Time Travel Dragon (TTD) - Submission 2

(01427555) Walcher Felix
(01427350) Pauschenwein Johannes

Anforderungen:

Gameplay:
Der Drache kann durch die Welt geflogen werden und “Feuerbälle” können abgefeuert werden. Man erhält Punkte, wenn man Gebäude mit Feuerbällen trifft (300 Punkte für einen getroffenen Turm, 100 Punkte für ein getroffenes Haus). Ab 1000 erhaltene Punkte erreicht man Level 2, der durch eine schnellere Fluggeschwindigkeit ausgezeichnet ist um den Schwierigkeitsgrad zu erhöhen. Bei einer Kollision des Drachens mit der Map oder Gebäuden verliert man eins von Anfangs drei Leben. Zusätzlich verliert man bei einer Kollision 50 Scorepunkte. Wenn alle Leben aufgebraucht sind ist das Spiel vorbei. Mit der Enter Taste kann das Spiel von neuem gestartet werden.

Steuerung:
- Mit der “0” Taste kann in den God Mode geschalten werden, dann kann der Drache mit W A S D gesteuert werden. Außerdem ist man unverwundbar.
- Mit der linken Maustaste kann ein Feuerball geschossen werden.
- Mit Mausbewegung kann man sich drehen um die Flugbahn des Drachens zu verändern.
- F2 aktiviert/deaktiviert den Frametime und Object-Counter.
- F3 aktiviert/deaktiviert Wire Frame
- F4 aktiviert/deaktiviert den HUD.
- 5 aktiviert/deaktiviert HDR
- F8 aktiviert/deaktiviert das Culling

Effekte:
- Cel shading + Countours (Backfaces): Cel Shading wurde im Modelfragmentshader implementiert. Dabei wird der “Specular” und “Diffuse” Lichtanteil auf 4 Farbwerte abgebildet. Um den Contour Effekt zu erzielen wurde vor dem Rendern des eigentlichen Objekts, das selbe Objekte skaliert und nur die Rückseiten des Objekts in schwarz gerendert.

- GPU-Particle System: Ein GPU Particle System wurde mithilfe eines Compute und Geometry Shaders implementiert um Feuer zu animieren. Dabei ist die implementierung an den Folien des Repetitoriums angelehnt. Das Partikelsystem wird beim Schießen eines Feuerballs sichtbar und bei brennenden Gebäuden.

- Watermash: Leider ist es uns nicht gelungen eine Videtextur zu verwenden. Dabei haben wir versucht mit hilfe der C Library FFMPEG ein MP4 Video zu laden und die einzelnen Frames in Texturen umzuwandeln. Das Laden des Videos ist uns gelungen und auch das Dekodieren. Jedoch sind wir an der Umwandlung der einzelnen Frames in Texturen gescheitert, da die Library eine sehr unverständliche/veraltete Dokumentation besitzt. Auch Tutorials im Internet waren nicht hilfreich, da sich die Library vor kurzem drastisch geändert haben muss, da die meisten verwendeten Methoden veraltet sind. Dadurch wurde alternativ ein Water Mesh implementiert, deren Implementierung sich an den Folien der Vorlesung anlehnt.

- Heads-UP-Display: mit Fadenkreuz und Score bzw Lebensanzeige mit hilfe der Library “Freetype”.

- HDR rendering + Tone mapping : Leider gab es Schwierigkeiten mit HDR in zusammenhang mit unserem Watermesh und Kamera. Es scheint, als würd die Renderreihenfolge nicht mehr stimmen. Mit 5 kann HDR ein und ausgeschaltet werden, allerdings empfehlen wir es ausgeschalten zu lassen.

- Physics Engine: Mithilfe von Nvidia Physx wurde die Collision Detection und das schießen des Feuerballs implementiert. Die Map wurde gecooked, allerdings scheint die Shape verdreht zu sein, da nur an manchen Stellen die Colision des Feuerballs richtig funktioniert. Die Collision mit der Map und dem Drachen funktioniert jedoch gut und die Collision mit dem Feuerball und den Gebäuden funktioniert auch.
- Light mapping: Mithilfe von Lightmapping wurde eine Textur für die Map erstellt.

Complex 3D Models:
Das Spiel verwendet komplexe schon fertig heruntergeladene .OBJ Objekte, die mithilfe von Assimp geladen werden (z.B die Map, der Drache, …)

Hierarchische Animation: Um den abgefeuerten Feuerball bewegt sich ein zweiter Feuerball kreisförmig um den Effekt noch zu verbessern.