Twilighthouse

A game prototype by
Martin Schreiber, 0425891
Andreas Poppernitsch, 0425537
Albert Kavelar, 0426856

Allgemeines

In unserem Spiele-Prototyp lässt sich der Charakter, derzeit noch ein Teapot, durch die Welt bewegen. Diese besteht aus einem per Heightmap erzeugtem Terrain und zwei texturierten Würfeln. Umgeben wird das ganze von einer Skybox. In der Welt lauert auch schon ein Gegner auf ihn, ein feindseliger Teapot.

Steuerung

Die Figur lässt sich mit den Tasten A, S, D und W sowie mit der Maus bewegen.

W ..... vor
S ..... zurück
A ..... links
D ..... rechts
Linke Maustaste ..... schießen
Maus bewegen ..... umherschauen
ESC ..... Spiel beenden

Features

Derzeit kann man den Charakter durch die Welt bewegen und einzelne Schüsse abfeuern. Wenn man sich in die Nähe des zweiten Teapots bewegt, rast dieser auf den Spieler zu. Bei Kollision erfolgt in der vorliegenden Version lediglich eine Textausgabe, später soll der Charakter Energie verlieren.

Datenstrukturen

Zurzeit verwenden lediglich der TexCube und Terrain eine primitive Datenstruktur. Diese beschränkt sich beim Texcube darauf, dass er seine 8 Eckpunkte und seine 6 Normalvektoren in eigenen Vektorarrays abspeichert. In der draw-Methode wird dann über die Indizes der Eckpunkte auf sie zugegriffen. Die Texturkoordinaten werden noch fix in der draw-Methode angegeben.

Texturierte Objekte

Es gibt im Moment erst zwei texturierte Objekte: den TexCube und die SkyBox. Bei ersterem wird eine TGA-Textur geladen, bei letzterem eine Bitmap. Das Texturemapping erfolgt quasi manuell, über Angabe der Texturkoordinaten in der entsprechenden draw-Methode. Bei komplexeren Objekten als Würfeln wäre dies nicht mehr so einfach möglich, man müsste UV-Maps verwenden.

Kameramodell

Wir haben eine 3rd-Person-Kamera implementiert. Die zentrale Funktionalität dafür stellt die Klasse Camera zur Verfügung. Man kann die Kamera frei mit der Maus nach oben und unten drehen. Bei gedrückter mittlerer Maustaste kann man die Kamera durch vor- und zurückbewegen der Maus zoomen. Links- und Rechtsrotation rotieren den Charakter mit der Kamera.

Bewegte Objekte

Neben dem Charakter selbst ist auch schon ein bewegter Geist (goldener Teapot) im Spiel vorhanden. Damit dieser aktiv wird, muss man sich in seine unmittelbare Nähe begeben. Dann beginnt er den Spieler zu verfolgen und stürzt immer wieder auf ihn zu. Weiters kann man mit der linken Maustaste einen Schuss abfeuern, der als sich weg bewegende Kugel dargestellt wird.

Beleuchtung und Materialien

Es gibt eine globale, unendlich weit entfernte Lichtquelle, die in der initLighting()-Methode in der main.cpp definiert wird und auf alle Objekte einwirkt. Es handelt sich hierbei um ein directional light.

Die verschiedenen Objekte (Character, Ghost, Terrain, TexCube) verfügen über individuelle Materialeigenschaften, die mit glMaterial* definiert werden. Da wir zum Teil glut-Objekte verwenden, müssen wir uns noch nicht um alle Normalvektoren selbst kümmern. Beim TexCube werden sie jedoch manuell angegeben und beim Terrain berechnet.

Spieldesign
Dieses ist erst rudimentär verfügbar. Die vorliegenden Möglichkeiten lassen aber schon erahnen, welch dichte Atmosphäre im fertigen Spiel vorherrschen wird. Furchterregende Verfolgungsjagden mit Geistern, bizarre, surreale schwebende Objekte und noch mehr – all das ist Twilighthouse!

Gegenwärtig kann man lediglich einzelne Schüsse abfeuern und vor dem Geist davonrennen. Man kann noch nicht sterben oder Energie verlieren und braucht sich deshalb auch noch nicht zu wehren. Außerdem ist das Kernkonzept, dass man den Sichtbereich auf einen kleinen Lichtkegel reduziert, noch nicht vorhanden. Hier gibt es also noch einiges zu tun.

Implementierung

main.cpp
Diese Datei stellt dein Einstiegspunkt in das Programm dar. Hier erfolgen alle wichtigen Initialisierungen, wie beispielsweise das Setzen der Callback-Funktionen, das Initialisieren des Rendering-Modus oder globale Beleuchtungseinstellungen. Hier befindet sich auch die zentrale Zeichenschleife drawScene(), von der aus die draw-Methoden der einzelnen Objekte und jene des HUD aufgerufen werden.

Vector3d.cpp
Hilfsklasse für Vektoren mit x, y und z-Komponente. Diese Klasse ermöglicht es, einfach mit Vektoren zu rechnen und diese im Bedarfsfall zu Arrays zu konvertieren.

TextureLoader.cpp
Diese Klasse soll in weiterer Folge zu einem vollwertigen Texturloader ausgebaut werden. Im Moment können damit nur TGA-Bilder geladen werden. Der Code wurde von nehe.gamedev.net übernommen.

TexCube.cpp
Diese Klasse repräsentiert einen texturierten Würfel. Er verfügt über Materialeigenschaften und eine Referenz auf den Texturloader, damit er seine Textur laden kann. Der Würfel setzt sich aus 12 Dreiecken zusammen, deren Eckpunkte im Array _vertices gespeichert werden. Für jede der 6 Seiten des Würfels gibt es einen entsprechenden Normalvektor, der im Array _normals gespeichert wird. Ein TexCube kann durch den Aufruf seiner draw-Methode gezeichnet werden. In dieser Methode werden auch die Materialeigenschaften geladen.

Terrain.cpp
Diese Klasse lädt eine Bitmap, dessen Graustufen als Höhenwerte interpretiert werden. Der Code zum Laden der Bitmaps stammt von nehe.gamedev.net. Für jeden aus den Höhenwerten erstellten Eckpunkt wird ein Normalvektor errechnet. Das Terrain verfügt, wie auch der TexCube, über individuelle Materialeigenschaften, die in der draw Methode geladen werden.

Skybox.cpp
Mit dieser Klasse werden Hintergrundlandschaft und Himmel im Spiel erstellt. Es handelt sich hierbei um einen Würfel, dessen Seiten alle unterschiedlich texturiert sind. Damit man die Kanten des Würfels, die sich bei der Beleuchtung ergeben würden, nicht sieht, wird diese deaktiviert, während die Skybox gezeichnet wird. Die Geometriedaten sind derzeit noch in der draw-Methode hardcodiert enthalten.

MemoryLeakTracker.cpp
Hilfsklasse zum Auffinden von Memoryleaks.

ImageLoader.cpp
Hilfsklasse zum Laden von Bitmaps, der Code stammt von nehe.gamedev.net. In Zukunft soll die Funktionalität des ImageLoaders in den TextureLoader wandern, der zu einer Sammelstelle zum Laden von Bildern unterschiedlichster Formate werden soll.

Ghost.cpp
Mit dieser Klasse können Gegner im Spiel erstellt werden. Sie verfügt über eine draw-Methode, in der der Geist, derzeit noch ein Teapot, gezeichnet wird. Der Geist verfügt über eigene Materialeigenschaften, die in der draw-Methode geladen werden. Die move-Methode stellt die Logik zur Bewegung des Geistes zur Verfügung.

Ein Geist kann verschiedene Zustände haben: wait, move to character, attack character und stop. Zu Beginn befindet er sich im Zustand wait. Sobald die euklidische Distanz zwischen Geist und Character unter einem bestimmten Wert liegt, wird er aktiv und fährt auf den Charakter zu.

Character.cpp
Der Hauptcharakter unseres Spiels.

Camera.cpp
Implementierung einer 3rd-Person-Kamera. Diese Klasse ist für das korrekte Platzieren und Ausrichten der Kamera zuständig und regelt den Abstand von Kamera zu Character.

Bullet.cpp
Klasse zum Zeichnen von Schüssen. Diese werden derzeit einfach als einfache Kugeln dargestellt.

Fremder Sourcecode

Die verwendeten Methoden zum Laden von Bildern sind an Soucrecode von nehe.gamedev.net angelehnt oder zur Gänze übernommen worden.