Submission 2: Daisy Quest

Gameplay

Die Welt von Daisy Quest ist in 3D implementiert. Dazu wird eine Heightmap geladen, auf der sich die Spielfigur frei bewegen lässt. Ziel des Spiels ist es, genügend Blumen einzusammeln, um das Level zu schaffen. Die Anzahl der zu sammelnden Blumen findet sich am rechten, oberen Bildschirmrand. Hier sieht man auch seinen aktuellen Stand an gesammelten Blumen und die Anzahl der Blumen, die insgesamt noch “sammelbar” sind (dh. noch nicht von Daisy gesammelt oder den Ziegen gefressen wurden). Das Einsammeln erfolgt, indem man zur jeweiligen Blume läuft. Die Steuerung der Spielfigur ist über die WASD-Tasten oder den Pfeiltasten möglich. Ziel des Spieles ist es so schnell wie möglich die geforderte Anzahl an Blumen einzusammeln, indem Daisy diese einfach berührt. Doch Vorsicht - auch wenn eine Ziege eine Blume berührt, verschwindet diese, da die Blume von der Ziege gefressen wird.
Die Ziegen laufen immer Daisy entgegen, da sie unbedingt die von Daisy gesammelten Blumen auffressen wollen :) Erreicht eine Ziege Daisy, so werden fünf Blumen abgezogen, die die Ziege quasi friest. Die entsprechende Ziege verschwindet sogleich, um ein neuerliches Abziehen vom

Blumenkonto zu vermeiden. Um sich zu verteidigen, ist es möglich die Ziegen mit Hilfe der rechten Maustaste abzuschießen.
Wenn nicht mehr genug Blumen auf der Welt vorhanden sind, um das Ziel von 20 gesammelten Blumen zu erreichen, ist das Spiel verloren. Es wird eine Information auf dem Bildschirm ausgegeben, dass das Spiel verloren ist und das Spiel wird von diesem Augenblick an nicht mehr upgedated. Selbiges passiert sofern man das Spiel gewonnen hat. Mit Enter kann man anschließend einen neuerlichen Versuch starten, 20 Blumen zu sammeln und sich vor den hinterlistigen Ziegen in Sicherheit zu bringen.

Beleuchtung

Unsere Welt wird von einem direktionalen Licht beleuchtet. Im Unterschied zu einer Punktlichtquelle fällt so das Licht auf jeden Punkt der Map von der gleichen Richtung. Für die Beleuchtung haben wir eine eigene Licht-Klasse implementiert, welche die Lichtinformation speichert und für alle Shader-Uniforms verwendet wird. Mit der F7-Taste kann dieses direktionale Licht um die Y-Achse (Achse nach oben) rotiert werden. So kann das Wandern der Schatten beobachtet werden.

Komplexe Objekte

Im Spiel sind die unterschiedlichsten komplexe Objekte in Form von den unterschiedlichsten Modellen eingebaut. Im folgenden Abschnitt wird auf die jeweiligen Objekte eingegangen und auf deren Quellen hingewiesen.

Daisy

Bei Daisy handelt es sich um die Hauptfigur unseres Spieles. Es soll ein süßes kleines Monster darstellen. Dieses Modell haben wir selber mit Hilfe von Blender angefertigt.

Blumen

Die Blumen sind die Objekte, die Daisy zum Erreichen des Spielzielss sammeln muss. Anhand der Blume wird die Transparenz von Objekten umgesetzt.
Quelle der Blume: *http://www.cgtrader.com/3d-models/plant-tree/flower?tag=fl30* (Zu dem Zeitpunkt an dem wir das Model runtergeladen haben, war es noch kostenlos.)

Haus

Das Haus befindet sich in der Mitte der Szene und hat eine normalgemappte Texturierung.
Quelle des Hauses: *http://www.mbsoftworks.sk/index.php?page=tutorials&series=1&tutorial=24*

Windmühle

Die Windmühle befindet sich im Zentrum der Heightmap und wird zur Implementierung von Hierachical Animations verwendet. Das Rotorblatt dreht sich eine Richtung, während sich die kleinen Rotorblätter in die Gegenrichtung drehen.
Quelle der Windmühle: *http://www.3dvia.com/models/B6229AACBE90A2B4/wind-turbine*

Ziegen

Die Ziegen sind die Gegner unserer Daisy, da sie zum einen die Blumen in der Szene fressen, sobald sie diese berühren und zum anderen, bei Berührung von Daisy für einen Abzug an gesammelten Blumen sorgen. Die Ziegen sind so programmiert, dass sie nach deren Hinzufügen zur Szene immer in Richtung Daisy laufen und sich das Modell dementsprechend zur Daisy dreht. Um sich vor den Ziegen zu schützen, kann Daisy diese abschießen. Um es dem Spieler allerdings nicht allzu leicht zu machen, tauchen immer wieder neue Ziegen auf.
Quelle der Ziegen: *http://tf3dm.com/3d-model/goat-44811.html*

Zaun

Der Zaun am Rand der Welt verhindert, dass wir nicht von der Heightmap runterfallen. Er befindet sich rund um die Szene.
Quelle des Zauns: *http://tf3dm.com/3d-model/simple-fence-68101.html*

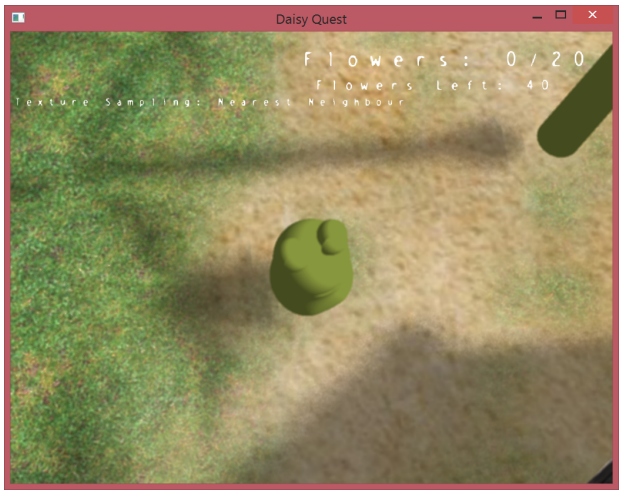
Effects

Für die erforderlichen 4 Effektpunkte haben wir folgende Effekte implementiert:

Shadow Maps (1.5)

Um dem Spiel und allen Modellen einen entsprechenden Schatten zu verleihen, wurde Shadow Mapping mit PCF implementiert. Dabei wird zuerst die Szene mit allen Objekten von der Position der Lichtquelle aus in eine Textur (Shadowmap) gerendert (1. Pass). Diese Schatten-Information wird dann beim tatsächlichen berücksichtigt. Um unschöne Artefakte beim Schatten zu vermeiden, wird die Shadowmap mehrere Male gesampled und dabei die Schatten-Intensität (visibility-factor) angepasst.

```
for(int i=0; i < 4; i++) {
        int index =
int(mod(16.0*random(floor(vWorldPos.xyz*1000
.0), i),16));
        visibility -= 0.2*(1.0-
        texture(shadowMap,
        vec3(ShadowCoord.xy +
        poissonDisk[index]/1000.0,
```



Christoph Prinz, Alexandra Geier

```
        (ShadowCoord.z-bias)/ShadowCoord.w) ));
}
```

Quelle: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/

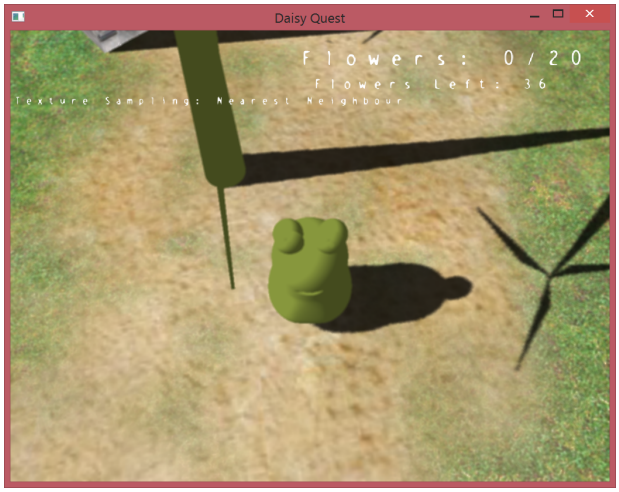
Probleme beim Shadow Mapping

Es stellte sich heraus, dass die Orthogonalmatrix beim Rendern der Shadowmap maßgeblich für die Qualität des Schattens ist. Beim ersten Bild handelt es sich um eine Orthogonalmatrix, die die ganze Map unseres Spiels umfasst. Da die einzelnen Objekte (zB feinen Rotorblätter des Windrades) sehr filigran im Verhältnis zur gesamten Welt sind, sind diese feinen Details bei unserer gewählten Methode nur sehr schwer erkennbar. Zu dünne Modelle führen dazu, dass Löcher im Schatten entstehen.

Im zweiten Bild hingegen wurde die Orthogonalmatrix viel kleiner definiert. So wird nicht mehr die gesamte Szene umfasst. Jedoch wirkt sich dies positiv auf den Schatten aus: Feine Details sind dann auch im Schatten erkennbar.

Tutorials & Quellen:

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/
http://ogldev.atspace.co.uk/www/tutorial23/tutorial23.html



Depth of Field (1.5)

Beim Depth of Field erfolgt das Rendering auch in 2 Schritten. Zuerst wird die Szene in eine Tiefentextur gerrendert. Hier kommen Frame Buffer Objects zum Einsatz. Nachdem die Tiefeninformation der Objekte in der Szene vorhanden sind, wird die Szene - in Abhängigkeit von der Tiefe - geblurt. Hier kommt ein spezieller Gaussian Blur zum Einsatz.

Quelle: http://www.pasteall.org/10779

Mit der F10-Taste kann das Depth of Field - Rendering ein- bzw- ausgeschaltet werden. Erkennen kann man den Effekt beispielsweise am Zaun in der Ferne. Ist DoF aktiviert, so ist dieser leicht verschwommen. Auch die Gegner sind in der Ferne mehr verschwommen, als wenn sie nahe an Daisy herankommen.

Normal Mapping (1.5)

Normalmapping wurde als dritter Effekt, auf dem Haus implementiert. Dazu wurde zusätzlich zur Farbtextur des Hauses mit Hilfe von CrazyBump (http://www.crazybump.com) eine Normalmap für das Haus erzeugt, die als Basis für die Generierung der Texturierung des Hauses dienen.
Es gibt einen eigenen Normalmap-Shader, der sich darum kümmert die Berechnung der Struktur zu übernehmen. Dabei spielt der Einfallswinkel des Lichtes und die Berechnung der Tangenten eine besondere Rolle.

Quelle: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

Animated Objects

Als animiertes Objekt haben wir ein Windrad eingefügt. Dabei rotiert der Hauptrotor im Kreis. Zusätzlich zu dieser Rotation rotieren dann noch die drei kleinen Rotorblätter um die eigene Achse. Die 3 kleinen Rotorblätter und der Hauptrotor sind eigene Meshes, die unabhängig geladen werden jedoch in Abhängigkeit animiert sind.

Frustum Culling

Beim View Frustum Culling haben wir uns an die Geometrischen Lösungsweg von Lighthouse3D [http://www.lighthouse3d.com/tutorials/view-frustum-culling/index/] gehalten. Dabei werden die für die Kamera und die für die Perspektive spezifizierten Parameter verwenden, um die 6 Planes für das View Frustum Culling zu berechnen. Um zu wissen, ob ein Model gezeichnet werden soll, muss überprüft werden, ob sich das Objekt ganz oder auch nur teilweise innerhalb dieser 6 Flächen befindet. Ist das der Fall, wird das Model gerendert. Befindet sich das Modell auch nur außerhalb einer dieser Flächen, wir das Modell nicht gerendert.
Um zu Überprüfen, ob sich ein Objekt innerhalb des Frustums befindet, wird jedes Modell durch eine Kugel angenähert, deren Mittelpunkt sich aus der Modelmatrix berechnet und deren Umfang abhängig von der Objektgröße ist.
Mit Hilfe der Taste F8 kann das View Frustum Culling ein und ausgeschalten werden.

Transparency

Die Implementierung von Transparenz kann bei unseren Blumen beobachtet werden. Mit Hilfe der F9-Taste kann dies aktiviert und deaktiviert werden.

Experimenting with OpenGL

Vertex-Array-Objects
Vertex-Array-Objects (oder kurz VAOs) kommen beim Laden und Rendern aller Objekte zum Einsatz. Grundsätzlich werden alle mittels Assimp geladenen Objekte an ein eigenes VAO gebunden. Der Boden (die Heightmap) wird an ein eigenes VAO gebunden. Beim Rendern wird immer das entsprechende VAO zuerst aktiviert.

Frame-Buffer-Objects
Wir haben in unserem Spiel Frame-Buffer-Objects verwendet zum einen um die Shadow Map mit den Tiefeninformationen für das Shadow Maping zu erzeugen.
Darüberhinaus haben wir auch ein FBO für das Depth-Of-Field, an das eine Color-Texture und eine Depth-Texture gebunden sind.

Mip Mapping
Wir haben MipMapping mit folgenden drei Modi implementiert: “Off”, “Linear”, “Nearest Neighbour”. Sichtbar ist der Wechsel dieser Modi nur am Boden. Mittels der Taste F5 kann zwischen diesen Modi gewechselt werden.

Texture Sampling
Auch beim Texture Sampling gibt es zwei verschiedene Modi (Nearest Neighbour und Bilinear) zwischen denen mittels der F4 Taste gewechselt werden kann.

Heightmap

Der Untergrund unserer Spielewelt wird über eine Heightmap erzeugt. Dabei werden die Höheninformationen in einem Bitmap-Bild gespeichert. Diese Höheninformationen werden dann mittels Bullet in Mesh umgewandelt, sodass sich Daisy frei auf der entsprechenden Höhe bewegen kann.

Die Physik & Collision Detection

Die physikalischen Gesetzmäßigkeiten werden über die Bullet Physx Library implementiert. Hier findet auch eine Überprüfung der der Kollisionen (Collision Detection) statt. Alle unsere Objekte haben eine kubische Form als “rigid Body”.

Für die Anzeige der einzelnen Bodies (rund um unsere Objekte) könnden diese mit der Taste B gerendert werden. Achtung: Dieser Modus ist nicht sehr performant implementiert, da die Shape des Untergrundes sehr aufwändig ist.

Das Schießen
Das Abfeuern von Wurfgeschoßen zum Töten der Gegner wird auch über Bullet realisert. Dabei wird ein neues Bullet-Objekt erzeugt und dies in Blickrichtung beschleunigt. Die Berechnung der Flugbahn übernimmt dann Bullet. Geschossen werden kann mit Hilfe der rechten Maustaste.

Controls

Folgende Controls und Steuermöglichkeiten gibt es:

W oder OBEN	Daisy bewegt sich nach vorne
A oder LINKS	Daisy bewegt sich nach links
S oder UNTEN	Daisy bewegt sich nach hinten
D oder RECHTS	Daisy bewegt sich nach rechts
F1	-
F2	Frametime in Millisekunden anzeigen
F3	Wire Frame off /on DOF STOP :
F4	Texture-Sampling-Quality kann zwischen Nearest Neighbour und Bilinear gewechselt werden. Auch hier wird eine Information am linken oberen Rand ausgegeben, welcher Mode aktuelle gewählt.
F5	Hier kann die Mip-Mapping-Quality eingestellt werden. Diese Funktion ist nur auf der Heightmap sichtbar. Es kann dabei zwischen den Optionen “Off”, “Nearest Neighbour” und “Linear” gewählt werden.
F6	Die Ziegen (also die Gegner im Spiel) werden gestoppt. Es werden auch keine neuen Ziegen zum Spiel hinzugefügt.
F7	Durch Drücken dieser Taste wird unsere direktionale Lichtquelle rotiert, um zu veranschaulichen, dass sich auch der Schatten mitverändert.
F8	Frustum Culling ein- beziehungsweise ausschalten. Standardmäßig ist das Frustum Culling eingeschalten. Nur wenn es ausgeschalten

	ist, wird am linken oberen Rand eine Information ausgegeben, dass es nicht eingeschalten ist.
F9	Transparenz wird ein- und ausgeschalten. Diese Funktion merkt man an den Blumen.
F10	Mittels der Taste F10 kann der Depth-Of-Field Effekt an und ausgeschalten werden. Wird aufgerufen, um den WireFrame-Modus korrekt abzubilden.
B	Bullet-Bodies rendern (Achtung Performance-Verlust!)
Enter	Enter startet das Spiel neu. Dies kann aber nur erfolgen, wenn das aktuelle Spiel beendet ist - also gewonnen oder verloren wurde.
Escape	Mittels Escape kann das Spiel zu jedem beliebigen Zeitpunkt beendet werden.

Das Sichtfeld wird über die Mausbewegungen gesteuert. Um die Ziegen abzuschießen, verwendet man die rechte Maustaste.

Libraries & eingesetzte Technologien

Bullet
http://www.bulletphysics.org

Assimp Model Loading
http://www.mbsoftworks.sk/index.php?page=tutorials&series=1&tutorial=23

Rendering von 2D-Text
http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-11-2d-text/

Simple basic Lightning
http://www.mbsoftworks.sk/index.php?page=tutorials&series=1&tutorial=11

OpenGL in General
http://opengl-tutorial.org/