Scharfschießen

Dokumentation

Kathleen Hübel, Ramazan Gündogdu, Linda Schey, Susanne Schmidt, Tobias Winterhalder

Juni 2014

Game Production

Fakultät Digitale Medien

Hochschule Furtwangen University

Inhalt

Vorwort

Das Computerspiel „Scharfschießen“ entstand im Sommersemester 2014 in der Veranstaltung *Game Production* unter der Leitung von Prof. Christoph Müller an der Hochschule Furtwangen University an der Fakultät Digitale Medien.

Aufgabe war es mit der *Furtwangen University Simulation and Entertainment Engine*[1] unter dem Überbegriff *Playbar* ein Computerspiel zu entwickeln. Das Spiel sollte dafür geeignet sein, heimlich während der Arbeit gespielt zu werden. Die Fenstergröße des Spiels sollte nur einen kleinen Bereich des Bildschirmes einnehmen und zwar entweder als schmale Leiste am oberen Bildschirmrand oder an der Seite des Bildschirmes. Die *Playbar*.

Ziel der Aufgabe war neben dem Erstellen des Spiels, das Kennenlernen der Production-Pipeline bei der Erstellung von Videospielen.

Im Folgenden wird eine kurze Übersicht zu den einzelnen Bestandteilen des Spiels gegeben und jeweils von der im Team dafür hauptverantwortlichen Person bzw. Personen beschrieben.

[1] Fusee, siehe: http://fusee3d.org/

Idee

Team (Kathleen Hübel, Ramazan Gündogdu, Linda Schey, Susanne Schmidt, Tobias Winterhalder)

Die Spielwelt in der der Spieler sich befindet, ist ein Bauernhof. Der Spieler befindet sich in der Mitte des Bauerhofes. Über dem Bauernhof kreisen fliegende Schafe. Der Spieler hat die Aufgabe, mit Tomaten so viele Schafe wie möglich innerhalb eines definierten Zeitraums abzuwerfen. Für getroffene Schafe bekommt der Spieler je nach dem wie weit das getroffene Schaf entfernt ist Punkte. Je nach Punktestand steigt der Spieler ein Level auf und der Schwierigkeitsgrad wird erhöht.

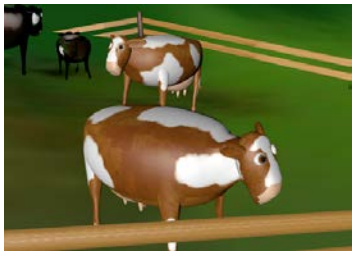
Ist die Zeit abgelaufen, kann der Spieler seinen erreichten Punktestand in eine Highscoreliste eintragen und sich mit den besten Fünf vergleichen.

Design

Kathleen Hübel, Tobias Winterhalder

Tiere

Die Tiere, Kühe, Hühner und Schafe, wurden vollständig im Cinema 4D erstellt. Nach mehreren Alternationen wurde diesen, besonders durch unnatürlich große Augen und abgerundeten Flächen und selbsterstellte Texturen eine comichafte Erscheinung verliehen. Die Farbgebung ist dennoch schlicht und in naturfarben gehalten. Zusätzlich sind noch die Flügel der Schafe zu erwähnen, die bereits seit den Anfängen der Konzeption feststanden, um dem Spiel einen witzigen verqueren Touch zu verleihen.



Gebäude und zusätzliche Objekte

Die Gebäude, bestehend aus einer alten Scheune, einer Kapelle und dem Schwarzwaldbauernhaus sind getreu dem Aussehen von typischen Schwarzwaldgebäuden und aus zahlreichen Detaileinbringungen entstanden.

Objekte wir ein Traktor, Zäune die eine Koppel andeuten aber auch die saftigen Tomaten, die statt der üblichen Gewehrmunition als Schießobjekt dienen sollen, unterstreichen den Bauernhoflook zusätzlich. Auch diese sind zum Teil recht einfach und grob gehalten und an das comichafte Aussehen der Tiere anzuknüpfen.



Landschaft

Die Landschaft, wie auch die einzelnen Objekte sind einer Schwarzwaldlandschaft nachempfunden. Durch Berge mit tiefgrünen Tannen und Wiesen in satten Grüntönen könnten die erstellten Objekte zielgerichtet eingebaut werden und speziell für einen 360° Blick angeordnet werden.

Einige Objekte, wie die Tannen, Hühner oder auch die bergige Landschaft wurden zu Gunsten der Objektgröße sowie der Kompatibilität mit Fusee stark vereinfacht oder gar weggelassen.

Programmierung

Ramazan Gündogdu, Linda Schey, Susanne Schmidt

Folgende Klassen wurden für das Spiel erstellt.

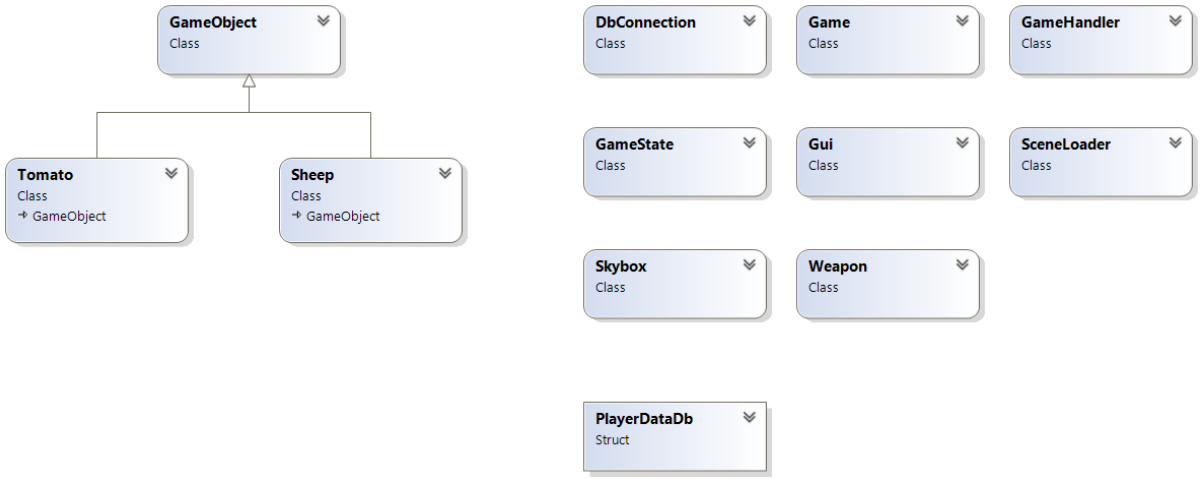


Abbildung 1: Klassen Übersicht

Die Klasse *Gui* wird im Unterkapitel GUI näher erklärt und die Klasse *DbConnection* im Kapitel Datenbank. Die restlichen Klassen werden im Kapitel Game Logik kurz erläutert, sowie der Zusammenhang, der zwischen den Klassen herrscht. Um den Rahmen nicht zu sprengen, werden die Klassen nicht allzu detailliert beschrieben, da der Source Code vorliegt.

Game Logik

Linda Schey

Die Klasse *GameHander* ist Dreh und Angelpunkt des Spiels. Beim Start der Applikation wird eine Instanz dieser Klasse erstellt, welche die ganze Zeit während das Spiel ausgeführt wird, egal ob pausiert oder nicht, vorhanden ist. Der *GameHandler* enthält und instanziiert alle für den Spielablauf relevanten Klassen: *Gui*, *GameState*, *DbConnection*. Die *Update()* Methode des *GameHandlers* wird in der *Main* Klasse pro Frame aufgerufen und dient als Game-Loop. Des Weitern ruft die Main Klasse jedes Frame die Methode *Hide()* des *GameHandlers* auf, die es dem Spieler zu jedem Moment ermöglicht das Spiel zu pausieren und zu verstecken.

Die Klasse *GameState* enthält den aktuellen Status des Spiels. Das Spiel kann sich je nur in einem der folgenden Status befinden: MainMenu, Playing, HidenPause, Highscore. Diese sind in einem *enum* definiert. Der Status kann durch Ereignisse im Spiel (z.B. Ablauf der Zeit) oder Eingaben des Spielers geändert werden. Des Weitern wird auch immer der vorherige Status gespeichert, damit das Spiel bei einem Wechsel aus dem HidePause Status (wenn der Spieler das Spiel versteckt und somit pausiert) zu seinem vorherigen Zustand zurückkehren kann.

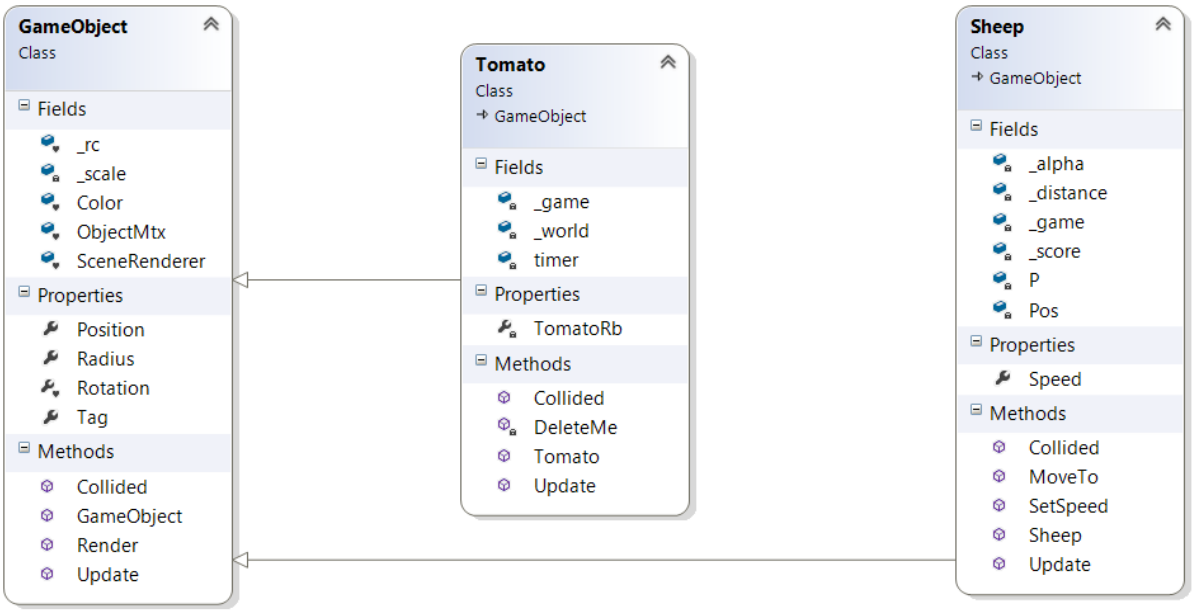


Abbildung 2: GameObjects

Die Klassen *Sheep* und *Tomato* sind, wie in dem obigen Diagramm zu sehen, aus der Klasse *GameObject* abgeleitet. Die Klasse *GameObject* enthält Methoden, die für jedes veränderbare Objekt im Spiel zutreffen (z.B. *Update()* und *Render(), …*). So ist es möglich eine Liste von *GameObjects* zu erstellen, die sowohl einfache *GameObjects* enthält aber auch Objekte vom Typ *Tomato* und *Sheep.* So kann für jedes Objekt der Liste pro Frame die Methode *Update()* und Render*()* aufgerufen werden.

Die Klasse *Tomato* enthält einen Rigidbody, der vom Spieler „geworfen“ wird. Anhand der Position des Rigidbody wird das Mesh der Tomate dargestellt. Trifft die Tomate auf ein Objekt der Klasse *Sheep* bekommt der Spieler Punkte gutgeschrieben. Geht die Tomate ins Leere, zerstört sie sich nach einer gewissen Zeit selbst.

Die Klasse *Sheep* berechnet für jedes einzelne Objekt dieses Typs die Position und anhand derer die Punkte, die ein Treffer des entsprechenden Schafes wert ist. Außerdem wird die Geschwindigkeit, mit der sich ein Schaf bewegt, durch den Abstand des Schafs zum Spieler und dem Level berechnet.

Die Klasse *Game* ist sehr groß und koordiniert viele verschiedene Dinge gleichzeitig. Zum einen werden die Eingaben des Spielers eingelesen und zum anderen werden alle *GameObjects* upgedated und gerendert. Des Weiteren wird abgefragt ob Objekte (*Tomato* und *Sheep*) kollidieren.

Die Klasse *SceneLoader* ist lediglich dafür zuständig, Assets zu laden und den *GameObjects* zugänglich zu machen.

Die Klasse *Skybox* hat nur den Zweck eine *Skybox* zu erstellen und zu rendern. Allerdings treten hierbei auf manchen Rechnern Probleme auf.

GUI

Susanne Schmidt

Das grafische User Interface regelt die Kommunikation zwischen dem Spiel und dem User.

Die wichtigsten Unterscheidungen sind die verschiedenen Ansichten/Funktionen während den verschiedenen Game-Status:

Main-Menü:

Das Startmenü enthält lediglich den Spieltitel und einen Button, mit dem das Spiel gestartet werden kann und ein Bild, mittels dem die Spielsteuerung erläutert wird.

Während dem Spiel:

Sobald der Spieler in den Status „Playing“ kommt, wird in der Gui die *MainMenu*-Methode aufgerufen. Dabei wird zunächst abgefragt, ob der Spieler das Spiel gerade neu begonnen hat, oder aus der Pause zurück ins Spiel kommt. Im zweiten Fall bleiben alle Informationen, die vor der Pause aktiv waren, unverändert (Zahl der Tomaten sowie die Zeit- und Punkteanzeige). Wenn das Spiel neu gestartet wurde, werden alle diese vorherigen GUI-Elemente entfernt und mit den Startwerten neu initialisiert.

Aus der *Game*-Klasse werden pro Frame der Countdown und die Zeit abgefragt und in den oberen Ecken des Bildschirms angezeigt.

Gültige Nutzereingaben sind in diesem Status nur die Mausbewegung und die Tabulator-Taste (für die Pause). Alle anderen Eingaben, wie zum Beispiel die Namenseingabe, sind in dieser Zeit durch die Gui deaktiviert.

Durch das Schießen mit der Maus wird die Information über die verbleibende Munition aus der *Game*-Klasse abgefragt und durch die Anzeige der entsprechenden Anzahl an Tomaten am unteren linken Bildschirmrand an den User kommuniziert.

Erreicht der Nutzer eine Levelaufwertung, wird diese Information kurz in der Mitte des Bildschirms eingeblendet. Dies wird durch einen kleinen Timer in der GUI realisiert.

Highscore-Menü:

Nach dem Spiel bekommt der Spieler zunächst seinen Spielstand angezeigt. Darunter kann er seinen Namen eingeben, um sich danach inklusive seiner erreichten Punkte und Level in den Highscore einzutragen.

Über den Button „*Play Again*“ wird das Spiel neu gestartet, wobei alle GUI-Informationen außer dem eingegebenen Namen zurückgesetzt werden. So muss der Spieler seinen Namen nicht jedes Mal neu eingeben.

Die Namenseingabe wird durch Input-Instanzen realisiert, die für jeden erlaubten Buchstaben implementiert werden mussten.

Bestätigt der Spieler seinen Namen per Klick auf den Butten „*Enter in Highscore*“, wird die Information aus der GUI in die Datenbank übertragen und direkt die aktuellen Werte abgefragt, um die aktuellen ersten 5 Einträge des Highscores anzuzeigen.

Die Credits verbergen sich hinter einem Button auf einem Klohäuschen, das sich beim Klick auf den Button öffnet und die Namen einblendet.

Datenbank

Ramazan Gündogdu

Die Datenbankentwicklung wurde mit dem relationalen Datenbankverwaltungssystem MySQL realisiert. Die Integration der MySQL mit der Visual Studio-Umgebung wurde mit dem MySQL .Net Connector durchgeführt. Die Kommunikation mit dem Datenbankserver ist in der Klasse *DbConnection* implementiert. Die Datenbank besteht aus einer Tabelle, die aus u.a drei Spalten besteht. In diesen drei Spalten werden Name, Punkte von Spielern und das Level, das der Spieler erreicht hat, gespeichert.

key	PlayerName	HighScore	Level
10381	Anonymous	2500	6
10380	Ramo	2400	5
10379	Sunny	4550	10
10378	Linda	3500	8
10377	Katleen	4100	9
10376	Tobias	1500	2

Tabelle 1: Datenbank Tabelle

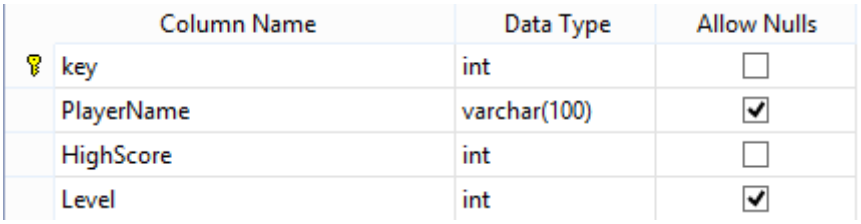


Abbildung 3: Entwurf der Tabelle

Diese empfangenen Informationen werden in der Klasse *DbConnection* mit der Methode *Insert()* in die Datenbank gespeichert und nach der Beendung des Spiels wieder aus der Datenbank gelesen und die ersten fünf Einträge ausgegeben. Im Folgenden ist das SQL-Script der Tabelle zu sehen.

```
CREATE TABLE `PlayerPoints` (`key` int(11) NOT NULL AUTO_INCRE
MENT, `PlayerName` varchar(100) DEFAULT NULL, `HighScore` int(
100) NOT NULL, `Level` int(11) DEFAULT NULL, PRIMARY KEY (`key
`)) ENGINE=MyISAM AUTO_INCREMENT=10455 DEFAULT CHARSET=latin1
```

Die herkömmlichen CRUD[2]-Datenbankoperationen wurden in der Klasse so implementiert, dass man bei einem nötigen Aufruf darauf zugreifen kann. Folgende Abbildung zeigt das Klassendiagramm der Klasse *DbConnection*

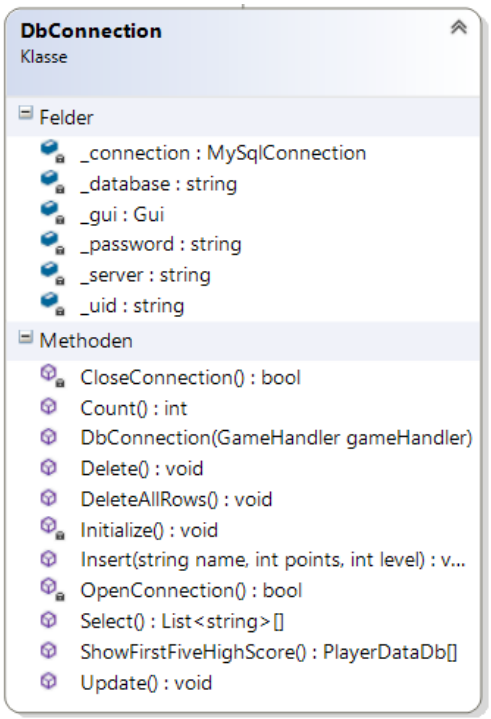


Abbildung 4: Klassendiagramm von DbConnection

[2] Create (Datensatz anlegen), Read (Datensatz lesen), Update (Datensatz aktualisieren) und Delete (Datensatz löschen).

Abbildungsverzeichnis

Tabellenverzeichnis