Kögler Alexander, Rauscher Nicole

Minion in Action

Computergraphik Übung SS 2016, TU-Wien

Inhalt

Die Geschichte… 3

Anforderungen 4

Grafikkarte 4

Steuerung 4

Gameplay 5

Effekte 5

Komplexe Objekte 10

Animierte Objekte 10

View-Frustum-Culling 10

Transparency 10

Experimente mit OpenGl 10

Softwarearchitektur 11

Bibliotheken 11

Import 11

Berechnung und Steuerung 11

Software 12

Die Geschichte…

…eines Minion auf abenteuerlicher Reise einen Schatz zu finden beinhaltet unser Spiel. Der Minion ist im ersten Level in einer offenen Welt, dabei darf sich die Person, die spielt mit der Steuerung vertraut machen. Der Minion sucht den Eingang zu einer Höhle in welcher ein Schatz versteckt ist. Es existieren Katzen, die der Minion mit dem Golfschläger attackieren kann und die bei Erfolg verschwinden. Ist die Katze länger als zwei Sekunden in der Nähe des Minions, verliert dieser ein Leben. Der Minion muss sich vor dem Betreten der Höhle stärken und muss mindestens 5 Bananen essen, bevor er sich in das Abenteuer wagt. Hat man zu wenig Bananen, macht einem der Minion mit Essensgeräuschen aufmerksam, dass er Hunger hat.

Sobald fünf Bananen verzehrt wurden, geht die Reise weiter nachdem der Eingang zur Höhle betreten wurde. Die Umgebung wird finster, ein unebenes Gemäuer umgibt den Minion, Licht ist kaum vorhanden, wenige Lampen erhellen einen Korridor. Der Minion sieht eine Information vor sich, die er aufsammelt. Es gibt eine Katze, die den Eingang zu einem Schatzversteck bewacht, erst wenn diese erfolgreich in die Flucht mit dem Golfschläger geschlagen wurde, kann der Eingang geöffnet werden. Aber nicht nur das, auch böse, dämonische, verzauberte Minions bewachen die Höhle. Sie können gefährlich werden, denn sobald der Minion mit einem kollidiert verliert dieser ein Leben. Mehr als drei Leben hat der Minion nicht zur Verfügung. Bemerkt ein böser Minion den Eindringling, so sorgt dieser dafür, dass das Licht verzaubert wird und sich gespenstische Atmosphäre breit macht.

Hat der Minion den Zugang zum Schatz freigelegt und diesen aufgesammelt ist das Abenteuer erfolgreich abgeschlossen.

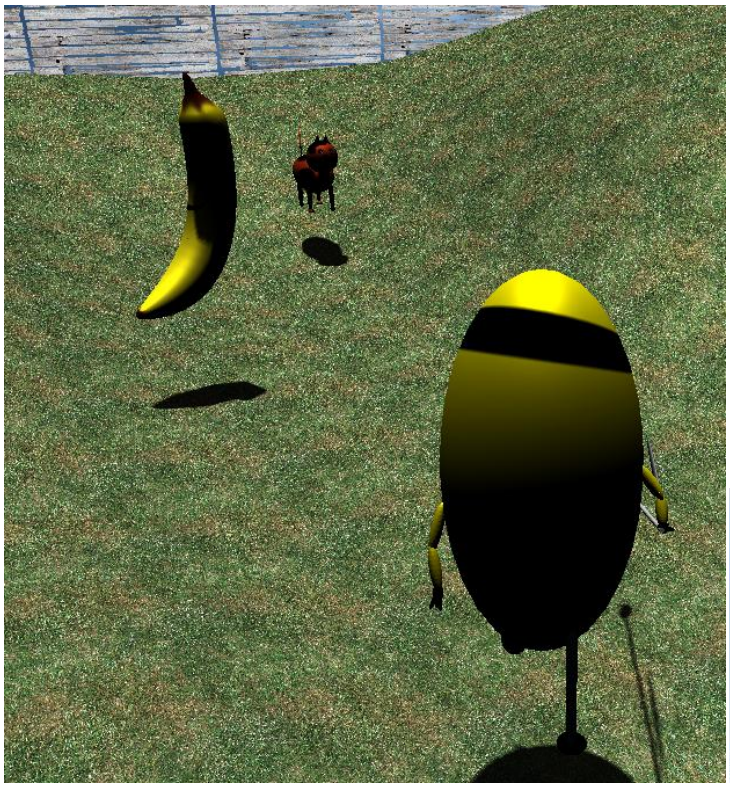
Anforderungen

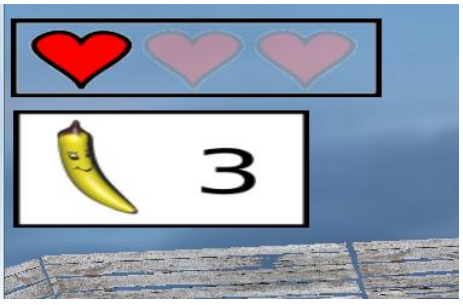
Grafikkarte

Zum Testen unseres Spiel bitte die AMD Grafikkarte verwenden!

Steuerung

Der Minion kann mittels Pfeiltasten in die entsprechende Richtung bewegt werden. Mit Hilfe der V-Taste kann man sich einen Überblick über die gesamte Spiellandschaft verschaffen, wobei 4 verschiedene Distanzen vorhanden sind, letztere beiden sind nur für Testzwecke eingebaut und sind dem Spielspaß nicht dienlich.





Die Anzahl verspeister Bananen wird bis zum mindestwert für Level1 angezeigt. Die Anzahl der Leben wird mit einem Herz Metaphorisch dargestellt.

Gameplay

Das Spiel beinhaltet die Möglichkeit tatsächliche 3D-Elemente abzubilden und mittels Physikengine, im konkreten Fall Nvidia-PhysX, ein realistisches Spielerlebnis zu bieten. In Level1 wird ein komplexes Gelände zur Verfügung gestellt, auf welchem Minion und Katzen sich fortbewegen können. Objekte deren Kollisionsmeshes im Rahmen der zur Verfügung stehenden Zeit angepasst werden konnten, befinden sich in Level2 (Lampe, Sessel, Säule, versperrende Wand). Damit ist sichergestellt, dass wir über ein Know-How, sowie über die technische Kompetenz einen Lösungsansatz für die Ansteuerung von Physik-Umgebungen zu entwickeln, verfügen.

Das Spiel verfügt über Sound. Im Level1 ist stets Vogelgezwitscher zu vernehmen, im Level2, sobald man von einem bösen Minion attackiert wird, dämonische, gruselige Geräusche. Beim Springen, Attackieren, Erfolgserlebnis, Gewinn, Verlust werden unterschiedliche Sounds eingespielt.

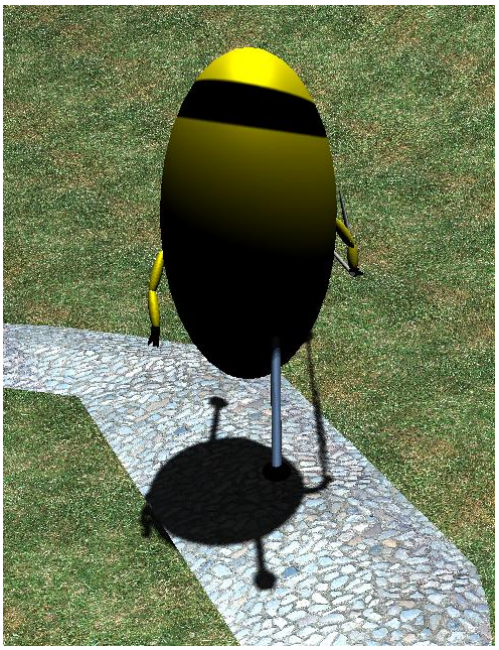
Aus nicht nachvollziehbaren Gründen bewegen sich die Katzen am Abgaberechner nicht, und das obwohl sie auf drei anderen Systemen funktionieren. Die bösen Minions in Level2 bewegen sich aber, trotzdem die Funktionalität gleichermaßen aufgebaut ist.

➔erfüllt

Effekte

Folgende Effekte sind im Spiel eingebaut

Shadow Maps mit PCF



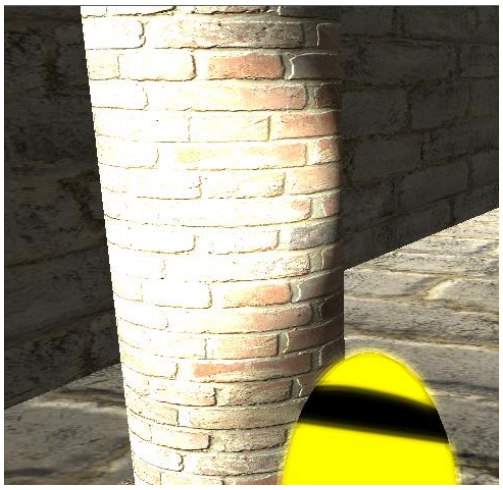
Im Level1 sind Schatten bei jedem gezeichneten Objekt zu sehen. Als Lichtquelle wurde ein directional light, dass sich fast senkrecht über dem Objekt befindet, als angemessen empfunden. Die Lichtstrahlen fallen allesamt parallel zueinander ein. Zuerst wurde ein Tiefenbild von der Szene aus Sicht der Lichtquelle gezeichnet. Diese Werte liegen zwischen 0 und 1 und werden in einer Textur abgespeichert. Danach kann die Berechnung der Schatten mit dem Blinn-Phong Beleuchtungsmodell durchgeführt werden. Zuerst muss die Position im light-space in Koordinaten des clipe-space umgerechnet werden. Dann wird die Methode PCF (percentage-closer filtering) angewendet, bei der die umliegenden 8 Werte plus den aktuellen Wert des Tiefenbilds zur Schattenberechnung herangezogen und ein Mittelwert dieser gebildet wird. Liegt ein Punkt außerhalb des eingestellten Sichtbereichs, wird ganz normal ohne Schatten gezeichnet. Zu der Schattenberechnung kommt dann ebenfalls die Berechnung des Blinn-Phong Beleuchtungsmodell und der Punkt bekommt die berechnete Farbe.

➔1.5 Punkte

Spotlights

Das Level zwei verfügt über die Möglichkeit 32 Spotlights gleichzeitig im Shader anzusteuern. Das Spotlight verfügt über eine Position und einen Normalvektor, der zugleich auch die Entfernung des zu beleuchtenden Abschnittes bestimmt. Der Radius um die Normale wird abhängig von der Entfernung des orthogonal zu beleuchtenden Punkt auf die Normale stehenden Distanz zur Position, in Relation zur maximalen Entfernung und Länge des Normalvektors vom Spotlight bestimmt. Der Lichtstrahl wird um den dadurch gebildeten Lichtkegel abgeschwächt, aber auch am Ende der Normale und damit Lichtreichweite ebenfalls. Im Gegensatz zu üblichen kürzeren Verarbeitungen mit Abstrahlwinkeln, wurde diese Methode, ohne vorliegende Informationen, aus dem Internet erarbeitet.

Es wurde auch auf Anordnung beim Abgabegespräch eingebaut, dass sich die Spotlights bewegen um zu beweisen, dass es sich um Spotlights handelt. Passend für die Hintergrundgeschichte in Level2 schien uns, dass die Spotlights, sobald man mit einem bösen Minion kollidiert sich wild durch die Gegend rotieren. Es ist ein von bösen Geistern und Dämonen befallener Korridor. Dabei wird die Position des Spotlights unverändert belassen, aber die Normale global rotiert.

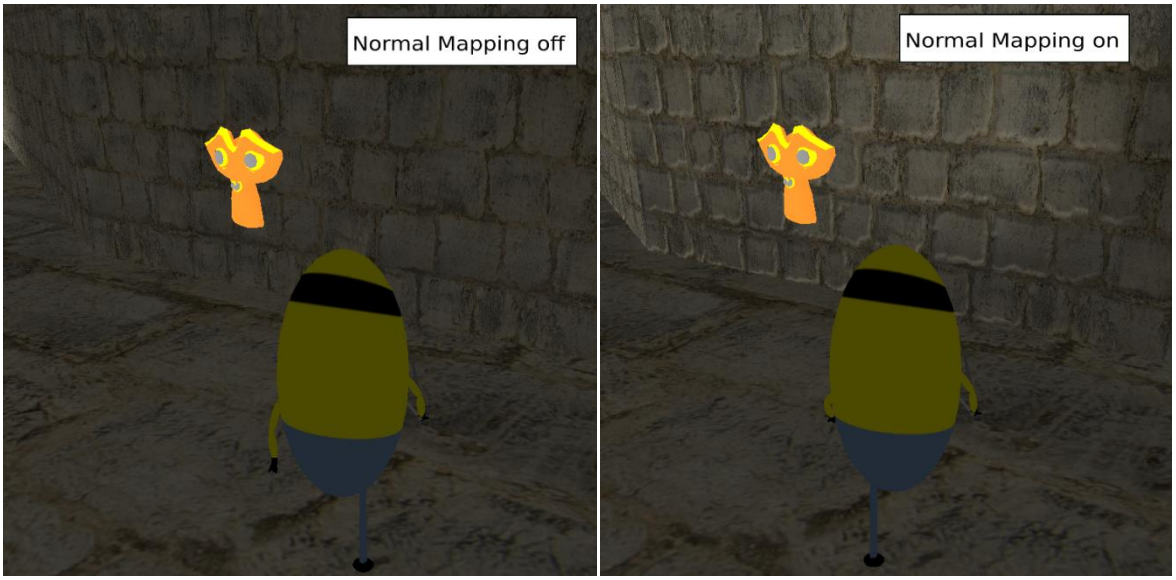


➔0.5 Punkte

Normal Mapping

Für die Berechnung der Normals werden die Tangenten und Bitangenten beim Auslesen der 3D-Daten mit Assimp, bei aktivierter Option für das Szeneobjekt, berechnet in Abhängigkeit mit den UV-Koordinaten. Eine Normal-Textur wird zusätzlich eingelesen, anhand derer der Normalvektor mit den Tangenten und Bitangenten im Shader transformiert wird.

Das Normalmapping ist sichtbar im Level2 an der Wand und dem Boden, sowie an den Säulen. Damit wurden auch komplexe Objekte im Spiel mit Normalmapping implementiert.



Mit den bewegenden Spotlights ist ebenfalls, wie beim Abgabegespräch gefordert, erkennbar, dass Normalmapping im Spiel eingebaut ist. Das Normalmapping kann mit F6 (de)aktiviert werden.

➔1 Punkt

Static LOD

Es gibt die Möglichkeit LOD für Szeneobjekte zur Verfügung zu stellen. Nähert man sich dem Objekt wird zur besseren Detailstufe gewechselt. Im zeitlichen Rahmen der LVA wurden ein Sessel und eine Stehlampe kreiert, die im Level2 platziert sind und von einer weniger auffallenden, wie beim Abgabegespräch gefordert, Statur auf ein mit mehr Details ausgestaltetes Objekt wechseln. Damit ist hinreichend, dass wir über das geforderte Know-How betreffend Static-LOD verfügen und auch verstanden haben wozu dies eingesetzt wird.

➔1 Punkt

Bloom

Die Implementation von Bloom/Glow beweisen folgende Objekte im Level2:

- Banane → Bloom, es wird ein größeres Banane Objekt in den temporären FBO gezeichnet
- BöserMinion/Minion → Glow, verfügen über eine Textur, die violette/gelbe Teile des Minions in den FBO zeichnen. Der Böse Minion verfügt auch über ein rotes Auge. Damit ist bewiesen, dass eine Glow-Textur zur Anwendung kommt, die Teile eines Objekts für Glow heranzieht und unterschiedliche Farben nutzt.
- Affenkopf →Teile des Affenkopfes erstrahlen in gelb, damit ist bewiesen, dass nur Teile eines Szeneobjektes mittel Glow-Textur implementiert wurden

Für das Bloom/Glow wird in zwei Texturen, die in TU0/1 abgelegt werden gezeichnet. Ein Durchlauf für einen horizontalen, als auch vertikalen Filter existiert im Shader. Im letzten Durchlauf wird das so produzierte gegaußte Bild auf die Szene mit einer Transparenz von 75% gelegt. Der Filter nimmt mit Abstand von 5 Pixeln zueinander Werte aus der Textur um einen besonders bemerkbaren Eindruck zu bieten, im Gegensatz zu Implementierungen aus dem Internet, bei denen Durchlauf eins und zwei in niedrig aufgelöste FBO gerendert werden.

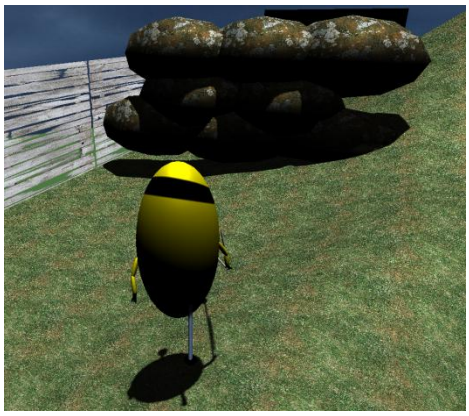
Bloom/Glow kann mittels F10 an und aus geschaltet werden.

→1 Punkt

Es sind mehr als die 4 geforderten Effektpunkte vorzuweisen.

→**<u>5 PUNKTE</u>**

Komplexe Objekte



Das Spiel beinhaltet mit Blender erstellte, komplexe Objekte. Es beinhaltet Steinblöcke, eine Säule, und alles andere an verschiedenen Formen, auch den Affenkopf aus Blender. Die Objekte werden im DAE-Format zur Verfügung gestellt, und mit Assimp ausgelesen. Beim Import wird automatisch Rücksicht auf Kollisions-, BoundingBox-, und Lodelemente sowie Hierarchien anhand des Präfix im Bezeichner genommen.

➔erfüllt

Animierte Objekte

Das Spiel verfügt über hierarchische Animationen, erkennbar am Minion beziehungsweise bösen Minion, dessen Beine und Arme sich bewegen. Szeneobjekte verfügen über Worldmatrix (Positionierung im Raum), jedes darin enthaltene Mesh verfügt über eine Modelmatrix (Positionierung innerhalb des Models). Daher ist es möglich das gesamte Szeneobjekt oder nur Teile davon individuell zu bewegen. Komplexe Drehbewegungen wie beim Minion werden mittels Kenntnis des Drehwinkels abgestimmt (Vor- und Rückbewegung)

➔erfüllt

View-Frustum-Culling

Es gibt im Spiel View-Frustum-Culling, das die Anzahl der gezeichneten Objekte begrenzt. Mittels BoundingBox und NP-Algorithmus wird entschieden, ob ein Objekt gezeichnet wird. Für die Ausgabe der Anzahl im letzten Durchlauf der Gameloop gezeichneten Objekte ist die O-Taste (nicht Zero) zu drücken.

➔erfüllt

Transparency

Das Spiel bietet Transparenz, welche mit F9 (de)aktiviert werden kann. Diese ist in Level1 am Zaun, der selbiges limitiert, erkennbar. Genauso wie bei der Anzeige der noch vorhandenen Leben in beiden Levels.

➔erfüllt

Experimente mit OpenGl

Die Engine des Spiels nutzt VBO, VAO, FBO für Szeneobjekte oder Effekte. MipMapping und Texture-Sampling-Quality sind einstellbar. Die Belegung der Funktionstasten entspricht der in den Anforderungen angegebenen. Eine Meldung über die aktivierte Option wird ausgegeben.

➔ erfüllt

Softwarearchitektur

Es gibt für Texturen einen Texturemanager, dessen belange sich darüber erstrecken Texturen nicht mehrmals in den Speicher zu laden und automatisch, je nach Nutzungshäufigkeit die Textureunits der Grafikkarte belegt. Weniger genutzte Texturen werden bei Bedarf ersetzt, sofern die Anzahl möglicher Textureunits begrenzt ist und mehr benötigt werden.

Für Sounddateien gibt es selbiges.

Der Shadermanager ist verantwortlich Shader nicht redundant zu laden, sondern diese je nach Anforderung über die Pfade der Shaderdateien zu cachen. Abhängig davon werden auch ohne Redundanz Shader-Programme aus einer Kombination unterschiedlicher Shader generiert. Der Shader wird beim Zeichnen nur dann gewechselt, sofern es vom zu zeichnenden Objekt eine Anforderung dafür gibt.

Das Sceneobjekt ist die Basisklasse für alle Szeneobjekte, in ihr finden die notwendigen OpenGL-Aufrufe sowie die verarbeitung und das Laden der 3D-Daten statt.

Die Kamera sorgt sich um die Steuerung des Hauptcharakters in Bezug auf die Ansicht. Auch aktualisiert die Kamera, bei Änderung von sich selbst oder einem Shaderwechsel, die Uniforms der Shader, insbesondere die Transformation in den Kameraraum.

Bibliotheken

Import

Zum Import der 3D Modelle wird die Assimp Bibliothek verwendet. Hierbei wird von einer unterstützten Datei jeweils das erste Mesh, dazugehörige Normals und UV-Koordinaten importiert. Das Modell muss auch darüber verfügen, da dies alles vorausgesetzt wird und somit Fehlerüberprüfungen und Sonderfälle vermieden werden.

http://assimp.sourceforge.net/

Das Laden der Texturen wird mit der FreeImage Bibliothek durchgeführt. Da erste Tests erfolgreich waren wurde diese Bibliothek vorerst beibehalten, wider erwarten gibt es allerdings Probleme mit den Farbkanälen der Grafikformate.

http://freeimage.sourceforge.net/

Dem Konstruktor eines SceneObjects wird der Pfad zu Model, Textur und TextureUnit übergeben. Dieser lädt automatisch die Daten mit Assimp und FreeImage in das Objekt.

Berechnung und Steuerung

Die GLM Bibliothek wird für die Grafischen Berechnungen, insbesondere Bewegungen von SceneObjekten, aber auch für die Kamera mit der Dritten-Person-Perspektive, genutzt.

http://glm.g-truc.net/0.9.5/index.html

GLFW wird für die Steuerung bzw. Benutzereingabe mittels Tastatur und Maus, darüber hinaus zur GUI Ansteuerung des Betriebssystems genutzt.

http://www.glfw.org/

Software

Benutzt wurden unter anderem:

- InkScape: Zwischensequenzen, Infotexte
- PaintShop Pro Version 5.01: Bildbearbeitung, Transparenz
- CodeXL: Debuggen der OpenGl Stati
- Nvidia PhysX Debugger: Debugging der PhysX Engine
- Blender: Erstellung der 3D-Modelle
- VisualStudio 2012: Programmierung in C++
- open3mod: Anzeige und Analyse der mit Blender erstellten 3D Modelle, besonders bezüglich Transformationen und um Fehler bei den Transformationsmatrizen zu finden
- Matlab R2011b: Analyse und Erarbeitung verschiedener Schritte bei Bloom/Glow Effekt
- GIT, Tortoise GIT, GIT-Bash: als Versionierungssystem
- FRAPS: analyse der Framerate
- Camtasia: Aufzeichnung eines Spiele-Videos
- Nero 7 WaveEditor: Schnitt von Soundfiles
- IrfanView: Analyse und Debugging im Falle falsch zugeordneter Farbkanäle bei Nutzung von Transparenz
- Windows 7, sowie Windows 10 als Betriebssystem auf denen Programmiert und das Spiel getestet wurde.