OTTO VON GUERICKE UNIVERSITÄT

Fakultät für Informatik

3D Game Project 2015

Softwareprojekt

Underlord

– Technical Design Document –

Patrick Schön, Dominik Schön und Max Frick

Inhaltsverzeichnis

1. Technische/technologische Vorarbeiten

1.1 Maya:

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

1.2 Photoshop:

Zur Erzeugung, Nachbearbeitung und Verbesserungen von Model-Texturen, den GUI-Elementen oder anderen Spritekomponenten. Auch war ursprüngliche geplant mittels Photoshop die Normalmaps zu generieren.

1.3 Sculptris:

Sculptris kommt nur dann zum Einsatz, wenn das in 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 anderer Modellierungstools zu erzielen, sollte dieser Versuch unternommen werden.

1.4 Ehemalige verwendete Programme:

Da nach dem Ausscheidens eines Teammitglieders einige Modelle übrig blieben, die mit keinem der oben genannten Tools entstanden sind, hielt wir es für angebracht diese der Vollständigkeit halber kurz aufzulisten. Dabei handelt es sich um die Folgenden:

- Blender
- Gimp
- Paint.Net

2. Verwendete Bibliotheken

2.1 Microsoft XNA Framework Redistributable 4.0:

Framework für C# zur Entwiklung von Spielen in Visual Studio 2010.

3. Zusätzliche Tools

- Visual Studio 2010
- GitHub
- TortoiseGit
- Skype
- Firefox
- OpenOffice
- Thunderbird

4. Formate

4.1 Modelformat

4.1.1 .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.

4.1.2 .OBJ:

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

4.2 Bildformat

4.2.1 .PNG:

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

4.3 Speicherformat

4.3.1 .TXT:

Einfaches Format für das Speichern und das Laden der Highscore-Punkte, sowie der Tutorial-Texte.

5. Softwarearchitektur

5.1 Klassendiagramme:

5.1.1 Allgemeine Übersicht:

Abbildung 1 stellt das zentrales Klassendiagramm dar, welches eine allgemeine Übersicht bietet und die wichtigsten Zusammenhängen zwischen den unterschiedlichen Software-Aspekten herstellt.

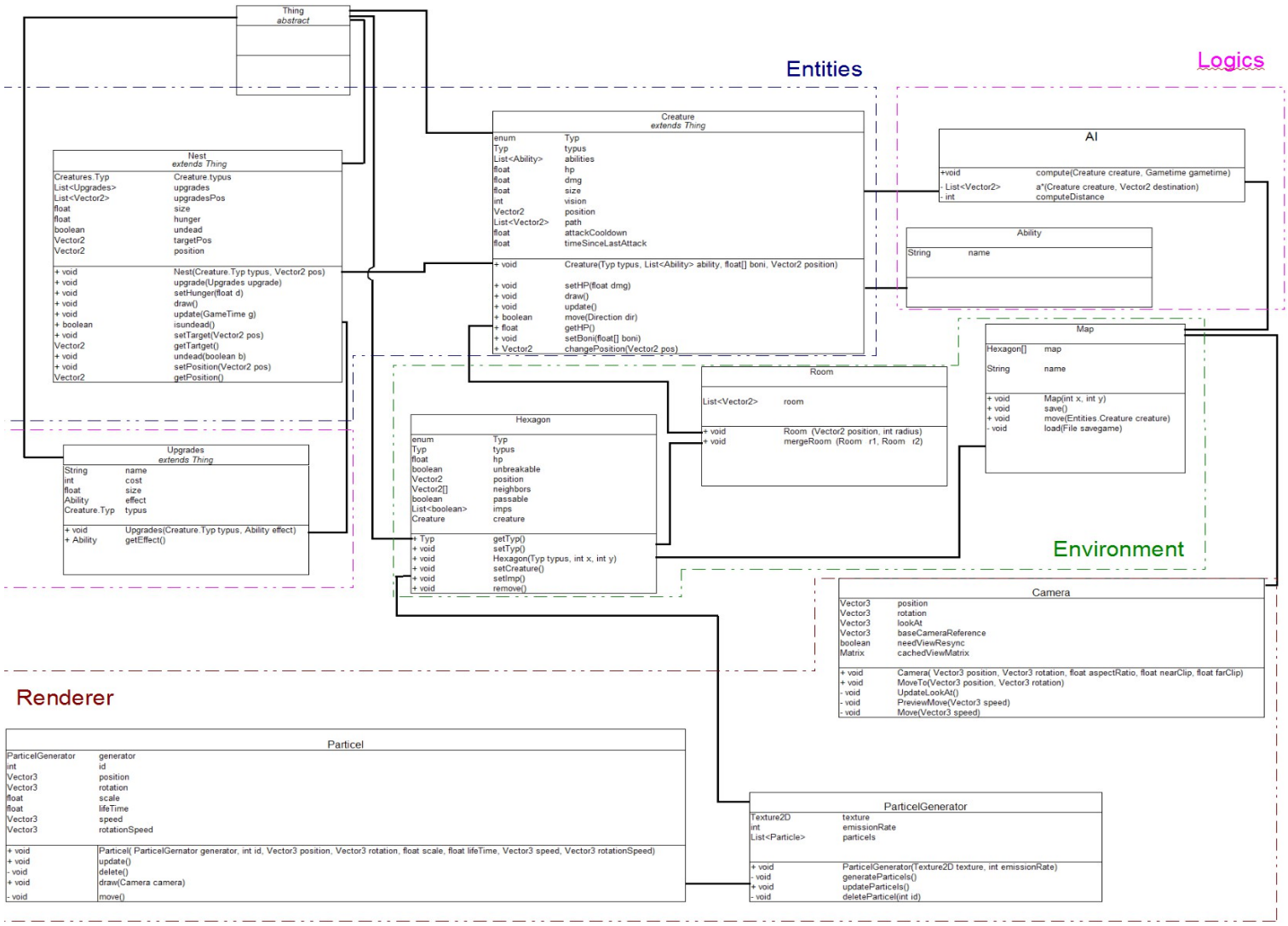


Abb. 1. Zentrales Übersichts-Klassendiagramm

5.1.2 Animations-Klassendiagramm:

In Abbildung 2 wird auf einer Klassen-Ebene noch einmal die, in der Dokumentation bereits ausführlich behandelte, Animations-Logik vorgestellt.

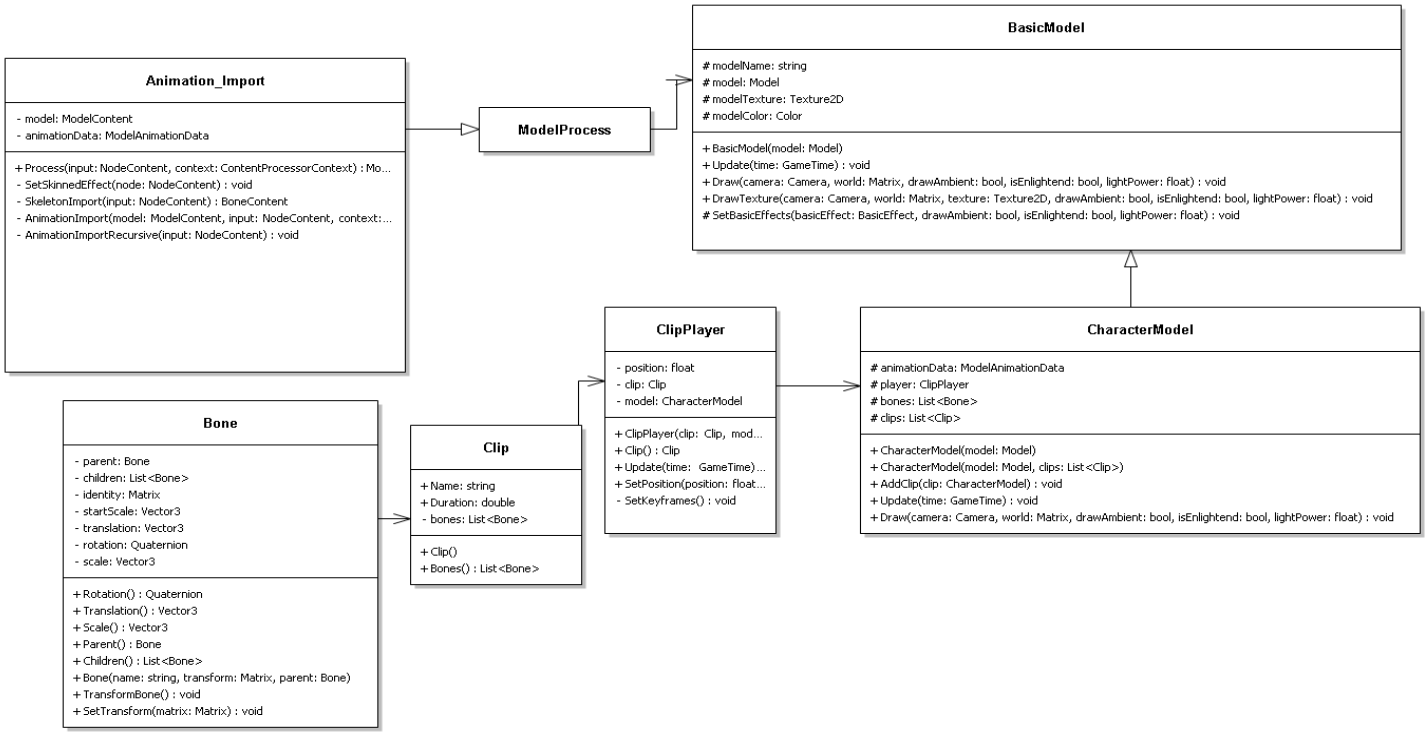


Abb. 2. Animations-Logik

5.1.3 GUI-Klassendiagramm:

Nachfolgend biete Abbildung 3 eine vollständige Übersicht über den GUI-Aufbau, unter Berücksichtigung sämtlicher Ableitungen.

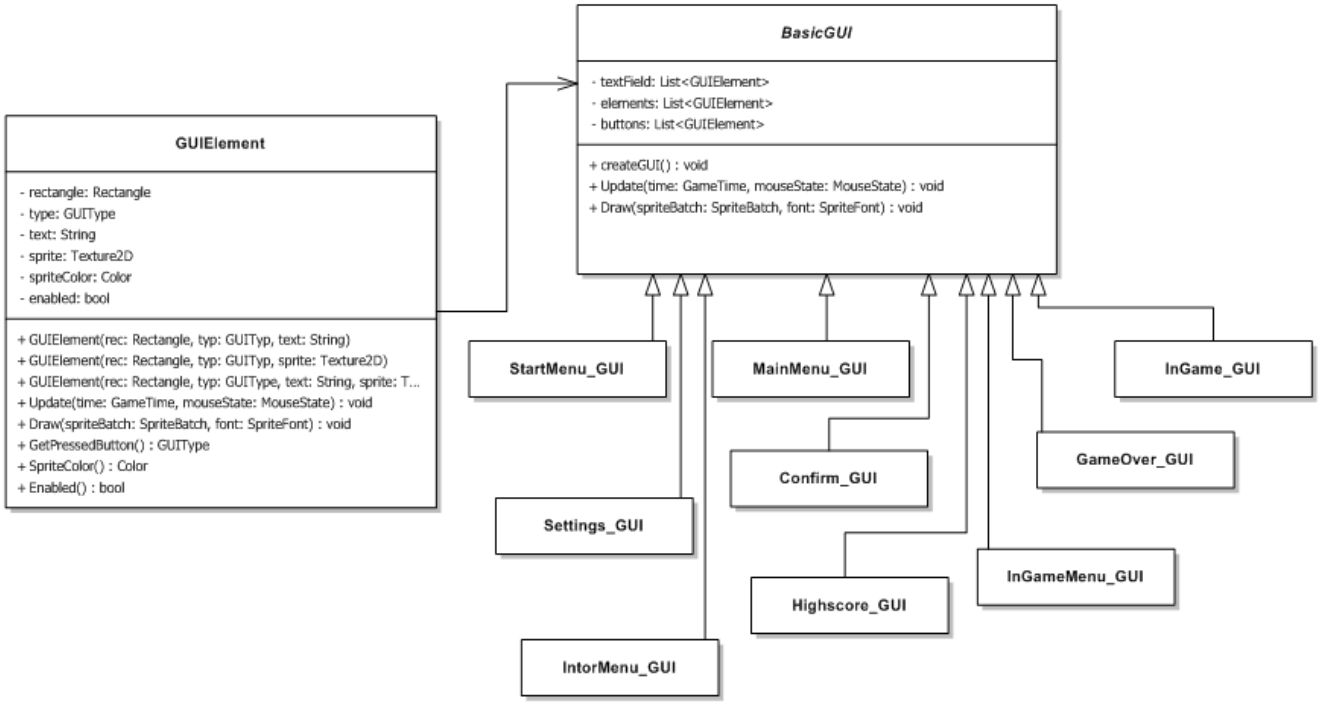


Abb. 3. Vollständiger *GUI*-Aufbau

5.2 Sequenzdiagramme:

5.2.1 Nestbau:

Abbildung 4 visualisiert den Ablauf für die Konstruktion eines Nests. Hierfür muss erst ein freier Raum geschaffen werden.

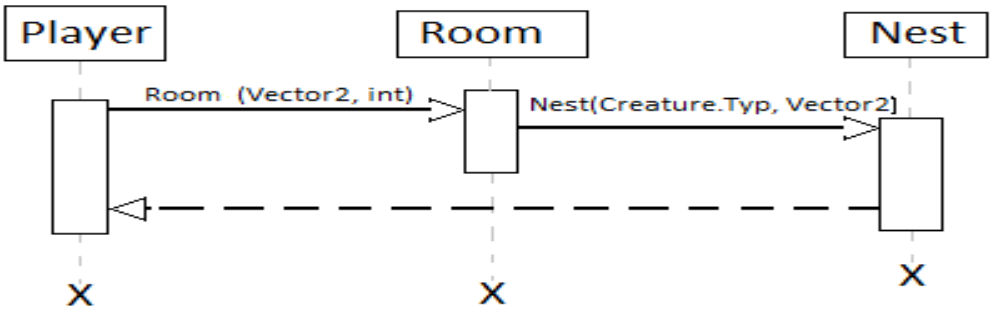


Abb. 4. Nestbau

5.2.2 Kampfablauf:

Abbildung 5 präsentiert die Abfolge eines Kampfes. Dabei wird eine Kreatur erst angreifen, wenn sie eine gegnerischer Held sieht und sich diese in ihrem Angriffsbereich befindet (hier mit 2/3 der Sichtweite angegeben).

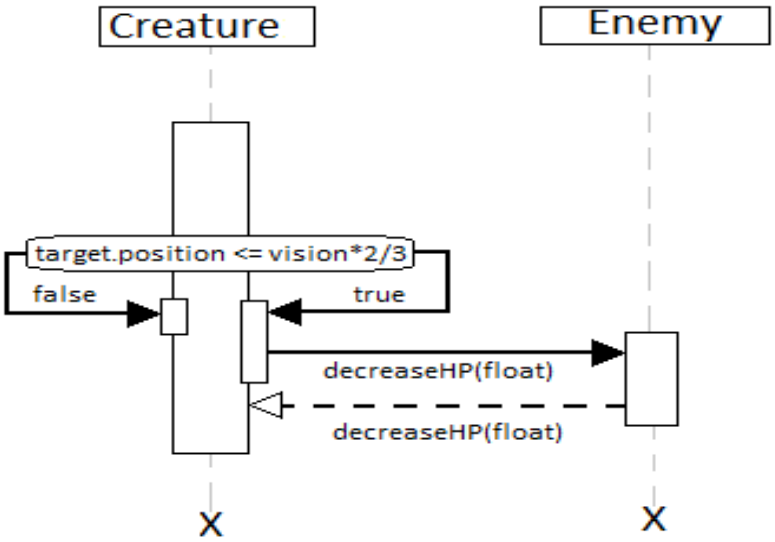
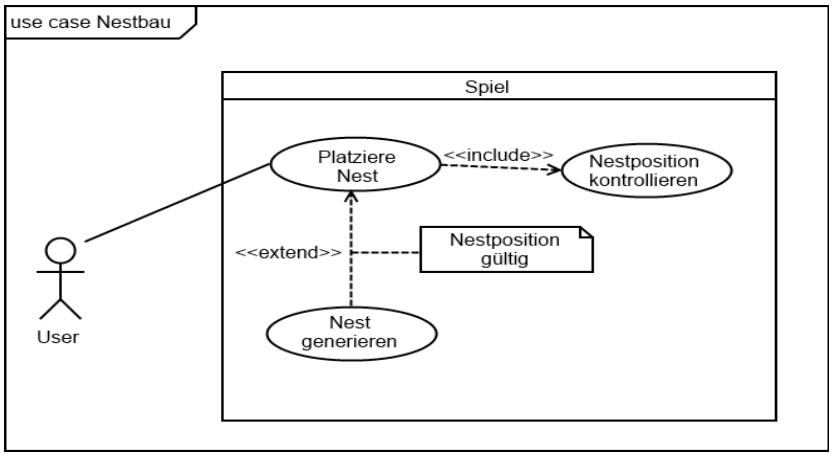


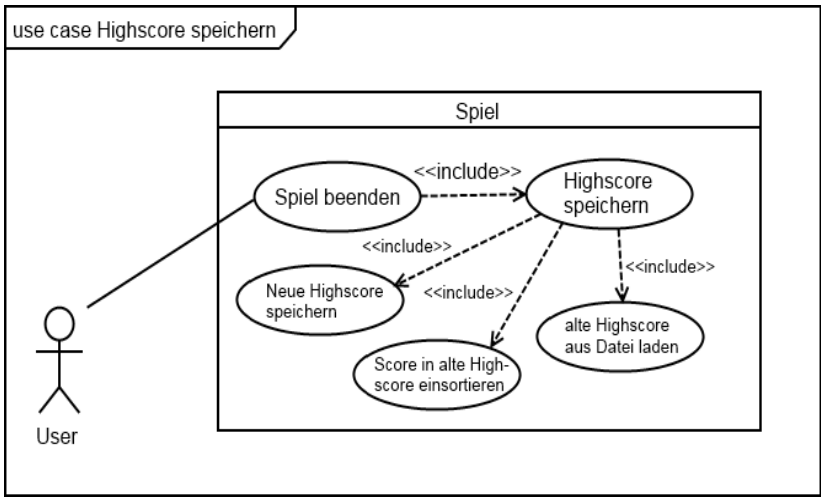
Abb. 5. Nestbau

5.3 Use-Case-Diagramme:

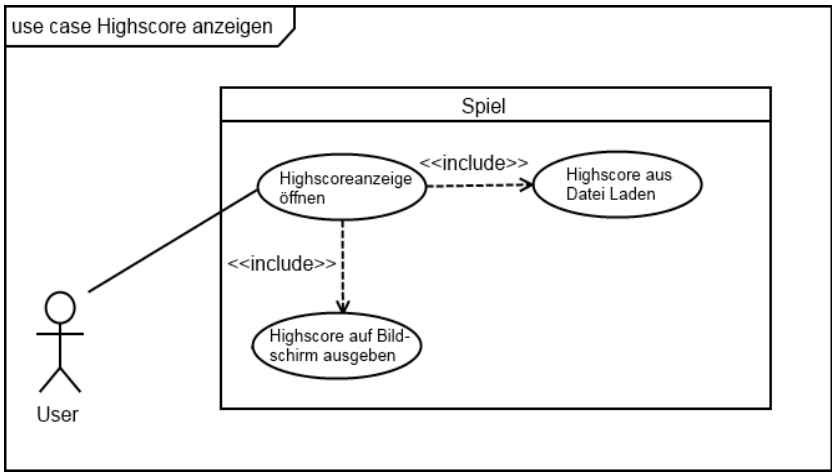
5.3.1 Nestbau:



5.3.2 Speichern des Highscores:



5.3.3 Anzeigen des Highscores:



5.4 Design Pattern:

5.4.1 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.
Abbildung 6 stellte diese Abhängigkeiten noch einmal visuell dar:

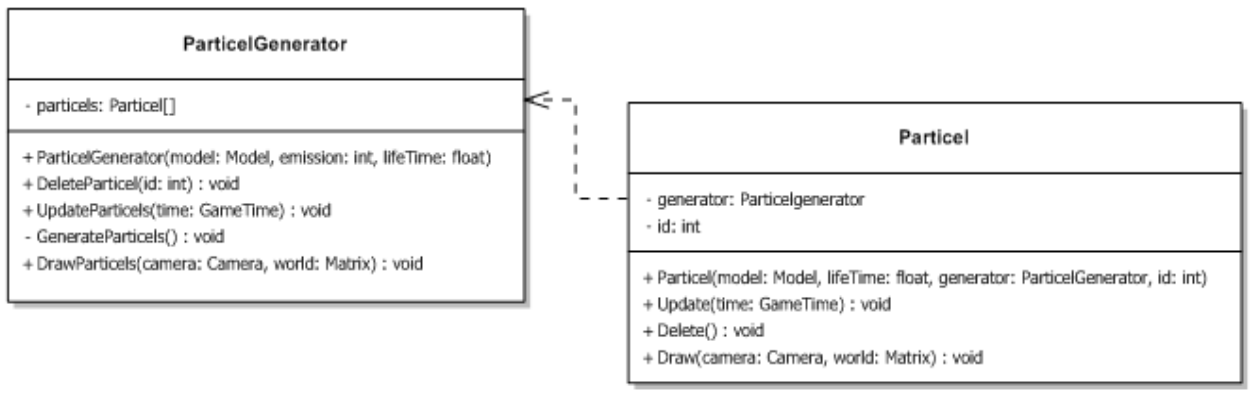


Abb. 6. Observer-Implementation des *ParticleGenerators*

6. Technische Mindestanforderungen

6.1 Eingabegeräte:

- Maus
- Tastatur

6.2 Desktop-PC/Laptop:

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

7. Projektmanagement

- Wöchentliche Treffen:
  Abgleich des derzeitigen Projektfortschrittes und direkte, detaillierte Absprache zukünftiger Arbeistpakte und Verantwortungen, sowie das gemeinschaftliche Arbeiten und Programmieren.
- Skype:
  Zur Kommunikation zwischen den Treffen, der Absprache von Terminen und der Klärung von kurzfristigen Problemstellungen und Lösungen.
- GitHub:
  Standardprogramm zur Versionierung und zum Austausch des gesamte Projekts.
- Redmine:
  Zur Aufgabenverteilung, Meilensteinkommunikation und Planung von Arbeistpakten.