TECHNISCHE UNIVERSITÄT WIEN
ÜBUNG COMPUTERGRAPHIK
186.831, SS 4.0 H (6.0 ECTS)

2. Abgabe
Spiel Dokumentation

David Fankhauser, 1025876
Katharina Ölsböck, 0926993

18. Juni 2013

1 BESCHREIBUNG DER ANFORDERUNGEN

1.1 FREI BEWEGLICHE KAMERA

Die Kamera wurde in einer Klasse 'Player' implementiert. Die Rotation wird durch Polling direkt in der Klasse abgelegt und bei jedem update-Zyklus zur Model-Matrix hinzugerechnet. Diese ist nach oben und unten auf der z-Achse limitiert, sodass man sich nicht überdrehen kann.
Die Translation bzw. die Bewegung der Kamera (Spieler) wird vorher in die Physik-Welt übertragen. Von dort aus wird sie dann im update-Zyklus von Player abgefragt. Somit ließ sich Gravity und Collision-Detection mit dem Boden und auch anderen Objekten implementieren.

1.2 BEWEGENDE OBJEKTE

- Aufzug: Der Aufzug befindet sich im ersten Level (von insgesamt vier). Sobald der Spieler den Weg vor dem Aufzug betritt, startet ein Event, wobei mehrere Gegner auf der Aufzugsplattform erscheinen. Sobald man alle Gegner runtergestoßen hat erscheint ein Diamant in der Mitte des Aufzuges. Sammelt man diesen ein fährt der Aufzug nach oben und der Spieler kann das nächste Level betreten.

- Gegner-Bewegung: Die Bewegung der Gegner verläuft über einen selbst entwickelten Pfadfindungs-Algorithmus in der Physik-Engine. Dabei wird per Ray-Tracing von unten nach oben ermittelt, ob sich dort ein Pfad befindet. In einer ersten Phase wird kreisförmig um den Gegner geschaut, wo sich Pfadstücke befinden. Es wird zufallsbedingt ein breites Stück ausgewählt, welches dann weiterverfolgt wird, bis der Pfad aufhört. Dabei wird in fixen Schritten überprüft, ob sich der berechnete Weg in der Mitte des Pfades befindet. Nachdem ein Wegpunkt berechnet wurde, bewegt sich der Gegner dorthin. Hat er sein Ziel erreicht, wird ein neuer Wegpunkt berechnet.
  So wird sichergestellt, dass sich egal wie sich die Position (durch Wegschleudern z.B.) eines Gegners ändert, sich dieser frei auf dem Pfad bewegen kann. So ist es auch möglich, dass, wenn ein Gegner mit dem Baseball Schläger weggeschleudert wird, dieser gegen den Weg prallen und in die entgegengesetzte Richtung fliegen kann. Es besteht zudem auch die Möglichkeit, dass er wieder auf dem Weg landet und weiterläuft.

- Gegner-Animationen (Vertex Skinning): Die Gegner wurden mittels Blender als Skeletal-Animation animiert und als Collada-File exportiert. Per Assimp werden die Gegner-Models geladen und die Bone-Struktur wird als Baum-Struktur abgespeichert. Bei einem update-Zyklus wird der aktuelle timestamp verwendet, um die Keyframes der einzelnen Bones in der Hierarchie zu interpolieren und damit die Transformations-Matrizen (Bone Matrix) rekursiv zu berechnen. Die fertigen Bone-Matrizen werden daraufhin dem Vertex-Shader übergeben, welcher anhand dieser und der 4 am stärksten Gewichteten Bones die fertige Animations-Matrix berechnet.
  Die Gegner besitzen zudem eine Zustands-Maschine, welche dafür zuständig ist, sie mit einer gewissen Wahrscheinlichkeit für eine gewisse Zeit anhalten, wieder weiterlaufen (z.B. Laufen beginnt, läuft, endet) und fallen zu lassen.

1.3 Texture Mapping

Die UV-Koordinaten stammen aus Blender. Um die Texturen zu laden wurde Devil verwendet. Die Graphiken selbst wurden zuvor mit Photoshop so bearbeitet, dass die Kanten gespiegelt sind und somit kein Eindruck einer Kante im Spiel entsteht.

1.4 Beleuchtung und Material

Es wird in den Levels eine Lichtquelle (Sonne) positioniert. Da lediglich die Parameter der Position eines exportieren Lichts stimmen, wurden Einstellungen wie z.B. die Helligkeit in die globalen Variablen der Game-Engine festgelegt.
Jedes Objekt im Spiel hat ein zugewiesenes Material (Die Textur wird in diesem Kontext auch als Material bezeichnet). So haben Gegner eine Farbe in Blender definiert und der Pfad hat zwei verschiedene Texturen definiert: Erde und Gras.
Mittels Normalen (welche in Blender zuvor smooth geshaded wurden) wird in verschiedenen Shadern für alle Objekte im Spiel Blinn-Phong-Shading angewandt. Das spekulare Licht wird allerdings nur beim Schläger berechnet, da es keinen glänzenden Boden gibt und glänzende Gegner optisch nicht gut gewirkt haben.

1.5 Implementierte Steuerung

- WASD/Pfeiltasten: Bewegen
- Maus: Rotieren
- Leertaste: Hüpfen
- Linke Maustaste: Schlagen
- F2: FPS-, View-Frustum-Culling-Anzeige ein/aus
- F3: Wireframe-Mode
- F8: View-Frustum-Culling ein/aus
- F9: Transparenz ein/aus (sichtbar beim Diamanten, der im ersten Level erscheint)
- F12: God-Mode: Gegner werden per Berührung mit dem Spieler weggeschleudert, Spieler kann höher bzw. auch in der Luft springen und ist schneller in der Luft.

1.6 3D Sounds

Die Sounds werden mittels FMOD geladen. Verändert sich die Position des Spielers, wird das System upgedated. Verändert sich die Position eines Gegners, erneuert sich dessen zugehörige Sounds um die neue Position. Zusätzlich wurde in FMOD ein Doppler-Effekt eingestellt, indem die Geschwindigkeit der Positionsänderungen gespeichert werden.

1.7 Winning Condition

Hat der Spieler alle 3 Level gemeistert kommt er in ein 4. Level, welches ihm durch eine Nachricht mitteilt, dass er gewonnen hat.

1.8 View-Frustum-Culling

Es wurde ein simples View-Frustum-Culling auf die Gegner angewandt. Jedem Gegner wurde ein Würfel als Bounding-Box zugeordnet. In jedem draw-Zyklus wird überprüft, ob sich die Bounding-Box im Frustum befindet. Dazu wird für jeden Eckpunkt festgestellt, auf welcher Seite der begrenzenden Ebenen des Frustums er sich befindet. Wenn zumindest ein Eckpunkt innerhalb des View-Frustums ist, so wird die Bounding-Box als innerhalb gewertet und der Gegner gezeichnet.

2 Effekte

2.1 GPU Vertex Skinning

siehe Gegner-Animationen

2.2 Shadow Mapping

Im Spiel werfen lediglich die Gegner Schatten auf den Pfad, da dies optisch am ansprechendsten erschien. Prinzipiell wurde bei der Implementierung so vorgegangen, wie im Repetitoriumsvortrag von Peter Houska erklärt. In einem ersten Rendering-Durchlauf wird die Shadow-Map (also die Szene von der Sicht der Lichtquelle aus, lediglich Gegner, reduzierte Shader) in ein FBO gezeichnet. Im zweiten Durchgang wird die Szene normal von Sicht der Kamera aus gerendert. Dem Shader, mit dem der Pfad gezeichnet wird, wird die Shadow-Map übergeben. Im Fragment Shader wird überprüft, ob der Tiefenwert des Fragments mit jenem übereinstimmt, der in der Shadow-Map abgespeichert wurde, und das Fragment andernfalls schattiert.

2.3 Variance Shadow Mapping

Zur Verbesserung der Schatten wurde zuätzlich Variance Shadow Mapping implementiert. Als Orientierung und Grundlage diente das Paper "Variance Shadow Maps" von Donnelly und Lauritzen und das Beispiel auf http://fabiensanglard.net/shadowmappingVSM.
Zusätzlich zum normalen Shadow-Mapping werden in der Shadow-Map nicht nur die Tiefenwerte, sondern auch ihre Quadrate gespeichert. Durch Mittelung der Werte erhält man daraus die ersten zwei Momente der entsprechenden Wahrscheinlichkeitsverteilung, realisiert wird das durch Blurren der Shadow-Map. Verwendet dazu wurde ein 2-Pass-Gauß-Filter. Zuerst wird in horizontaler Richtung gefiltert und dabei in ein weiteres FBO gezeichnet, anschließend in vertikaler Richtung gefiltert und wieder ins ursprüngliche FBO gerendert. Aus dieser neuen Shadow-Map wird nun für jedes Fragment eine Schranke für die Wahrscheinlichkeit berechnet, mit der es im Schatten liegt. Dadurch wird eine Abstufung der Schattenwerte möglich, ein Fragment kann nunmehr auch im Halbschatten liegen. Anstatt Schatten mit harten Grenzen erhält man nun weiche Schatten (soft shadows).

3 Zusammengefasste Features

- Collision-Detection: Gegner, Spieler, Boden, Aufzug
- Gegner runterwerfen mittels Raytracing
- Von Gegner weggeschleudert werden
- Path-Finding-Algorithmus für Gegner
- Zufälliges Verhalten der Gegner
- God Mode
- Texture Mapping
- Vertex Skinning

- 4 Levels
- Blinn Phong Shading
- Events
- Anisothropische Filterung
- Sounds (zb. Hit und Crit Sound bei Treffer mit Schläger)
- Shadow Mapping (mit Erweiterung Variance Shadow Mapping)
- View-Frustum-Culling angewandt auf Gegner
- Crosshair, welches rot aufleuchtet, falls ein Gegner getroffen würde

4 Verwendete Libraries

- Assimp: Laden von Modellen
- Bullet: Physik Engine
- FMOD: Sound Engine
- GLM: Mathematik Library
- Devil: Laden von Texturen
- GLEW
- GLFW