Prince of Balls
Abgabe 3

CG2LU, SS 2008
Radax Ingo, 033 532, 0500848, ingo.radax@gmx.net
Bressler Michael, 033 532, 0425576, michael.bressler@gmx.net

Gameplay

Man spielt eine Kugel (den Kugel-Prinzen) und muss an das andere Ende der Welt gelangen wo die Kugel-Prinzessin auf einen wartet. Erreicht man diese so gewinnt man das Spiel.

Als Hindernis gibt es in der Welt verschiedene Räume zu überwinden in denen unterschiedliche Aufgaben auf einen warten (zB man muss über Röhren fahren ohne runter zu fallen, etc). Die meisten Aufgaben beschränken sich darauf dass man den Prinzen steuern muss. Vereinzelt kommen noch „Schlüssel suchen“ und „Schalter betätigen“ Aufgaben dazu, welche aber mit einer einzigen Taste durchführbar sind.

Effekte

Es wurden folgende Effekte programmiert:

- Shadow Maps
- Quad-Tree
- projizierte Texturen + planare Spiegelungen

Shadow Maps

Shadow Maps werden natürlich für Schatten eingesetzt. Bei der implementierung haben wir uns an das zweite Repetitorum und den dortigen Folien gehalten. Wir rendern die Szene von der Kamera aus in eine Textur (mit einem FBO), bauen die Matrix für den Lookup auf und führen dann in den Shadern den Schatten-Lookup aus.

Zu bemerken ist dass wir die Schatten nur in der Nähe der Prinz-Kugel berechnen (dazu richten wir die Kamera der Lichtquelle immer auf den Prinzen aus.9, weiter entfernte Gebiete sind schattenlos. Das fällt allerdings nicht besonders auf da die Welt in kleinere Gebiete geteilt ist und die Schatten meistens das ganze Gebiet abdecken.

Der Code für das Shadow-Mapping findet sich in den Klassen Lighting und Game, und natürlich in unseren Shadern.

Quad-Tree

Wir teilen unser Terrain in einen Quad-Tree auf und verwenden diesen dann zum cullen. Insgesamt haben wir 256 Zellen (16x16), die allesamt Achs-parallel sind und sich nicht überlappen.

Durch den Quad-Tree sind die Zellen jeweils in 4er Blöcke gefasst, (und diese wiederum in 4er Blöcke, etc, bishin zur Wurzel des Baumes). Beim cullen fangen wir an der Wurzel an und testen ob der aktuelle Block in Frustum liegt und falls ja führen wir den Test mit den Sub-Blöcken fort, bis hin zu den einzelnen Zellen.

Es wurden keine externe Quellen für den Quad-Tree verwendet, wir haben nach Erinnerung programmiert.

Der Code für den Quad-Tree findet sich in den Klassen Terrain und TerrainCell. Terrain ist die Wurzel des Quad-Trees.

Entities können auch von einer Zelle in einer andere Wechseln falls sie sich bewegen. Dazu überprüft jede Terrrain-Zelle im Update Schritt für jedes Entity ob es noch in der Zelle ist. Falls nicht gibt die Zelle der Wurzel des Quad-Trees den Befehl das Entity neu zu positionieren, diese schickt dann das Entity an alle untergeordneten Zellen, und diejenige in welche das Entity gehört nimmt es dann auf. Genauergesagt muss das Entity nicht selbst positioniert werden, sondern nur eine Referenz auf das Entity, sämtliche Entities werden nämlich in einer eigenen Liste (Klasse Entities), und die Zellen brauchen nur die Namen ihrer Entities.

projizierte Texturen + planare Spiegelungen (aka „Wasser Effekt“)

Um die Spiegelungen und Lichtbrechungen zu simulieren ist es notwendig, die Szene einmal an der Y-Achse gespiegelt zu rendern, wobei die Geometrie unterhalb der Wasseroberfläche weggeclippt wird (nach dem spiegeln darüber), und einmal die Szene normal zu rendern, wobei die Geometrie oberhalb der Wasseroberfläche geclippt wird. Beide Rendervorgänge werden mittels FrameBuffer in eine Textur gespeichert.
Um das eigentliche Wasser zu zeichnen ist es nun notwendig, die Texturen auf das Wasser zu mappen. Dies geschieht mit einem Shader, der anhand einer Normal Map und einer Fresnel Formel die
Texturen leicht verzerrt.
Schließlich müssen die gerenderten Texturen noch richtig auf das Wasser gemappt werden. Das funktioniert mittels projektivem Mapping recht einfach, die Texturkoordinaten werden anhand des Eye Spaces der Kamera von OpenGL erstellt.

Der Code für diesen Effekt findet sich in den Klassen Game und Water, und im water.shader file.

Wir haben uns bei diesem Effekt an folgender Quelle orientiert:
http://www.gametutorials.com/Articles/RealisticWater.pdf

Animierte Objekte

Als animierte Objekte kommen lediglich unsere Kugeln vor. Die Kugel bestehen im wesentlichen aus 2 Teilen: dem Kugelkörper und einer Deko. Der Kugelkörper rollt in der Gegend herum wie es im passt. Die Deko folgt dem Körper, rollt aber nicht sonder dreht sich nur (Y-Achse, im Prinzip so wie ein Kopf).

Frustum culling

Zum frustum culling verwenden wir hauptsächlich unser Terrain (siehe Effekt Quad-Tree). Sobald eine Terrain-Zelle erkannt wurde die nicht weggecullt worden ist, werden sämtliche Entities die dieser Zelle zugewiesen wurden nochmals separat getestet und gegebenenfalls gecullt.

Für das Culling verwenden wir eine eigene Frustum Klasse in Kombination mit einer Bounding Klasse (wir verwenden Bounding-Spheres zum cullen).

Orientiert haben wir uns bei der Implementierung an folgender Quelle.

http://www.lighthouse3d.com/opengl/viewfrustum/

Den Effekt von unserem Culling kann man bei den Debug-Ausgaben gut sehen (Taste F2). Hier wird angezeigt wie viele Obejte (besser gesagt wie viele verschiedene Modelle und Instanzen dieser Modelle) gerade gerendert werden. Bewegt man sich so sieht man dass diese Zahlen rauf und runter gehen.

Transparenz

Wir verwenden an 2 Stellen Transparenz. Zuerst mal gibt es ein paar Objekte in der Welt (Zäune, Netz vom Fussballtor) bei denen wir eine Textur verwenden die teilweise komplett durchsichtig ist. Dadurch sehen wir hindurch. Die transparenten Objekte werden separat gerendert (damit die sichtbarkeit stimmt), wir verwenden jedoch nur eine einfache Sortierung dieser Objekte die für unser Spiel ausreicht).

Außerdem verwenden wir Transparenz für die Skybox. Wir haben 2 Skyboxen (eine für Tag, eine für Nacht) und beim Übergang von Tag und Nacht blenden wir nach und nach über, so dass wir einen nahtlosen übergang haben.

Besonderheiten

Es gibt im Spiel Tag und Nacht. Die Lichtquelle dreht sich um die Welt und ist einmal die Sonne und ein anderes mal der Mond, das sieht man schön an den Schatten. Nachts wird alles etwas dunkler (und auch etwas bläulicher). Beim Übergang von Tag und Nacht würde es zu einem Springen von Schatten kommen wenn die Lichtquelle von Sonne auf Mond wechselt. Das haben wir so gelöst in dem wir ein paar Sekunden vor dem Springen den Diffusen anteil des Lichts langsam auf 0 setzten und sobald der Wechsel vollzogen ist wieder hinauf setzten. Im Spiel wirkt es dann so dass kurz alle Schatten verblassen, ganz verschwunden sind und dann wieder auftauchen.

Tools

Folgende Tools wurden im Spiel eingesetzt:

FBX ...	zum Laden der Modelle
Aigea Physx ...	für die Collisionserkennung und auch zum Bewegen der Kugeln in der Welt
Devil ...	zum Laden von Texturen
irrKlang ...	zum Abspielen von Sounds (Quelle: http://www.ambiera.com/irrklang/)

Modellierung

Modelliert wurde mit Maya und 3DS Max. Lediglich das Terrain wird zur Laufzeit aus einer Heightmap generiert.

Das Zusammenstellen der Objekte zur gesamten Welt erfolgte großteils händisch (siehe data/entities.ini) bzw ein Teil der Welt wird aus einem Bild generiert (siehe data/city.png, die einzelnen Farben legen Mauern, etc in der Welt fest).

Steuerung

Wir verwenden klassisch **ASDW** um den Prinzen zu Steuern und die **Maus** um die Kamera zu drehen/neigen. Zusätzlich gibt es noch die Taste **E** welche für generelle Interaktionen verwendet wird (ist man in der nähe von einem Objekt mit dem man interagieren kann erscheint ein Pfeil über dem Objekt, dann einfach **E** drücken).

Taste **P** schaltet in die Pause und mit **Esc** kann man das Programm beenden (einmal Esc drücken und es gibt ne Nachfrage mit Pfeiltasten 'Ja' auswählen und Enter drücken zum entgültigen beenden)

F1 bis F9 sind belegt wie gefordert, bei **F1** gibt’s in der Hilfe nochmals die Steuerung kurz erklärt.

Es gibt noch ein paar „Non-Game“ Sachen die wir drinnen gelassen haben, da es sich gut zum Testen eignet.

Mit **F11** wechselt man vom Spiel in den Freiflugmodus. Dabei ist die Kamera nicht mehr an den Prinzen gebunden und frei steuerbar, damit kann man sich schön die ganze Spielwelt ansehen. Drückt man wieder auf F11 springt die Kamera zurück auf den Prinzen. Zu bemerken sei dass einige Sachen in der Welt (zB Schatten) nur funktionieren wenn der Prinz in der Nähe ist, will man also die Welt erkunden und sich alles ansehen sollte man den Prinzen immer wieder mitnehmen (siehe nächster Absatz).

Mit der **rechten Maustaste** kann man den Prinzen teleportieren (insofern man im Freiflugmodus ist). Der Prinz wird dabei von seiner alten Position auf die Position unterhalb der Kamera (da ist dann eine rote Wire-Sphere) gesetzt. Damit kann man leicht einzelne Räume in der Welt auslassen und in die nächsten gelangen.

Hat man das Spiel gewonnen gibt es eine Gewonnen-Nachricht die mitten im Bild hängt. Will man diese weg haben drückt man auf **F12**.

Walkthrough

Um das Spiel zu gewinnen muss man einfach zur Prinzessin-Kugel gelangen (die ist beim Schloss, gegenüber vom eigenen Startplatz) und mit ihr reden (Prinz in die Nähe bringen und E drücken).

Um zur Prinzessin zu gelangen kann man jetzt einfach den Freiflugmodus benutzen (siehe Steuerung) oder aber alle einzelnen Räume des Spiels durchspielen. Wir beschreiben nicht wie alle Räume funktionieren, das gilt es für den schlauen Tutor selbst raus zu finden (für den nicht ganz so schlauen Tutor gibt’s den Freiflugmodus).