0    100          if (SystemInfo.IsRunningOnLinux ()) {
0    101            directory = Environment.GetEnvironmentVariable ("HOME");
0    102          }
0    103          else {
0    104            directory = Environment.GetFolderPath (System.Environment.SpecialFol
0    105          }
0    106          Directory.CreateDirectory (directory);
0    107          return directory;
0    108        }
     109      }
     110
     111      public static string DecodedMusicCache
     112      {
0    113        get {
     114          string directory;
0    115          if (SystemInfo.IsRunningOnLinux ()) {
0    116            directory = "/var/tmp/knot3/";
0    117          }
0    118          else {
0    119            directory = Environment.GetFolderPath (System.Environment.SpecialFol
0    120          }
0    121          Directory.CreateDirectory (directory);
0    122          return directory;
0    123        }
     124      }
     125
     126      public static string BaseDirectory
     127      {
1    128        get {
1    129          if (baseDirectory != null) {
0    130            return baseDirectory;
     131          }
1    132          else {
1    133            string cwd = Directory.GetCurrentDirectory ();
1    134            string[] binDirectories = new string[] {
     135              "Debug",
     136              "Release",
     137              "x86",
     138              "bin"
     139            };
15   140            foreach (string dir in binDirectories) {
6    141              if (cwd.ToLower ().EndsWith (dir.ToLower ())) {
2    142                cwd = cwd.Substring (0, cwd.Length - dir.Length - 1);
2    143              }
4    144            }
     145            // Environment.CurrentDirectory = cwd;
1    146            Log.Debug (cwd);
1    147            cwd = FileUtility.AbsoluteToRelative (cwd);
1    148            baseDirectory = cwd;
     149            return cwd;
1    150          }
     151        }
     152      }
1    153
```

```
154      private static string baseDirectory = null;
198 155  public readonly static char PathSeparator = Path.DirectorySeparatorChar;
156
157      #endregion
158    }
159  }
```

\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.Linq;
34
35 using Microsoft.Xna.Framework;
36 using Microsoft.Xna.Framework.Audio;
37 using Microsoft.Xna.Framework.Content;
38 using Microsoft.Xna.Framework.GamerServices;
39 using Microsoft.Xna.Framework.Graphics;
40 using Microsoft.Xna.Framework.Input;
41 using Microsoft.Xna.Framework.Media;
42 using Microsoft.Xna.Framework.Net;
43 using Microsoft.Xna.Framework.Storage;
44
45 using Knot3.Core;
46 using Knot3.Data;
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.Platform
      56  {
      57    public static partial class SystemInfo
      58    {
      59      public static bool IsRunningOnMono ()
   0  60      {
   0  61        return false;
   0  62      }
      63
      64      public static bool IsRunningOnMonogame ()
   0  65      {
   0  66        return false;
   0  67      }
      68
      69      public static bool IsRunningOnLinux ()
  61  70      {
  61  71        return false;
  61  72      }
      73
      74      public static bool IsRunningOnWindows ()
   0  75      {
   0  76        return true;
   0  77      }
      78    }
      79  }
```

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:	24
Coverable lines:	24
Total lines:	85

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
GetHashCode()	1	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.Linq;
      33  using System.Text;
      34
      35  using Microsoft.Xna.Framework;
      36
      37  #endregion
      38
      39  namespace Knot3.Utilities
      40  {
      41    public struct BoundingCylinder : IEquatable<BoundingCylinder> {
      42      public Vector3 SideA;
      43      public Vector3 SideB;
      44      public float Radius;
      45
      46      public BoundingCylinder (Vector3 sideA, Vector3 sideB, float radius)
  0   47      {
  0   48        this.SideA = sideA;
  0   49        this.SideB = sideB;
  0   50        this.Radius = radius;
  0   51      }
      52
      53      public static bool operator == (BoundingCylinder a, BoundingCylinder b)
  0   54      {
  0   55        if (System.Object.ReferenceEquals (a, b)) {
  0   56          return true;
      57        }
  0   58        if (((object)a == null) || ((object)b == null)) {
  0   59          return false;
      60        }
  0   61        return a.Equals (b);
  0   62      }
      63
      64      public static bool operator != (BoundingCylinder a, BoundingCylinder b)
  0   65      {
  0   66        return !(a == b);
  0   67      }
      68
      69      public bool Equals (BoundingCylinder other)
  0   70      {
  0   71        return SideA == other.SideA && SideB == other.SideB && Radius == other.R
  0   72      }
      73
      74      public override bool Equals (object other)
  0   75      {
  0   76        return other != null && Equals ((BoundingCylinder)other);
  0   77      }
      78
      79      public override int GetHashCode ()
  0   80      {
      81        // irgendwas mglichst eindeutiges
  0   82        return (Radius * (SideA + SideB)).GetHashCode ();
  0   83      }
      84    }
      85  }
```

Knot3.Utilities.ColorHelper

Summary

Class:	Knot3.Utilities.ColorHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\ColorHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	10
Coverable lines:	10
Total lines:	75

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
Mix(...)	1	0	0
Luminance(...)	1	0	0
SortColorsByLuminanc	1	0	0

File(s)

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

```
#region Copyright

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

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

```

```
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Data;
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.Utilities
56  {
57    public static class ColorHelper
58    {
59      public static Color Mix (this Color a, Color b, float percent = 0.5f)
0   60      {
0   61        percent = MathHelper.Clamp (percent, 0f, 1f);
0   62        return new Color (a.ToVector3 () * (1f - percent) + b.ToVector3 () * per
0   63      }
64
65      public static int Luminance (this Color color)
0   66      {
0   67        return (color.R * 3 + color.B + color.G * 4) >> 3;
0   68      }
69
70      public static int SortColorsByLuminance (Color left, Color right)
0   71      {
0   72        return left.Luminance ().CompareTo (right.Luminance ());
0   73      }
74    }
75  }
```

Knot3.Utilities.DictionaryHelper

Summary

Class:	Knot3.Utilities.DictionaryHelper
Assembly:	Knot3
File(s):	ers\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\DictionaryHelper.cs
Coverage:	100%
Covered lines:	6
Uncovered lines:	0
Coverable lines:	6
Total lines:	69

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
Add(...)	4	100	60

File(s)

ers\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\DictionaryHelper.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.Linq;
34
35 using Microsoft.Xna.Framework;
36 using Microsoft.Xna.Framework.Audio;
37 using Microsoft.Xna.Framework.Content;
```

```
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Data;
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.Utilities
56  {
57    public static class DictionaryHelper
58    {
59      public static void Add<KeyType, ListType, ValueType> (this Dictionary<KeyT
60              KeyType key, ValueType value)
61      where ListType : IList<ValueType>, new ()
62      {
63        if (!dict.ContainsKey (key)) {
64          dict.Add (key, new ListType ());
65        }
66        dict [key].Add (value);
67      }
68    }
69  }
```

4	62
8	63
4	64
4	65
4	66
4	67

Knot3.Utilities.EnumHelper

Summary

Class:	Knot3.Utilities.EnumHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\EnumHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	49
Coverable lines:	49
Total lines:	140

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
GetDescriptionTable(	3	0	0
ToEnumValues()	1	0	0
ToEnumDescription(..	4	0	0
ToEnumValue(...)	5	0	0
MoveNext()	5	0	0
MoveNext()	6	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\EnumHelper.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.Linq;
35  using System.Reflection;
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.Core;
49  using Knot3.Data;
50  using Knot3.GameObjects;
51  using Knot3.Input;
52  using Knot3.RenderEffects;
53  using Knot3.Screens;
54  using Knot3.Widgets;
55
56  #endregion
57
58  namespace Knot3.Utilities
59  {
60    public static class EnumHelper
61    {
62      public static IEnumerable<string> ToEnumDescriptions<T> (this IEnumerable<
0 63      {
0 64        foreach (T val in enumValues) {
0 65          yield return val.ToEnumDescription<T> ();
0 66        }
0 67      }
68
69      public static Hashtable GetDescriptionTable<T> ()
0 70      {
0 71        Hashtable table = new Hashtable ();
0 72        foreach (T val in ToEnumValues<T>()) {
0 73          string descr = val.ToEnumDescription<T> ();
0 74          table [val] = descr;
0 75          table [descr] = val;
0 76        }
0 77        return table;
0 78      }
79
80      public static IEnumerable<T> ToEnumValues<T> ()
0 81      {
0 82        Type enumType = typeof (T);
83
0 84        return enumType.ToEnumValues<T> ();
0 85      }
86
87      public static IEnumerable<T> ToEnumValues<T> (this Type enumType)
0 88      {
0 89        if (enumType.BaseType != typeof (Enum)) {
0 90          throw new ArgumentException ("T must be of type System.Enum");
91        }
92
```

```
0    93        Array enumValArray = Enum.GetValues (enumType);
     94
0    95        foreach (int val in enumValArray) {
0    96          yield return (T)Enum.Parse (enumType, val.ToString ());
0    97        }
0    98      }
     99
    100      public static string ToEnumDescription<T> (this T value)
0   101      {
0   102        Type enumType = typeof (T);
    103
0   104        if (enumType.BaseType != typeof (Enum)) {
0   105          throw new ArgumentException ("T must be of type System.Enum");
    106        }
    107
0   108        FieldInfo fi = value.GetType ().GetField (value.ToString ());
    109
0   110        DescriptionAttribute[] attributes =
    111            (DescriptionAttribute[])fi.GetCustomAttributes (
    112                typeof (DescriptionAttribute),
    113                false);
    114
0   115        if (attributes != null && attributes.Length > 0) {
0   116          return attributes [0].Description;
    117        }
0   118        else {
0   119          return value.ToString ();
    120        }
0   121      }
    122
    123      public static T ToEnumValue<T> (this string value)
0   124      {
0   125        Type enumType = typeof (T);
0   126        if (enumType.BaseType != typeof (Enum)) {
0   127          throw new ArgumentException ("T must be of type System.Enum");
    128        }
    129
0   130        T returnValue = default (T);
0   131        foreach (T enumVal in ToEnumValues<T>()) {
0   132          if (enumVal.ToEnumDescription<T> () == value) {
0   133            returnValue = enumVal;
0   134            break;
    135          }
0   136        }
0   137        return returnValue;
0   138      }
    139    }
    140  }
```

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:	99

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;
```

```
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.Data;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Utilities
{
  public class FileIndex
  {
    private HashSet<string> hashes;
    private string filename;

    public FileIndex (string filename)
    {
      this.filename = filename;
      try {
        hashes = new HashSet<string> (FileUtility.ReadFrom (filename));
      }
      catch (IOException) {
        hashes = new HashSet<string> ();
      }
    }

    public void Add (string hash)
    {
      hashes.Add (hash);
      Save ();
    }

    public void Remove (string hash)
    {
      hashes.Remove (hash);
      Save ();
    }

    public bool Contains (string hash)
    {
      return hashes.Contains (hash);
    }

```

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

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:	13.8%
Covered lines:	5
Uncovered lines:	31
Coverable lines:	36
Total lines:	139

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	0	0
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.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.Widgets;
      54
      55  #endregion
      56
      57  namespace Knot3.Utilities
      58  {
      59    /// <summary>
      60    /// Eine Hilfsklasse fr Dateioperationen.
      61    /// </summary>
      62    public static class FileUtility
      63    {
      64      #region Methods
      65
      66      /// <summary>
      67      /// Konvertiert einen Namen eines Knotens oder einer Challenge in einen g
      68      /// </summary>
      69      public static string ConvertToFileName (string name)
 32   70      {
 32   71        char[] arr = name.ToCharArray ();
190   72        arr = Array.FindAll<char> (arr, (c => (char.IsLetterOrDigit (c)
      73                                              || char.IsWhiteSpace (c)
      74                                              || c == '-'))
      75                                 );
 32   76        return new string (arr);
 32   77      }
      78
      79      /// <summary>
      80      /// Liefert einen Hash-Wert zu der durch filename spezifizierten Datei.
      81      /// </summary>
      82      public static string GetHash (string filename)
  0   83      {
  0   84        return string.Join ("\n", FileUtility.ReadFrom (filename)).ToMD5Hash ();
  0   85      }
      86
      87      public static string ToMD5Hash (this string TextToHash)
  0   88      {
  0   89        if (string.IsNullOrEmpty (TextToHash)) {
  0   90          return string.Empty;
      91        }
      92
```

```
0    93        MD5 md5 = new MD5CryptoServiceProvider ();
0    94        byte[] textToHash = Encoding.Default.GetBytes (TextToHash);
0    95        byte[] result = md5.ComputeHash (textToHash);
     96
0    97        return System.BitConverter.ToString (result);
0    98      }
     99
    100      public static IEnumerable<string> ReadFrom (string file)
0   101      {
    102        string line;
0   103        using (var reader = File.OpenText (file)) {
0   104          while ((line = reader.ReadLine ()) != null) {
0   105            yield return line;
0   106          }
0   107        }
0   108      }
    109
    110      public static string AbsoluteToRelative (string directory)
0   111      {
0   112        Console.WriteLine ("Absolute: " + directory);
0   113        Uri currentUri = new Uri (Directory.GetCurrentDirectory ()+"/");
0   114        Console.WriteLine ("Current: " + currentUri.ToString ());
0   115        Uri relativeUri = currentUri.MakeRelativeUri (new Uri (directory+"/"));
    116        Console.WriteLine ("Relative: " + relativeUri);
    117        return relativeUri.ToString ();
0   118      }
0   119
0   120      public static void SearchFiles (IEnumerable<string> directories, IEnumerab
0   121      {
0   122        foreach (string directory in directories) {
0   123          SearchFiles (directory, extensions, add);
0   124        }
0   125      }
    126
    127      public static void SearchFiles (string directory, IEnumerable<string> exte
    128      {
    129        Directory.CreateDirectory (directory);
    130        var files = Directory.GetFiles (directory, "*.*", SearchOption.AllDirect
    131                    .Where (s => extensions.Any (e => s.EndsWith (e)));
    132        foreach (string file in files) {
    133          add (file);
    134        }
    135      }
    136
    137      #endregion
    138    }
    139  }
```

Knot3.Utilities.FrustumHelper

Summary

Class:	Knot3.Utilities.FrustumHelper
Assembly:	Knot3
File(s):	\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FrustumHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	85
Coverable lines:	85
Total lines:	172

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
GetNegativeVertex(..	4	0	0
GetPositiveVertex(..	4	0	0
FastIntersects(...)	2	0	0
FastIntersects(...)	8	0	0

File(s)

\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\FrustumHelper.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.Linq;
```

```
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Data;
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.Utilities
56  {
57    public static class FrustumHelper
58    {
59      public static Vector3 GetNegativeVertex (this BoundingBox aabb, ref Vector
0   60      {
0   61        Vector3 p = aabb.Max;
0   62        if (normal.X >= 0) {
0   63          p.X = aabb.Min.X;
0   64        }
0   65        if (normal.Y >= 0) {
0   66          p.Y = aabb.Min.Y;
0   67        }
0   68        if (normal.Z >= 0) {
0   69          p.Z = aabb.Min.Z;
0   70        }
71
0   72        return p;
0   73      }
74
75      public static Vector3 GetPositiveVertex (this BoundingBox aabb, ref Vector
0   76      {
0   77        Vector3 p = aabb.Min;
0   78        if (normal.X >= 0) {
0   79          p.X = aabb.Max.X;
0   80        }
0   81        if (normal.Y >= 0) {
0   82          p.Y = aabb.Max.Y;
0   83        }
0   84        if (normal.Z >= 0) {
0   85          p.Z = aabb.Max.Z;
0   86        }
87
0   88        return p;
0   89      }
90
91      public static bool FastIntersects (this BoundingFrustum boundingfrustum, r
0   92      {
0   93        if (boundingfrustum == null) {
0   94          return false;
```

```
      }
0     var box = BoundingBox.CreateFromSphere (aabb);
0     return boundingfrustum.FastIntersects (ref box);
0   }

  public static bool FastIntersects (this BoundingFrustum boundingfrustum, r
0 {
0   if (boundingfrustum == null) {
0     return false;
    }

    Plane plane;
    Vector3 normal, p;

0   plane = boundingfrustum.Bottom;
0   normal = plane.Normal;
0   normal.X = -normal.X;
0   normal.Y = -normal.Y;
0   normal.Z = -normal.Z;
0   p = aabb.GetPositiveVertex (ref normal);
0   if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
0     return false;
    }

0   plane = boundingfrustum.Far;
0   normal = plane.Normal;
0   normal.X = -normal.X;
0   normal.Y = -normal.Y;
0   normal.Z = -normal.Z;
0   p = aabb.GetPositiveVertex (ref normal);
0   if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
0     return false;
    }

0   plane = boundingfrustum.Left;
0   normal = plane.Normal;
0   normal.X = -normal.X;
0   normal.Y = -normal.Y;
0   normal.Z = -normal.Z;
0   p = aabb.GetPositiveVertex (ref normal);
0   if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
0     return false;
    }

0   plane = boundingfrustum.Near;
0   normal = plane.Normal;
0   normal.X = -normal.X;
0   normal.Y = -normal.Y;
0   normal.Z = -normal.Z;
0   p = aabb.GetPositiveVertex (ref normal);
0   if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
0     return false;
    }

0   plane = boundingfrustum.Right;
0   normal = plane.Normal;
0   normal.X = -normal.X;
0   normal.Y = -normal.Y;
0   normal.Z = -normal.Z;
0   p = aabb.GetPositiveVertex (ref normal);
0   if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
```

```
0   156          return false;
    157        }
    158
0   159        plane = boundingfrustum.Top;
0   160        normal = plane.Normal;
0   161        normal.X = -normal.X;
0   162        normal.Y = -normal.Y;
0   163        normal.Z = -normal.Z;
0   164        p = aabb.GetPositiveVertex (ref normal);
0   165        if (-plane.D + normal.X * p.X + normal.Y * p.Y + normal.Z * p.Z < 0) {
0   166          return false;
    167        }
    168
0   169        return true;
0   170      }
    171    }
    172  }
```

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:	52.1%
Covered lines:	36
Uncovered lines:	33
Coverable lines:	69
Total lines:	129

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
.ctor(...)	9	100	70.59
Dispose()	1	0	0
Dispose(...)	2	0	0
Save()	8	0	0
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.Globalization;
```

```
        33  using System.IO;
        34  using System.Linq;
        35
        36  #endregion
        37
        38  namespace Knot3.Utilities
        39  {
        40    public sealed class IniFile : IDisposable
        41    {
        42      private string Filename;
        43      public Dictionary<string, Dictionary<string, string>> Data;
        44
   2    45      public IniFile (string filename)
   2    46      {
   2    47        Data = new Dictionary<string, Dictionary<string, string>> ();
   2    48        Filename = filename;
   4    49        if (File.Exists (filename)) {
   4    50          using (StreamReader reader = new StreamReader (filename)) {
   2    51            string section = null;
2704    52            while (reader.Peek () != -1) {
1351    53              string line = StripComments (reader.ReadLine ().Trim ());
1702    54              if (line.StartsWith ("[") && line.EndsWith ("]")) {
 351    55                section = line.Substring (1, line.Length - 2);
 702    56                if (!Data.ContainsKey (section)) {
 351    57                  Data [section] = new Dictionary<string,string> ();
 351    58                }
 351    59              }
2000    60              else if (line.Contains ("=")) {
1000    61                string[] parts = line.Split ('=');
2000    62                if (section != null) {
1000    63                  Data [section] [parts [0].Trim ()] = parts [1].Trim ();
1000    64                }
1000    65              }
1351    66            }
   2    67          }
   2    68        }
   2    69      }
        70
        71      public void Dispose ()
   0    72      {
   0    73        Dispose (true);
   0    74        GC.SuppressFinalize (this);
   0    75      }
        76
        77      private void Dispose (bool disposing)
   0    78      {
   0    79        if (disposing) {
   0    80          Save ();
   0    81        }
   0    82      }
        83
        84      public void Save ()
   0    85      {
   0    86        using (StreamWriter writer = new StreamWriter (Filename)) {
   0    87          foreach (string section in Data.Keys.OrderBy (x => x)) {
   0    88            writer.WriteLine ("[" + section + "]");
   0    89            foreach (string key  in Data[section].Keys.OrderBy (x => x)) {
   0    90              writer.WriteLine (key + "=" + Data [section] [key]);
   0    91            }
   0    92          }
   0    93        }
```

```
    0      }

      private static string StripComments (string line)
 1351 {
 2702   if (line != null) {
 1351     if (line.IndexOf (';') != -1) {
    0       return line.Remove (line.IndexOf (';')).Trim ();
        }
 1351   return line.Trim ();
      }
    0 return string.Empty;
 1351 }

      public string this [string section, string key, string defaultValue = null
      {
    5   get {
    5     if (!Data.ContainsKey (section)) {
    0       Data [section] = new Dictionary<string,string> ();
    0     }
    5     if (!Data [section].ContainsKey (key)) {
    0       Data [section] [key] = defaultValue;
    0       Save ();
    0     }
    5     string value = Data [section] [key];
    5     return value;
    5   }
    0   set {
    0     if (!Data.ContainsKey (section)) {
    0       Data [section] = new Dictionary<string,string> ();
    0     }
    0     Data [section] [key] = value;
    0     Save ();
    0   }
      }
    }
}
```

Knot3.Utilities.InputHelper

Summary

Class:	Knot3.Utilities.InputHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\InputHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	10
Coverable lines:	10
Total lines:	82

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
IsDown(...)	3	0	0
IsHeldDown(...)	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\InputHelper.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.Linq;
    34
    35  using Microsoft.Xna.Framework;
```

```
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

using Knot3.Core;
using Knot3.Data;
using Knot3.GameObjects;
using Knot3.Input;
using Knot3.RenderEffects;
using Knot3.Screens;
using Knot3.Widgets;

#endregion

namespace Knot3.Utilities
{
  public static class InputHelper
  {
    /// <summary>
    /// Wurde die aktuelle Taste gedrckt und war sie im letzten Frame nicht g
    /// </summary>
    public static bool IsDown (this Keys key)
0   {
      // Is the key down?
0     if (InputManager.CurrentKeyboardState.IsKeyDown (key)) {
        // If not down last update, key has just been pressed.
0       if (!InputManager.PreviousKeyboardState.IsKeyDown (key)) {
0         return true;
        }
0     }
0     return false;
0   }

    /// <summary>
    /// Wird die aktuelle Taste gedrckt gehalten?
    /// </summary>
    public static bool IsHeldDown (this Keys key)
0   {
0     return InputManager.CurrentKeyboardState.IsKeyDown (key);
0   }
  }
}
```

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:	146

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.Linq;`
	33	`using System.Text;`
	34	
	35	`using Microsoft.Xna.Framework;`
	36	
	37	`#endregion`

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

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

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:	129

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

```
#region Copyright

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

#endregion

#region Using

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

```

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

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

Knot3.Utilities.TextHelper

Summary

Class:	Knot3.Utilities.TextHelper
Assembly:	Knot3
File(s):	c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\TextHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	222
Coverable lines:	222
Total lines:	343

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
TryTextInput(...)	13	0	0
TryConvertKey(...)	59	0	0
.cctor()	1	0	0

File(s)

c:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\TextHelper.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.Linq;`
	34	

```
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Data;
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.Utilities
56  {
57    public static class TextHelper
58    {
0   59      private static Keys lastKey = Keys.None;
0   60      private static double lastMillis = 0;
61
62      public static bool TryTextInput (ref string str, GameTime time)
0   63      {
0   64        bool catched = false;
0   65        if (lastKey != Keys.None) {
0   66          if (InputManager.CurrentKeyboardState.IsKeyUp (lastKey)) {
0   67            lastKey = Keys.None;
0   68          }
0   69          else if ((time.TotalGameTime.TotalMilliseconds - lastMillis) > 100) {
0   70            lastKey = Keys.None;
0   71          }
0   72        }
0   73        Keys[] keys = InputManager.CurrentKeyboardState.GetPressedKeys ();
0   74        if (lastKey == Keys.None) {
0   75          for (int i = 0; i < keys.Length; ++i) {
0   76            if (keys [i] != Keys.LeftShift && keys [i] != Keys.RightShift) {
0   77              lastKey = keys [i];
0   78            }
0   79          }
0   80          if (lastKey != Keys.None) {
0   81            if (lastKey == Keys.Back) {
0   82              if (str.Length != 0) {
0   83                str = str.Substring (0, str.Length - 1);
0   84              }
0   85              catched = true;
0   86            }
0   87            else if (str.Length < 100) {
88              char c;
0   89              if (TryConvertKey (lastKey, out c)) {
0   90                str += c.ToString ();
0   91              }
0   92              catched = true;
0   93            }
0   94          }
95
```

```
0    96          lastMillis = time.TotalGameTime.TotalMilliseconds;
0    97        }
0    98        return catched;
0    99      }
     100
     101      private static bool TryConvertKey (Keys keyPressed, out char key)
0    102      {
0    103        bool shift = Keys.LeftShift.IsHeldDown () || Keys.RightShift.IsHeldDown
     104
0    105        char c = (char)keyPressed.GetHashCode ();
0    106        if (c >= 'A' && c <= 'Z') {
0    107          if (shift) {
0    108            key = char.ToUpper (c);
0    109          }
0    110          else {
0    111            key = char.ToLower (c);
0    112          }
0    113          return true;
     114        }
     115
0    116        switch (keyPressed) {
     117          //Decimal keys
     118        case Keys.D0:
0    119          if (shift) {
0    120            key = ')';
0    121          }
0    122          else {
0    123            key = '0';
0    124          }
0    125          return true;
     126        case Keys.D1:
0    127          if (shift) {
0    128            key = '!';
0    129          }
0    130          else {
0    131            key = '1';
0    132          }
0    133          return true;
     134        case Keys.D2:
0    135          if (shift) {
0    136            key = '@';
0    137          }
0    138          else {
0    139            key = '2';
0    140          }
0    141          return true;
     142        case Keys.D3:
0    143          if (shift) {
0    144            key = '#';
0    145          }
0    146          else {
0    147            key = '3';
0    148          }
0    149          return true;
     150        case Keys.D4:
0    151          if (shift) {
0    152            key = '$';
0    153          }
0    154          else {
0    155            key = '4';
0    156          }
```

```
0   157          return true;
    158        case Keys.D5:
0   159          if (shift) {
0   160            key = '%';
0   161          }
0   162          else {
0   163            key = '5';
0   164          }
0   165          return true;
    166        case Keys.D6:
0   167          if (shift) {
0   168            key = '^';
0   169          }
0   170          else {
0   171            key = '6';
0   172          }
0   173          return true;
    174        case Keys.D7:
0   175          if (shift) {
0   176            key = '&';
0   177          }
0   178          else {
0   179            key = '7';
0   180          }
0   181          return true;
    182        case Keys.D8:
0   183          if (shift) {
0   184            key = '*';
0   185          }
0   186          else {
0   187            key = '8';
0   188          }
0   189          return true;
    190        case Keys.D9:
0   191          if (shift) {
0   192            key = '(';
0   193          }
0   194          else {
0   195            key = '9';
0   196          }
0   197          return true;
    198
    199          //Decimal numpad keys
    200        case Keys.NumPad0:
0   201          key = '0';
0   202          return true;
    203        case Keys.NumPad1:
0   204          key = '1';
0   205          return true;
    206        case Keys.NumPad2:
0   207          key = '2';
0   208          return true;
    209        case Keys.NumPad3:
0   210          key = '3';
0   211          return true;
    212        case Keys.NumPad4:
0   213          key = '4';
0   214          return true;
    215        case Keys.NumPad5:
0   216          key = '5';
0   217          return true;
```

```
      218        case Keys.NumPad6:
0     219          key = '6';
0     220          return true;
      221        case Keys.NumPad7:
0     222          key = '7';
0     223          return true;
      224        case Keys.NumPad8:
0     225          key = '8';
0     226          return true;
      227        case Keys.NumPad9:
0     228          key = '9';
0     229          return true;
      230
      231          //Special keys
      232        case Keys.OemTilde:
0     233          if (shift) {
0     234            key = ' ';
0     235          }
0     236          else {
0     237            key = '`';
0     238          }
0     239          return true;
      240        case Keys.OemSemicolon:
0     241          if (shift) {
0     242            key = ':';
0     243          }
0     244          else {
0     245            key = ';';
0     246          }
0     247          return true;
      248        case Keys.OemQuotes:
0     249          if (shift) {
0     250            key = '"';
0     251          }
0     252          else {
0     253            key = '\'';
0     254          }
0     255          return true;
      256        case Keys.OemQuestion:
0     257          if (shift) {
0     258            key = '?';
0     259          }
0     260          else {
0     261            key = '/';
0     262          }
0     263          return true;
      264        case Keys.OemPlus:
0     265          if (shift) {
0     266            key = '+';
0     267          }
0     268          else {
0     269            key = '=';
0     270          }
0     271          return true;
      272        case Keys.OemPipe:
0     273          if (shift) {
0     274            key = '|';
0     275          }
0     276          else {
0     277            key = '\\';
0     278          }
```

```
0   279          return true;
    280        case Keys.OemPeriod:
0   281          if (shift) {
0   282            key = '>';
0   283          }
0   284          else {
0   285            key = '.';
0   286          }
0   287          return true;
    288        case Keys.OemOpenBrackets:
0   289          if (shift) {
0   290            key = '{';
0   291          }
0   292          else {
0   293            key = '[';
0   294          }
0   295          return true;
    296        case Keys.OemCloseBrackets:
0   297          if (shift) {
0   298            key = '}';
0   299          }
0   300          else {
0   301            key = ']';
0   302          }
0   303          return true;
    304        case Keys.OemMinus:
0   305          if (shift) {
0   306            key = '_';
0   307          }
0   308          else {
0   309            key = '-';
0   310          }
0   311          return true;
    312        case Keys.OemComma:
0   313          if (shift) {
0   314            key = '<';
0   315          }
0   316          else {
0   317            key = ',';
0   318          }
0   319          return true;
    320        case Keys.Space:
0   321          key = ' ';
0   322          return true;
    323        }
    324
0   325        key = (char)0;
0   326        return false;
0   327      }
    328
0   329      public static List<Keys> ValidKeys = new List<Keys> (
    330      new Keys[] {
    331        Keys.A, Keys.B, Keys.C, Keys.D, Keys.E, Keys.F, Keys.G, Keys.H, Keys.I,
    332        Keys.L, Keys.M, Keys.N, Keys.O, Keys.P, Keys.Q, Keys.R, Keys.S, Keys.T,
    333        Keys.W, Keys.X, Keys.Y, Keys.Z,
    334        Keys.D0, Keys.D1, Keys.D2, Keys.D3, Keys.D4, Keys.D5, Keys.D6, Keys.D7,
    335        Keys.NumPad0, Keys.NumPad1, Keys.NumPad2, Keys.NumPad3, Keys.NumPad4, Ke
    336        Keys.NumPad6, Keys.NumPad7, Keys.NumPad8, Keys.NumPad9,
    337        Keys.OemTilde, Keys.OemSemicolon, Keys.OemQuotes, Keys.OemQuestion, Keys
    338        Keys.OemPipe, Keys.OemPeriod, Keys.OemOpenBrackets, Keys.OemCloseBracket
    339        Keys.OemComma, Keys.Space, Keys.Back
```

```
      }
      );
    }
  }
```

Knot3.Utilities.TextureHelper

Summary

Class:	Knot3.Utilities.TextureHelper
Assembly:	Knot3
File(s):	\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\TextureHelper.cs
Coverage:	0%
Covered lines:	0
Uncovered lines:	91
Coverable lines:	91
Total lines:	215

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
LoadTexture(...)	1	0	0
LoadFont(...)	1	0	0
Create(...)	1	0	0
Create(...)	3	0	0
CreateGradient(...)	2	0	0
DrawColoredRectangle	1	0	0
DrawStringInRectangl	1	0	0
ScaleStringInRectang	2	0	0
TextPosition(...)	4	0	0
.cctor()	1	0	0

File(s)

\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\TextureHelper.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.Linq;
34
35  using Microsoft.Xna.Framework;
36  using Microsoft.Xna.Framework.Audio;
37  using Microsoft.Xna.Framework.Content;
38  using Microsoft.Xna.Framework.GamerServices;
39  using Microsoft.Xna.Framework.Graphics;
40  using Microsoft.Xna.Framework.Input;
41  using Microsoft.Xna.Framework.Media;
42  using Microsoft.Xna.Framework.Net;
43  using Microsoft.Xna.Framework.Storage;
44
45  using Knot3.Core;
46  using Knot3.Data;
47  using Knot3.Development;
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.Utilities
57  {
58    public static class TextureHelper
59    {
60      #region Real Textures
61
62      public static Texture2D LoadTexture (this IGameScreen screen, string name)
0   63      {
0   64        try {
0   65          return screen.Content.Load<Texture2D> ("Textures/" + name);
66        }
0   67        catch (ContentLoadException ex) {
0   68          Log.Debug (ex);
0   69          return null;
70        }
0   71      }
72
73      public static SpriteFont LoadFont (this IGameScreen screen, string name)
0   74      {
0   75        try {
0   76          return screen.Content.Load<SpriteFont> ("Fonts/" + name);
77        }
0   78        catch (ContentLoadException ex) {
0   79          Log.Debug (ex);
0   80          return null;
81        }
0   82      }
83
84      #endregion
85
86      #region Dummy Textures
87
88      public static Texture2D Create (GraphicsDevice graphicsDevice, Color color
```

```
0    89      {
0    90        return Create (graphicsDevice, 1, 1, color);
0    91      }
     92
0    93      private static Dictionary<string, Texture2D> textureCache = new Dictionary
     94
     95      public static Texture2D Create (GraphicsDevice graphicsDevice, int width,
0    96      {
0    97        string key = color.ToString () + width.ToString () + "x" + height.ToStri
0    98        if (textureCache.ContainsKey (key)) {
0    99          return textureCache [key];
    100        }
0   101        else {
    102          // create a texture with the specified size
0   103          Texture2D texture = new Texture2D (graphicsDevice, width, height);
    104
    105          // fill it with the specified colors
0   106          Color[] colors = new Color[width * height];
0   107          for (int i = 0; i < colors.Length; i++) {
0   108            colors [i] = new Color (color.ToVector3 ());
0   109          }
0   110          texture.SetData (colors);
0   111          textureCache [key] = texture;
0   112          return texture;
    113        }
0   114      }
    115
    116      public static Texture2D CreateGradient (GraphicsDevice graphicsDevice, Col
0   117      {
0   118        string key = color1.ToString () + color2.ToString () + "gradient";
0   119        if (textureCache.ContainsKey (key)) {
0   120          return textureCache [key];
    121        }
0   122        else {
    123          // create a texture with the specified size
0   124          Texture2D texture = new Texture2D (graphicsDevice, 2, 2);
    125
    126          // fill it with the specified colors
0   127          Color[] colors = new Color[texture.Width * texture.Height];
0   128          colors [0] = color1;
0   129          colors [1] = color2;
0   130          colors [2] = color2;
0   131          colors [3] = color1;
0   132          texture.SetData (colors);
0   133          textureCache [key] = texture;
0   134          return texture;
    135        }
0   136      }
    137
    138      public static void DrawColoredRectangle (this SpriteBatch spriteBatch, Col
0   139      {
0   140        Texture2D texture = TextureHelper.Create (spriteBatch.GraphicsDevice, Co
0   141        spriteBatch.Draw (
    142            texture, bounds, null, color, 0f, Vector2.Zero, SpriteEffects.None,
    143        );
0   144      }
    145
    146      public static void DrawStringInRectangle (this SpriteBatch spriteBatch, Sp
    147              string text, Color color, Rectangle bounds,
    148              HorizontalAlignment alignX, VerticalAlignment alignY)
0   149      {
```

```
0   150         Vector2 scaledPosition = new Vector2 (bounds.X, bounds.Y);
0   151         Vector2 scaledSize = new Vector2 (bounds.Width, bounds.Height);
0   152         try {
    153           // finde die richtige Skalierung
0   154           Vector2 scale = spriteBatch.ScaleStringInRectangle (font, text, color,
    155
    156           // finde die richtige Position
0   157           Vector2 textPosition = TextPosition (
    158                                 font: font, text: text, scale: scale,
    159                                 position: scaledPosition, size: scaledSize,
    160                                 alignX: alignX, alignY: alignY
    161                               );
    162
    163           // zeichne die Schrift
0   164           spriteBatch.DrawString (font, text, textPosition, color, 0, Vector2.Ze
0   165         }
0   166         catch (ArgumentException exp) {
0   167           Log.Debug (exp);
0   168         }
0   169         catch (InvalidOperationException exp) {
0   170           Log.Debug (exp);
0   171         }
0   172       }
    173
    174       public static Vector2 ScaleStringInRectangle (this SpriteBatch spriteBatch
    175               string text, Color color, Rectangle bounds,
    176               HorizontalAlignment alignX, VerticalAlignment alignY)
0   177       {
0   178         Vector2 scaledSize = new Vector2 (bounds.Width, bounds.Height);
0   179         try {
    180           // finde die richtige Skalierung
0   181           Vector2 scale = scaledSize / font.MeasureString (text) * 0.9f;
0   182           if (!text.Contains ("\n")) {
0   183             scale.Y = scale.X = MathHelper.Min (scale.X, scale.Y);
0   184           }
0   185           return scale;
    186         }
0   187         catch (Exception exp) {
0   188           Log.Debug (exp);
0   189           return Vector2.One;
    190         }
0   191       }
    192
    193       public static Vector2 TextPosition (SpriteFont font, string text, Vector2
    194                                           HorizontalAlignment alignX, VerticalAl
0   195       {
0   196         Vector2 textPosition = position;
0   197         Vector2 minimumSize = font.MeasureString (text);
0   198         switch (alignX) {
    199         case HorizontalAlignment.Left:
0   200           textPosition.Y += (size.Y - minimumSize.Y * scale.Y) / 2;
0   201           break;
    202         case HorizontalAlignment.Center:
0   203           textPosition += (size - minimumSize * scale) / 2;
0   204           break;
    205         case HorizontalAlignment.Right:
0   206           textPosition.Y += (size.Y - minimumSize.Y * scale.Y) / 2;
0   207           textPosition.X += size.X - minimumSize.X * scale.X;
0   208           break;
    209         }
0   210         return textPosition;
```

```
0    211        }
     212
     213        #endregion
     214      }
     215    }
```

Knot3.Utilities.VectorHelper

Summary

Class:	Knot3.Utilities.VectorHelper
Assembly:	Knot3
File(s):	:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\VectorHelper.cs
Coverage:	19.6%
Covered lines:	65
Uncovered lines:	265
Coverable lines:	330
Total lines:	593

Metrics

Method	Cyclomatic Complexity	Sequence Coverage	Branch Coverage
ArcBallMove(...)	3	0	0
MoveLinear(...)	1	0	0
MoveLinear(...)	1	0	0
AngleBetween(...)	3	0	0
AngleBetween(...)	1	0	0
RotateX(...)	1	0	0
RotateY(...)	1	0	0
RotateZ(...)	1	0	0
RotateAroundVector(.	1	0	0
Clamp(...)	1	0	0
Clamp(...)	3	0	0
PrimaryVector(...)	3	0	0
PrimaryVector(...)	7	0	0
PrimaryDirection(...	1	0	0
PrimaryDirection(...	1	0	0
PrimaryDirectionExce	4	0	0
Abs(...)	1	100	100
Clamp(...)	1	0	0
Bounds(...)	2	0	0
Bounds(...)	1	0	0
Scale(...)	1	0	0
Scale(...)	1	0	0
Translate(...)	1	0	0
Rotate(...)	1	0	0
Scale(...)	1	0	0
Translate(...)	1	0	0
ToVector2(...)	1	0	0
ToScreenPoint(...)	1	0	0
ToPoint(...)	1	0	0
ToVector2(...)	1	100	100
Center(...)	1	0	0
Length(...)	1	0	0
ToVector2(...)	1	0	0
ToPoint(...)	1	100	100
Plus(...)	1	0	0
Join(...)	2	0	0
ScaleFactor(...)	1	0	0
RelativeTo(...)	1	0	0
Scale(...)	3	75	60
Scale(...)	1	0	0
Grow(...)	1	0	0
Shrink(...)	1	0	0

Grow(...)	1	0	0
Shrink(...)	1	0	0
Translate(...)	1	0	0
Resize(...)	1	0	0
Swap(...)	1	0	0
Print(...)	1	0	0
CylinderBounds(...)	2	100	100
CreateRectangle(...)	1	0	0
CreateRectangle(...)	1	0	0
CreateRectangle(...)	3	0	0
At(...)	4	0	0
At(...)	4	81.25	85.71
At(...)	4	0	0
At(...)	5	0	0
RandomIndex(...)	1	0	0
RandomElement(...)	1	0	0
SetCoordinates(...)	1	0	0
ReverseDictionary(..	1	0	0
DistanceTo(...)	1	100	100
SetDistanceTo(...)	4	100	85.71
Shuffle(...)	1	0	0
Repeat(...)	2	0	0
Repeat(...)	2	100	100
.cctor()	1	100	100
MoveNext()	5	0	0
MoveNext()	6	0	0
MoveNext()	5	100	83.33
MoveNext()	5	100	83.33

File(s)

:\Users\Pascal\Documents\GitHub\knot3-code\src\Knot3\Utilities\VectorHelper.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.Linq;
34  using System.Text;
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.RenderEffects;
52  using Knot3.Screens;
53  using Knot3.Widgets;
54
55  #endregion
56
57  namespace Knot3.Utilities
58  {
59    public static class VectorHelper
60    {
1   61      private static readonly float MinAngleY = 0.1f;
1   62      private static readonly float MaxAngleY = MathHelper.Pi - 0.1f;
63
64      public static Vector3 ArcBallMove (this Vector3 position, Vector2 mouse, V
0   65      {
0   66        Vector3 side = Vector3.Normalize (Vector3.Cross (up, forward));
67        //Vector3 relUp = Vector3.Normalize (Vector3.Cross (side, forward));
68
69        // horizontal rotation
0   70        float diffAngleX = MathHelper.Pi / 300f * mouse.X;
0   71        Vector3 rotated = position.RotateAroundVector (up, diffAngleX);
72
73        // vertical rotation
0   74        float currentAngleY = position.AngleBetween (up);
0   75        float diffAngleY = MathHelper.Pi / 200f * mouse.Y;
0   76        if (currentAngleY + diffAngleY > MinAngleY && currentAngleY + diffAngleY
0   77          rotated = rotated.RotateAroundVector (-side, diffAngleY);
0   78        }
79
0   80        return rotated;
0   81      }
82
83      public static Vector3 MoveLinear (this Vector3 vectorToMove, Vector3 mouse
0   84      {
85        /*Vector3 side = Vector3.Cross (up, forward);
86        side.Normalize ();
```

```
      87        Vector3 relUp = Vector3.Cross (side, forward);
      88        relUp.Normalize ();
      89        Vector3 movedVector = vectorToMove - side * mouse.X - relUp * mouse.Y -
      90        return movedVector;*/
0     91        Vector3 movedVector = Vector3.Zero;
0     92        Profiler.ProfileDelegate ["gesamt"] = () => {
0     93          Vector3 side = Vector3.Zero;
0     94          Profiler.ProfileDelegate ["Cross1"] = () => {
0     95            side = Vector3.Cross (up, forward);
0     96            side.Normalize ();
      97          };
0     98          Vector3 relUp = Vector3.Zero;
0     99          Profiler.ProfileDelegate ["Cross2"] = () => {
0    100            relUp = Vector3.Cross (side, forward);
0    101            relUp.Normalize ();
     102          };
     103
0    104          Profiler.ProfileDelegate ["PlusMinux"] = () => {
0    105            movedVector = vectorToMove - side * mouse.X - relUp * mouse.Y - forw
     106          };
0    107        };
0    108        return movedVector;
0    109      }
     110
     111      public static Vector3 MoveLinear (this Vector3 vectorToMove, Vector2 mouse
0    112      {
0    113        return vectorToMove.MoveLinear (new Vector3 (mouse.X, mouse.Y, 0), up, f
0    114      }
     115
     116      public static float AngleBetween (this Vector2 a, Vector2 b)
0    117      {
0    118        return ((b.X - a.X) > 0 ? 1 : -1) * (float)Math.Acos ((double)Vector2.Do
0    119      }
     120
     121      public static float AngleBetween (this Vector3 a, Vector3 b)
0    122      {
0    123        return //((b.X - a.X) > 0 ? 1 : -1) *
     124            (float)Math.Acos ((double)Vector3.Dot (Vector3.Normalize (a), Vector
0    125      }
     126
     127      public static Vector3 RotateX (this Vector3 vectorToRotate, float angleRad
0    128      {
0    129        return Vector3.Transform (vectorToRotate, Matrix.CreateRotationX (angleR
0    130      }
     131
     132      public static Vector3 RotateY (this Vector3 vectorToRotate, float angleRad
0    133      {
0    134        return Vector3.Transform (vectorToRotate, Matrix.CreateRotationY (angleR
0    135      }
     136
     137      public static Vector3 RotateZ (this Vector3 vectorToRotate, float angleRad
0    138      {
0    139        return Vector3.Transform (vectorToRotate, Matrix.CreateRotationZ (angleR
0    140      }
     141
     142      public static Vector3 RotateAroundVector (this Vector3 vectorToRotate, Vec
0    143      {
0    144        return Vector3.Transform (vectorToRotate, Quaternion.CreateFromAxisAngle
0    145      }
     146
     147      public static Vector3 Clamp (this Vector3 v, Vector3 lower, Vector3 higher
```

```
0   148     {
0   149       return new Vector3 (
            150                 MathHelper.Clamp (v.X, lower.X, higher.X),
            151                 MathHelper.Clamp (v.Y, lower.Y, higher.Y),
            152                 MathHelper.Clamp (v.Z, lower.Z, higher.Z)
            153             );
0   154     }
            155
            156     public static Vector3 Clamp (this Vector3 v, int minLength, int maxLength)
0   157     {
0   158       if (v.Length () < minLength) {
0   159         return v * minLength / v.Length ();
            160       }
0   161       else if (v.Length () > maxLength) {
0   162         return v * maxLength / v.Length ();
            163       }
0   164       else {
0   165         return v;
            166       }
0   167     }
            168
            169     public static Vector2 PrimaryVector (this Vector2 v)
0   170     {
0   171       if (v.X.Abs () > v.Y.Abs ()) {
0   172         return new Vector2 (v.X, 0);
            173       }
0   174       else if (v.Y.Abs () > v.X.Abs ()) {
0   175         return new Vector2 (0, v.Y);
            176       }
0   177       else {
0   178         return new Vector2 (v.X, 0);
            179       }
0   180     }
            181
            182     public static Vector3 PrimaryVector (this Vector3 v)
0   183     {
0   184       if (v.X.Abs () > v.Y.Abs () && v.X.Abs () > v.Z.Abs ()) {
0   185         return new Vector3 (v.X, 0, 0);
            186       }
0   187       else if (v.Y.Abs () > v.X.Abs () && v.Y.Abs () > v.Z.Abs ()) {
0   188         return new Vector3 (0, v.Y, 0);
            189       }
0   190       else if (v.Z.Abs () > v.Y.Abs () && v.Z.Abs () > v.X.Abs ()) {
0   191         return new Vector3 (0, 0, v.Z);
            192       }
0   193       else {
0   194         return new Vector3 (v.X, 0, 0);
            195       }
0   196     }
            197
            198     public static Vector2 PrimaryDirection (this Vector2 v)
0   199     {
0   200       Vector2 vector = v.PrimaryVector ();
0   201       return new Vector2 (Math.Sign (vector.X), Math.Sign (vector.Y));
0   202     }
            203
            204     public static Vector3 PrimaryDirection (this Vector3 v)
0   205     {
0   206       Vector3 vector = v.PrimaryVector ();
0   207       return new Vector3 (Math.Sign (vector.X), Math.Sign (vector.Y), Math.Sig
0   208     }
```

```
      209
      210     public static Vector3 PrimaryDirectionExcept (this Vector3 v, Vector3 wron
0     211     {
0     212       Vector3 copy = v;
0     213       if (wrongDirection.X != 0) {
0     214         copy.X = 0;
0     215       }
0     216       else if (wrongDirection.Y != 0) {
0     217         copy.Y = 0;
0     218       }
0     219       else if (wrongDirection.Z != 0) {
0     220         copy.Z = 0;
0     221       }
0     222       return copy.PrimaryDirection ();
0     223     }
      224
      225     public static float Abs (this float v)
35    226     {
35    227       return Math.Abs (v);
35    228     }
      229
      230     public static float Clamp (this float v, float min, float max)
0     231     {
0     232       return MathHelper.Clamp (v, min, max);
0     233     }
      234
      235     public static BoundingSphere[] Bounds (this Model model)
0     236     {
      237       //Log.Debug (model);
0     238       BoundingSphere[] bounds = new BoundingSphere[model.Meshes.Count];
0     239       int i = 0;
0     240       foreach (ModelMesh mesh in model.Meshes) {
0     241         bounds [i++] = mesh.BoundingSphere;
0     242       }
0     243       return bounds;
0     244     }
      245
      246     public static BoundingBox Bounds (this Vector3 a, Vector3 diff)
0     247     {
0     248       return new BoundingBox (a, a + diff);
0     249     }
      250
      251     public static BoundingSphere Scale (this BoundingSphere sphere, float scal
0     252     {
0     253       return new BoundingSphere (sphere.Center, sphere.Radius * scale);
0     254     }
      255
      256     public static BoundingSphere Scale (this BoundingSphere sphere, Vector3 sc
0     257     {
0     258       return new BoundingSphere (sphere.Center, sphere.Radius * scale.PrimaryV
0     259     }
      260
      261     public static BoundingSphere Translate (this BoundingSphere sphere, Vector
0     262     {
0     263       return new BoundingSphere (Vector3.Transform (sphere.Center, Matrix.Crea
0     264     }
      265
      266     public static BoundingSphere Rotate (this BoundingSphere sphere, Angles3 r
0     267     {
0     268       return new BoundingSphere (Vector3.Transform (sphere.Center, Matrix.Crea
0     269     }
```

```
270
271     public static BoundingBox Scale (this BoundingBox box, float scale)
0 272   {
0 273     return new BoundingBox (box.Min * scale, box.Max * scale);
0 274   }
275
276     public static BoundingBox Translate (this BoundingBox box, Vector3 positio
0 277   {
0 278     Matrix translation = Matrix.CreateTranslation (position);
0 279     return new BoundingBox (Vector3.Transform (box.Min, translation), Vector
0 280   }
281
282     public static Vector2 ToVector2 (this MouseState mouse)
0 283   {
0 284     return new Vector2 (mouse.X, mouse.Y);
0 285   }
286
287     public static ScreenPoint ToScreenPoint (this MouseState mouse, IGameScree
0 288   {
0 289     Vector2 vector = mouse.ToVector2 () / screen.Viewport.ToVector2 ();
0 290     return new ScreenPoint (screen, vector);
0 291   }
292
293     public static Point ToPoint (this MouseState mouse)
0 294   {
0 295     return new Point (mouse.X, mouse.Y);
0 296   }
297
298     public static Vector2 ToVector2 (this Viewport viewport)
6 299   {
6 300     return new Vector2 (viewport.Width, viewport.Height);
6 301   }
302
303     public static Vector2 Center (this Viewport viewport)
0 304   {
0 305     return new Vector2 (viewport.Width, viewport.Height) / 2;
0 306   }
307
308     public static float Length (this Point p)
0 309   {
0 310     return p.ToVector2 ().Length ();
0 311   }
312
313     public static Vector2 ToVector2 (this Point p)
0 314   {
0 315     return new Vector2 (p.X, p.Y);
0 316   }
317
318     public static Point ToPoint (this Vector2 v)
6 319   {
6 320     return new Point ((int)v.X, (int)v.Y);
6 321   }
322
323     public static Point Plus (this Point a, Point b)
0 324   {
0 325     return new Point (a.X + b.X, a.Y + b.Y);
0 326   }
327
328     public static string Join (this string delimiter, List<int> list)
0 329   {
0 330     StringBuilder builder = new StringBuilder ();
```

```
0   331        foreach (int elem in list) {
    332          // Append each int to the StringBuilder overload.
0   333          builder.Append (elem).Append (delimiter);
0   334        }
0   335        return builder.ToString ();
0   336      }
    337
    338      public static Vector2 ScaleFactor (this Viewport viewport)
0   339      {
0   340        Vector2 max = viewport.ToVector2 ();
0   341        return max / 1000f;
0   342      }
    343
    344      public static Vector2 RelativeTo (this Vector2 v, Viewport viewport)
0   345      {
0   346        Vector2 max = viewport.ToVector2 ();
0   347        return v / max;
0   348      }
    349
    350      public static Vector2 Scale (this Vector2 v, Viewport viewport)
6   351      {
6   352        Vector2 max = viewport.ToVector2 ();
6   353        if (v.X > 1 || v.Y > 1) {
0   354          return v / 1000f * max;
    355        }
6   356        else {
6   357          return v * max;
    358        }
6   359      }
    360
    361      public static Rectangle Scale (this Rectangle rect, Viewport viewport)
0   362      {
0   363        Point max = viewport.ToVector2 ().ToPoint ();
    364
0   365        return new Rectangle (rect.X * max.X / 1000, rect.Y * max.Y / 1000, rect
0   366      }
    367
    368      public static Rectangle Grow (this Rectangle rect, int x, int y)
0   369      {
0   370        return new Rectangle (rect.X - x, rect.Y - y, rect.Width + x * 2, rect.H
0   371      }
    372
    373      public static Rectangle Shrink (this Rectangle rect, int x, int y)
0   374      {
0   375        return Grow (rect, -x, -y);
0   376      }
    377
    378      public static Rectangle Grow (this Rectangle rect, int xy)
0   379      {
0   380        return Grow (rect, xy, xy);
0   381      }
    382
    383      public static Rectangle Shrink (this Rectangle rect, int xy)
0   384      {
0   385        return Grow (rect, -xy, -xy);
0   386      }
    387
    388      public static Rectangle Translate (this Rectangle rect, int x, int y)
0   389      {
0   390        return new Rectangle (rect.X + x, rect.Y + y, rect.Width, rect.Height);
0   391      }
```

```
392
393     public static Rectangle Resize (this Rectangle rect, int w, int h)
394     {
395       return new Rectangle (rect.X, rect.Y, rect.Width + w, rect.Height + h);
396     }
397
398     public static void Swap<T> (ref T lhs, ref T rhs)
399     {
400       T temp;
401       temp = lhs;
402       lhs = rhs;
403       rhs = temp;
404     }
405
406     public static string Print (this Vector3 v)
407     {
408       return   "("
409                + v.X.ToString ()
410                + ","
411                + v.Y.ToString ()
412                + ","
413                + v.Z.ToString ()
414                + ")";
415     }
416
417     public static BoundingSphere[] CylinderBounds (float length, float radius,
418     {
419       float distance = radius / 4;
420       BoundingSphere[] bounds = new BoundingSphere[(int)(length / distance)];
421       for (int offset = 0; offset < (int)(length / distance); ++offset) {
422         bounds [offset] = new BoundingSphere (position + direction * offset *
423         //Log.Debug ("sphere[", offset, "]=", Bounds [offset]);
424       }
425       return bounds;
426     }
427
428     public static Rectangle CreateRectangle (this Vector2 topLeft, Vector2 siz
429     {
430       return CreateRectangle (0, topLeft.X, topLeft.Y, size.X, size.Y);
431     }
432
433     public static Rectangle CreateRectangle (this Vector2 topLeft, Vector2 siz
434     {
435       return CreateRectangle (lineWidth, topLeft.X, topLeft.Y, size.X, size.Y)
436     }
437
438     public static Rectangle CreateRectangle (int lineWidth, float x, float y,
439     {
440       if (w == 0) {
441         return new Rectangle ((int)x - lineWidth / 2, (int)y - lineWidth / 2,
442       }
443       else if (h == 0) {
444         return new Rectangle ((int)x - lineWidth / 2, (int)y - lineWidth / 2,
445       }
446       else {
447         return new Rectangle ((int)x, (int)y, (int)w, (int)h);
448       }
449     }
450
451     public static T At<T> (this List<T> list, int index)
452     {
```

```
  0    453        if (list.Count == 0) {
  0    454          return default (T);
       455        }
  0    456        else {
  0    457          while (index < 0) {
  0    458            index += list.Count;
  0    459          }
  0    460          if (index >= list.Count) {
  0    461            index = index % list.Count;
  0    462          }
  0    463          return list [index];
       464        }
  0    465      }
       466
       467      public static T At<T> (this IEnumerable<T> list, int index)
606    468      {
606    469        int count = list.Count ();
804    470        if (count == 0) {
198    471          return default (T);
       472        }
408    473        else {
408    474          while (index < 0) {
  0    475            index += count;
  0    476          }
409    477          if (index >= count) {
  1    478            index = index % count;
  1    479          }
408    480          return list.ElementAt (index);
       481        }
606    482      }
       483
       484      public static T At<T> (this Tuple<T,T> tuple, int i)
  0    485      {
  0    486        return i == 0 ? tuple.Item1 : i == 1 ? tuple.Item2 : default (T);
  0    487      }
       488
       489      public static T At<T> (this Tuple<T,T,T> tuple, int i)
  0    490      {
  0    491        return i == 0 ? tuple.Item1 : i == 1 ? tuple.Item2 : i == 2 ? tuple.Item
  0    492      }
       493
       494      public static IEnumerable<T> ToEnumerable<T> (this Tuple<T,T> tuple)
  0    495      {
  0    496        yield return tuple.Item1;
  0    497        yield return tuple.Item2;
  0    498      }
       499
       500      public static IEnumerable<T> ToEnumerable<T> (this Tuple<T,T,T> tuple)
  0    501      {
  0    502        yield return tuple.Item1;
  0    503        yield return tuple.Item2;
  0    504        yield return tuple.Item3;
  0    505      }
       506
  1    507      private static Random random = new Random (Guid.NewGuid ().GetHashCode ())
       508
       509      public static int RandomIndex<T> (this IEnumerable<T> list)
  0    510      {
  0    511        int index = random.Next (list.Count ());
  0    512        return index;
  0    513      }
```

```
514
515     public static T RandomElement<T> (this IEnumerable<T> list)
516     {
517       return list.At (list.RandomIndex ());
518     }
519
520     public static void SetCoordinates (this Widget widget, float left, float t
521     {
522       widget.Bounds.Position = new ScreenPoint (widget.Screen, left, top);
523       widget.Bounds.Size = new ScreenPoint (widget.Screen, right - left, botto
524     }
525
526     public static Dictionary<A, B> ReverseDictionary<A,B> (this Dictionary<B,A
527     {
528       return dict.ToDictionary (x => x.Value, x => x.Key);
529     }
530
531     public static float DistanceTo (this Vector3 origin, Vector3 target)
532     {
533       Vector3 toPosition = origin - target;
534       return toPosition.Length ();
535     }
536
537     public static Vector3 SetDistanceTo (this Vector3 origin, Vector3 target,
538     {
539       Vector3 to = origin - target;
540       float oldDistance = to.Length ();
541       double scale = (double)distance / (double)to.Length ();
542       if (Math.Abs (oldDistance) > 1 && Math.Abs (oldDistance - distance) > 1
543         return target + to * (float)scale;
544       }
545       else {
546         return origin;
547       }
548     }
549
550     public static IEnumerable<T> Shuffle<T> (this IEnumerable<T> source)
551     {
552       Random rnd = new Random ();
553       return source.OrderBy<T, int> ((item) => rnd.Next ());
554     }
555
556     public static void Repeat (this int count, Action action)
557     {
558       for (int i = 0; i < count; i++) {
559         action ();
560       }
561     }
562
563     public static void Repeat (this int count, Action<int> action)
564     {
565       for (int i = 0; i < count; i++) {
566         action (i);
567       }
568     }
569
570     public static IEnumerable<T> Repeat<T> (this int count, Func<int, T> func)
571     {
572       List<T> list = new List<T> ();
573       for (int i = 0; i < count; i++) {
574         list.Add (func (i));
```

```
  9        }
           return list;
         }

  2
604      public static IEnumerable<int> Range (this int count)
200      {
200        for (int i = 0; i < count; i++) {
  2          yield return i;
           }
         }

         public static void ForEach<U> (this IEnumerable<U> enumerable, Action<U> a
         {
           foreach (U item in enumerable) {
             action (item);
           }
         }
       }
     }
```