Technical Design Document

Underlord

Gliederung:

I. Technische/technologische Vorarbeiten

1. *Blender*
2. *Gimp*
3. *Paint.Net*
4. *Free Studio*
5. *Audacity*
6. *Maya*
7. *Photoshop*
8. *Sculptris*

II. Verwendete Bibliotheken

1. *Microsoft XNA Framework Redistributable 4.0*
2. *NVIDIA Texture Tools for Adobe Photoshop*

III. Zusätzliche Tools

IV. Formate

1. *Modelformat*
2. *Bildformat*
3. *Tonformat*
4. *Speicherformat*

V. Softwarearchitektur

1. *Klassendiagramm*
2. *Sequenzdiagramme*
3. *Use-Case-Diagramme*

VI. Technische Mindestanforderungen

VII. Projektmanagement

Technische/technologische Vorarbeiten:

Verwendete Programme:

Blender:
Erzeugung und Bearbeitung der Modelle
Erzeugung von Cutscenes
Erzeugung von Intro

Gimp:
Erstellung von Texturen

Paint.Net:
Erstellung von Diagrammen und Dummygrafiken.

Free Studio:
Sammlung von Programmen zur Bearbeitung und Konvertierung von Audio- und Video-Content.

Audacity:
Das primäre Programm zur Realisierung der Ton-Aufnahme, welche die Grundlage für den Game-Score bilden. Basierend auf den Aufnahmen finden weitere Verarbeitungsschritte statt welche als Ergebnis schlussendlich die Musikkulisse sowie die entsprechenden Sound-Clips liefern.

Maya:
Neben Blender das zentrale Objekt- und Charaktermodellierungstool. Von der eigentlichen Modellierung einmal abgesehen, wird das gesamte Character-Rigging, d.h. der Aufbau eines Bone-Skeletts und dessen Verknüpfung mit einem Model, ebenfalls in Maya umgesetzt. Darauf Aufbauen werden dann spieltechnisch, sinnvolle Animation definiert und mittels der Skelettstruktur realisiert. Dieser Animationsclip wird abschließend zusammen mit dem zugehörigen Model in ein einziges .FBX-File „gebacken“ und in XNA importiert.

Photoshop:
Zur Erzeugung, Nachbearbeitung und Verbesserungen von Model-Texturen oder anderen Sprite-Elementen. Aber auch für die externe Generierung von Normal-Maps ist Photoshop, unter Verwendung des zuvor genannten Plugins, geeignet.

Sculptris:
Sculptris kommt nur dann zum Einsatz, wenn das in Blender oder Maya erzeugte Model keine ausreichend gute Qualität aufweist, d.h. wenn das Model für den angedachten Spiel-Zweck nicht schön genug aussieht. Da sich auf diese Weise fast immer automatisch die Komplexität des Models erhöht, sollte Sculptris nur sehr sparsam eingesetzt werden. Nur wenn es absolut nötig ist und bereits alle anderen Möglichkeiten ausgeschöpft sind, eine vergleichbare Qualität im Vorfeld mittels anderen Modellierungstools zu erzielen, sollte dieser Versuch unternommen werden.

Verwendete Bibliotheken:

- *Microsoft XNA Framework Redistributable 4.0:*
  Framework für C# zur Entwiklung von Spielen in Visual Studio 2010.

- *NVIDIA Texture Tools for Adobe Photoshop:*
  Plugin für Photoshop, welches unter anderem für die Erzeugung von Normal-Maps geeignet ist. Gerade für Texturen mit einer festen Struktur, wie zum Beispiel Steinwände, Böden und Gestein, können durch Kombination mit einer Normal-Map den angedachten Realitätsgrad deutlich steigern ohne dabei eine sonderlich großen grafischen Mehraufwand zu betreiben.

Zusätzliche Tools:

- Visual Studio 2010
- GitHub
- TortoiseGit
- Skype
- Firefox
- VLC Media Player
- OpenOffice

Formate:

Modelformat:

- *.FBX:*
  Neben .X das einzige, offizielle Datenformat zur Importirrung von externen Modellen in XNA. Beim Import von Modellen mit Skelettstruktur und Animationsinformation ist besonders drauf zu achten, dass die durch XNA vorgeschriebene, maximale Anzahl von 59 Bones bzw. Joints nicht überschritten wird.

- *.OBJ:*
  Internes, Austauschformat zwischen den Modellierungstools Blender, Maya und Sculptris, um einen problemlosen Datenaustausch zu garantieren.

Bildformat:

- *.PNG:*
  Zentrales Bildformat in welchem die Modeltexturen und Sprits hinterlegt und in XNA importierte werden.

Tonformat:

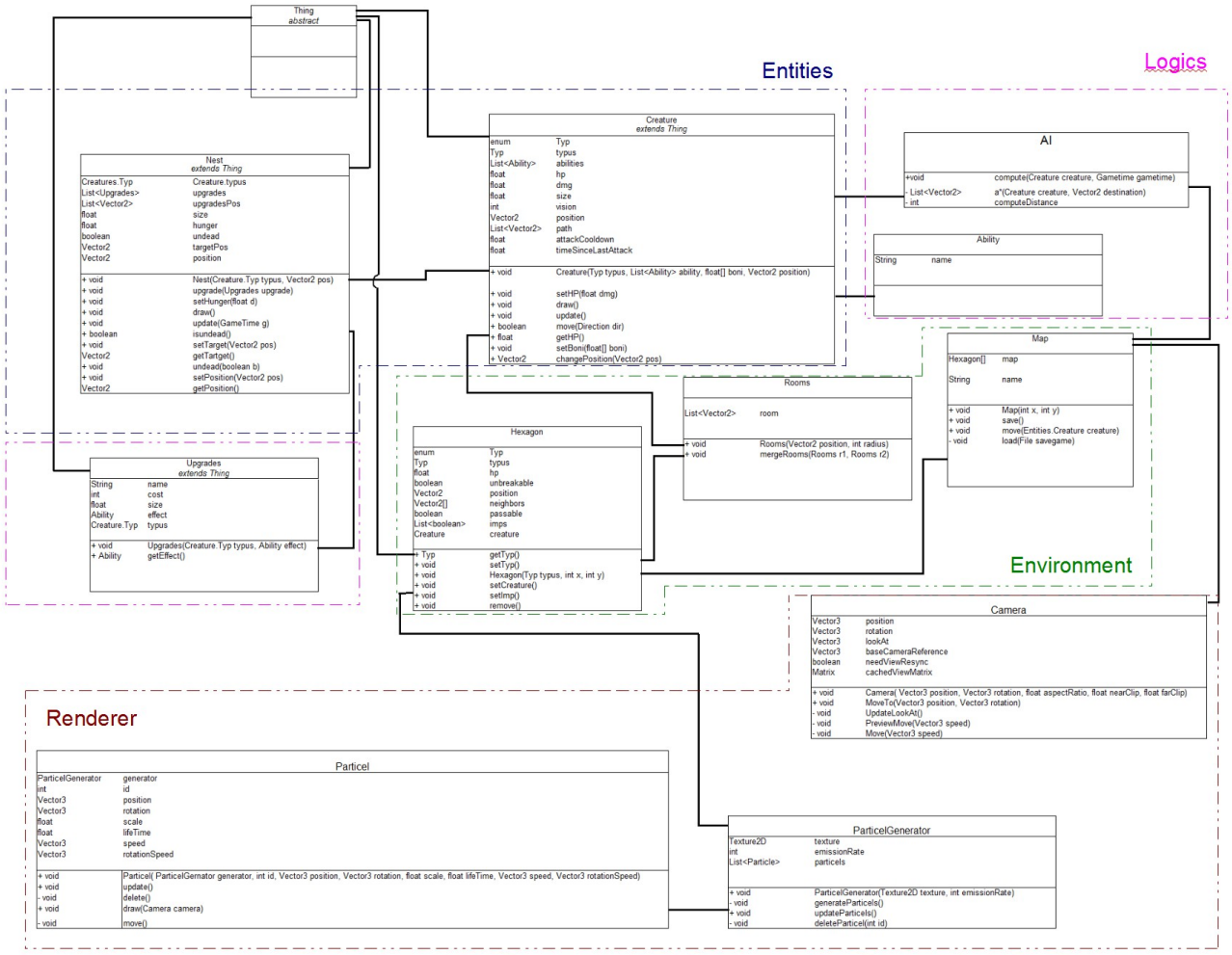
- *.WAV:*
  Zentrales Audioformat für Hintergrundmusik und Effektsounds in XNA.

Speicherformat:

- *.TXT:*
  Einfaches Format für das Speichern und das Laden von Leveln.

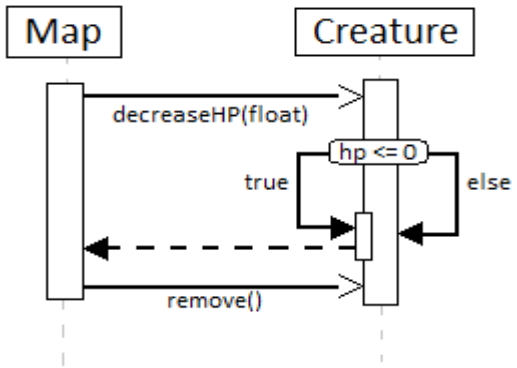
Softwarearchitektur:

Klassendiagramm:

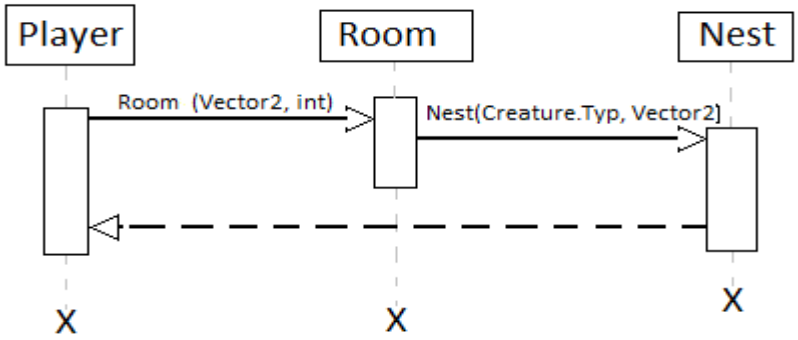


Sequenzdiagramme:

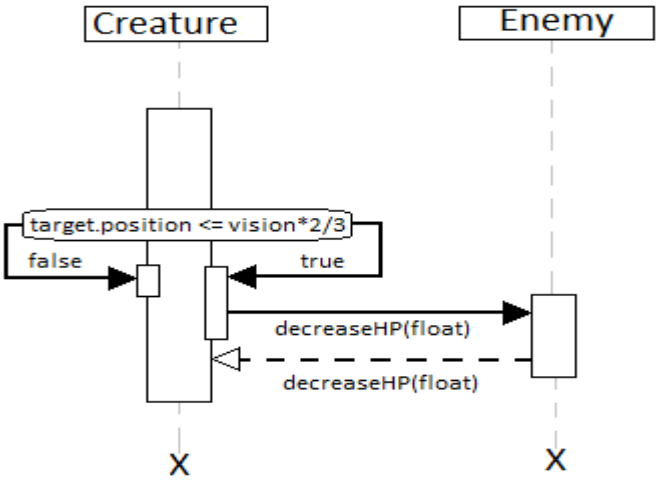
1. *Lebenszyklus einer Kreatur*
   Kreaturen (außer Imps) verlieren kontinuierlich Lebenspunkte (HP).



2. *Konstruktion eines Nestes*
   Für ein Nest muss erst ein freier Raum geschaffen werden.

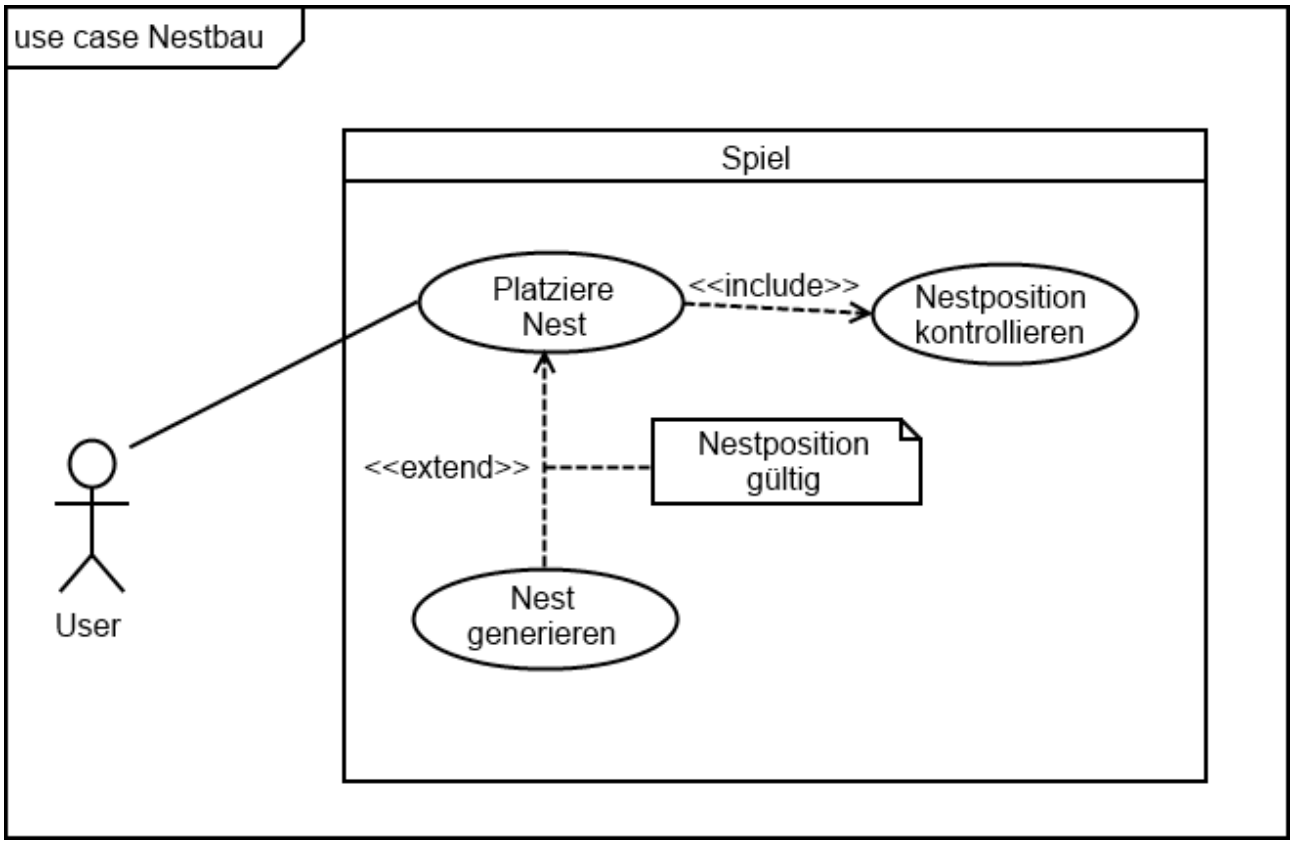


3. *Beispielhafter Kampf*
   Eine Kreatur wird erst angreifen, wenn sie eine gegnerische Kreatur sieht und sich diese in ihrem Angriffsbereich befindet (hier mit 2/3 der Sichtweite angegeben).

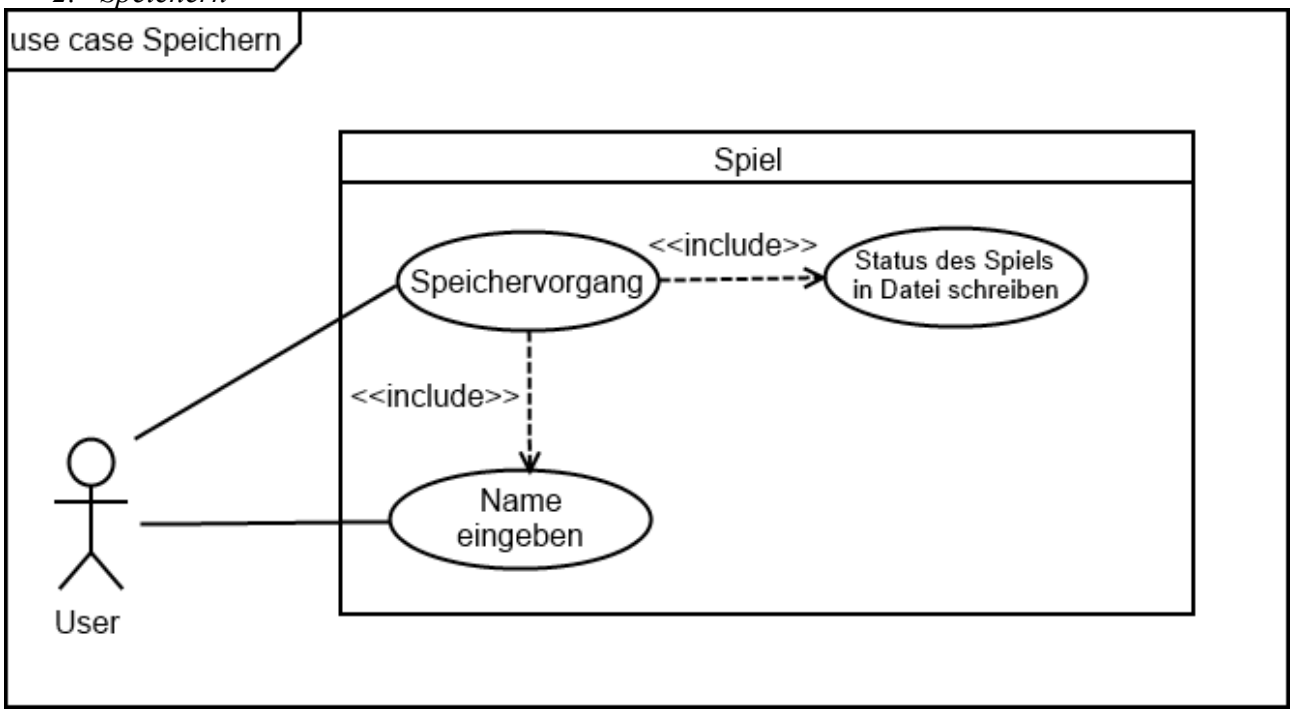


Use-Case-Diagramme:

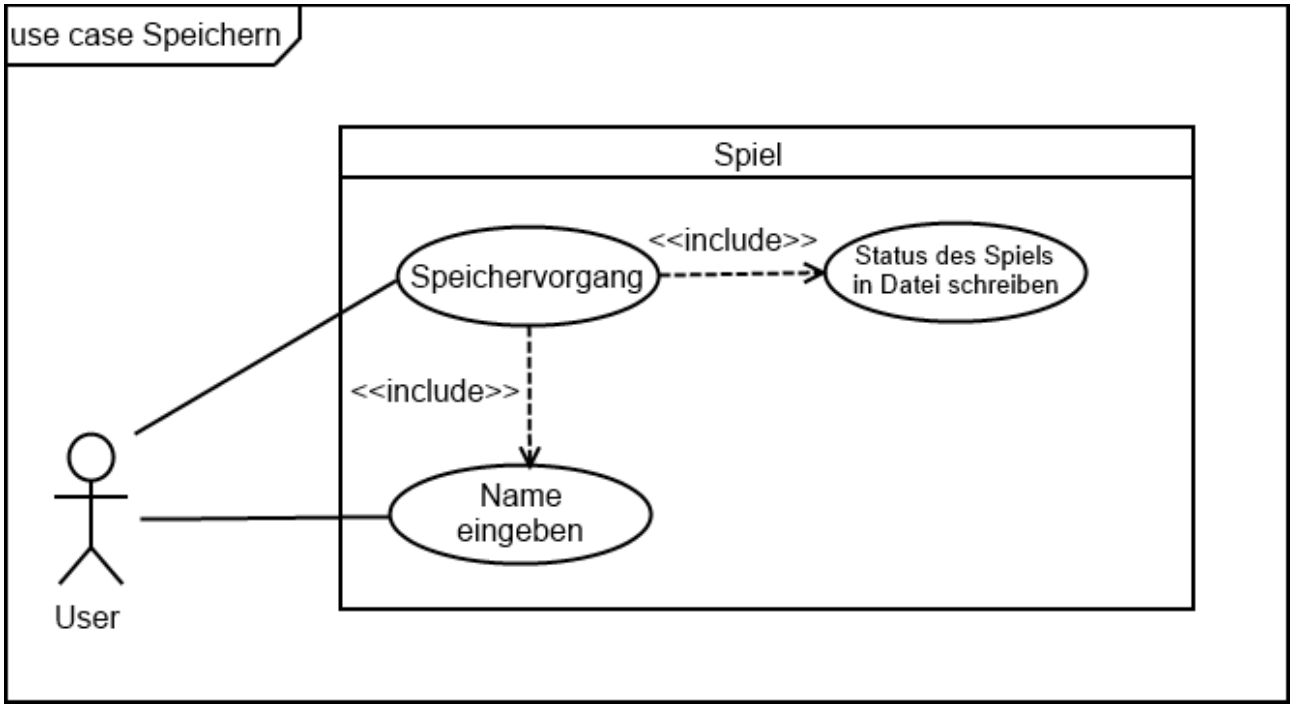
1. *Nestbau*



2. *Speichern*



3. *Laden*



Design Pattern:

- *Observer:*
  Implementiert durch den ParticleGenerator zur Steuerung und Erzeugung neuer Effektpartikel ohne dabei immer direkt über alle Partikel zu iterieren. Vielmehr bekommt jedes generierte Partikelelement durch den Generator eine eindeutige ID zugewiesen und hält eine Referenz auf diesen. Bei Änderungen aktualisieren die Partikel über diese Zuweisung ihren Status und übermitteln diesen an den Generator, welcher dann gegebenenfalls weitere Anpassungen vornimmt. Zum Beispiel sendet ein gestorbenes Partikel an den ParticleGenerator das es nun gelöscht werden kann, welcher dann die eigentliche Entfernung vornimmt.

Technische Mindestanforderungen:

Eingabegeräte:

- Maus
- Tastatur

Desktop/Laptop:

- *Prozessor:*
  Intel Core i5 M460 2,53GHz oder Vergleichbares
- *Grafikkarte:*
  ATI Mobility Redeon HD 5400 Series oder Vergleichbares
- *Arbeitsspeicher:*
  4 GB
- *Betriebsystem:*
  Windows 7 oder höher

Projektmanagement:

- Wöchentliche Treffen:
  Abgleich des derzeitigen Projektfortschrittes und direkte, detaillierte Absprache zukünftiger Arbeistpakte und Verantwortungen sowie des gemeinschaftlichen Arbeiten und Programmieren.

- Skype:
  Zur Kommunikation zwischen den Treffen, der Absprache von Terminen und der Klärung von kurzfristigen Problemstellungen und Lösungen.

- GitHub:
  Zur Versionierung und zum Austausch.

- Redmine:
  Zur Aufgabenverteilung, Meilensteinkommunikation und Planung von Arbeistpakten.