Rota 1.0 – Abgabe 2

Computergraphik 2 LU

19.06.2009
Georg Zankl
Timo Kropp

Kurzbeschreibung

In Rota 1.0 spielt man einen Reifen aus der 3rd Person Perspektive. Der Spieler kann mit diesem frei im Level herumfahren, Objekte verschieben und über Hindernisse hinwegspringen. Die Gravitation ändert sich, sobald er mit dem Reifen auf eine angrenzende Fläche mit einem Winkel kleiner als 45 Grad fährt und wirkt nur auf den Reifen und Objekte die kurz zuvor berührt wurden.

Ziel ist es, Fässer zu bestimmten Zielen (transparente Fässr) zu verschieben. Und anschließend den geheimen, runden Raum zu finden. Dort ist das 6. Fass versteckt. Dies ist in möglichst kurzer Zeit zu schaffen. Dabei gilt es Hindernisse zu überwinden ohne einen Schwindelanfall zu bekommen (noch mehr Loopings!). Die Lösung des Levels steht am Ende des Dokuments.

Anforderungen

Nichttriviale Objekte:

Der Reifen selbst, sowie viele andere Objekte im Level sind nicht konvex und besitzen Materialeigenschaften, die dem Objekt einen gewissen Glanz (Specular-Anteil) verschaffen. Einige dieser Objekte besitzen sogar nicht konvexe Kollision Geometrie.

Animierte Objekte:

Bei Rota 1.0 gibt es nur hierarchisch animierte Objekte, wie z.B. den Ventilator (in der Röhre), den man beliebig verschieben kann, wobei sich der Rotor ständig dreht.

View Frustum Culling:

Für das View Frustum Culling wurde die Collision Detection von PhysX genutzt, indem ein Actor mit einem View Frustum Shape erstellt und transformiert wurde. Für jede Bounding Sphere, die in das View Frustum ein- oder wieder austritt wird damit eine Callback Methode aufgerufen, die die sichtbarkeit der Objekte umstellt. Dies gilt auch für Lichtquellen, sodass die Beleuchtung, aber auch die Schatten einer Lichtquelle nur berechnet werden, wenn ihr Einflussbereich das View Frustum schneidet.

Alpha Blending:

Alle Ziele, an die die Objekte verschoben werden sollen, sind transparent, um zu signalisieren „Hier sollte ein Objekt stehen“. Implementiert wurde die Sortierung indem sich

bei einem Update die geblendeten Objekte selbst zu einer Liste hinzufügen, die direkt beim Einfügen nach der quadratischen Distanz zur Kamera sortiert.

OpenGL Experimente:
Die Modi (siehe Steuerung - Funktionstasten) wirken, sobald aktiviert, sofort auf alle betroffenen Objekte, d.h. werden Display Lists eingeschalten, so werden alle Objekte per Display List gerendert. Ist der Render Modus auf Vertex Arrays oder Vertex Buffer Objects eingestellt, wird die Display List Einstellung allerdings ignoriert, da diese nur im Immediate Mode Sinn macht. Die Vertex Daten selbst werden in keinem Fall geändert.

Steuerung

Der Reifen wird mit der klassischen W, A, S, D Steuerung gespielt. W beschleunig und S bremst den Reifen. A und D drehen den Reifen nach links bzw. nach rechts. Mit der Leertaste kann man hüpfen.

Die Kamera wird mit der Maus frei um den Reifen herum bewegt. Dabei ist zu beachten, dass sobald der Spieler den Reifen in Bewegung setzt, die Kamera so schwenkt, dass sie den Reifen von hinten verfolgt. Dies lässt sich per Druck auf die linke Shift Taste umschalten. Durch gedrückt halten der linken Maustaste lässt sich dieser Modus kurzfristig umschalten (sobald man die Taste loslässt, wird der Modus wieder zurückgestellt)

Funktionstasten:

F2	Framerateanzeige [an/aus]
F3	Wire Frame Modus [an/aus]
F4	Texturqualität [Nearest Neighbor/Bilinear]
F5	MipMapping Qualität [Aus/Nearest Neighbor/Linear]
F6	Render Modus [Immediate/Vertex Arrays/VBOs]
F7	Display Lists [an/aus]
F8	View Frustum Culling [an/aus]
F9	Alpha Blending [an/aus]
F10	Debug Renderer [an/aus]
F11	Shader [an/aus]
F12	Volume Shadows [an/aus]

Features

Über die Rampen können die Wände befahren werden. Röhren können von innen und außen befahren werden. Außerdem können Abgründe übersprungen werden. Es gibt Objekte (Holzkisten,..) mit den der Spieler interagieren kann. Diese können umgeworfen und verschoben werden. Außerdem kann der Spieler auf einige Objekte springen. Wenn der Spieler es in die große Röhre geschafft hat, dann kann er sich bei der Öffnung an jeder

beliebigen Stelle herausfallen lassen und landet entsprechend auf der darunter liegenden Ebene.

Effekte

Shadow Volumes:

Als Grundlage wurde folgendes Howto verwendet:
http://http.developer.nvidia.com/GPUGems/gpugems_ch09.html
Wir haben die z-Fail Methode für die CPU implementiert. Für jede Lichtquelle gibt es ein Renderpass und innerhalb diesem wird für jedes Objekt, was einen Schatten werfen soll, die Sillhouette berechnet und die Vertices, die auf dieser liegen werden von der Lichtquelle weg extruhiert. Die daraus entstehende Side, zusammen mit dem LightCape und dem DarkCape, wird in den Stencilbuffer gerendert.
Wenn eine Lichtquelle und ihr Einflussbereich außerhalb des View Frustums liegen, wird diese gecullt und die Lichtquelle wird bei dem Renderpass ausgelassen. Außerdem wird das Shadow Volume (Side, Lightcap, Darkcap) nur dann neu berechnet, wenn sich das Objekt bewegt, d.h. wenn PhysX meldet, dass es nicht im „Sleep“ Zustand ist.

Per Pixel Lighting:
Phong Shading wurde nach einem Tutorial (http://www.lighthouse3d.com/opengl/glsl/index.php?dirlightpix) implementiert und auf mehrere an- und abschaltbare Lichtquellen sowie das Phong Beleuchtungsmodell erweitert.

Verwendete Libraries

GLEW - http://glew.sourceforge.net/
GLFW - http://glfw.sourceforge.net/
Nvidia PhysX - http://www.nvidia.com/object/nvidia_physx.html
Für die Simulation der Physik im Spiel inkl. Collision Detection
FBX - http://www.autodesk.com/fbx
Zum Laden der Modelle aus 3D Studio Max, exportiert als FBX File.

Modelle

Wir haben alle Modelle selber mit 3D Studio Max erstellt und diese als FBX abgespeichert. Dazu haben wir einen eigenen FBX Importer programmiert, um die Daten aus dem FBX-File zu importieren. Dafür wurde das FBX SDK verwendet.

Kurzbeschreibung der Implementierung

Wir haben mehrere Hierarchische Strukturen für das Rendern und das Update implementiert. Es gibt Strukturen für die Meshes, für Transparente Objekte und für das HUD.

Die Klasse GeMesh beinhaltet die Datenstrukturen mit den Geometriedaten. Diese werden mittels der Klasse FBXConverter aus den FBX Files herausgelesen und erzeugt. Des Weiteren enthält GeMesh ein Objekt GeMeshRenderer, der das Rendern im entsprechenden Modus übernimmt (Immediate Mode, Vertex Arrays, Vertex Buffer Objects). GeMesh kann mehrere Kinder vom selben Typ haben, um eine Hierarchische Animation, oder Mesh-Teile mit unterschiedlichen Materialien bzw. Texturen zu ermöglichen.
Zusätzlich gibt es noch die, von GeMesh abgeleitete, Klasse GeActor, welche einen NxActor aus dem PhysX SDK nutzt um die physikalische Simulation (insbesondere auch die Collision Detection) zu ermöglichen.

Die Rotation des Reifens, wenn sich die Gravitation ändert, haben wir mittels der Rotationsmatrix von PhysX gelöst. Die Kamera hingegen arbeitet mit Quaternionen. Bei jeder Änderung werden diese Werte abhängig vom Normalvektoren der Collision, dem Gravitationsvektor, und dem Richtungsvektor der Fahrtrichtung, geändert.

Die Keyboard Abfrage für die Steuerung wurde mittels Polling realisiert, Leertaste und Funktionstasten sollen aber nur einmal kurz gedrückt werden und nutzen daher eine Callback Methode. Dies befindet sich in der Klasse Input.

Um eine Frameraten-Unabhängige Spiellogik zu erhalten, wird die Zeit mit einem Performance Counter gemessen und alle 20ms ein Updates ausgeführt.

Walkthrough

Im ersten großen Raum müssen zwei Fässer jeweils an die Decke und an die Wand geschoben werden. Dort befinden sich die transparenten Zielfässer. Sobald sich ein Fass in Bewegung befindet, ändert sich für dieses die Gravitation mit, somit kann der Spieler ein Fass an eine Wand schieben. Nun versuchen über eine Rampe auf die Wand zu fahren und von dort das Fass an der Wand weiter entlang schieben.

Im nächsten Schritt muss das Loch im Boden im ersten Raum gefunden werden. Einen Weg zu dieser Wand gibt es mittels den Rampen. Wenn man sich durch das Loch fallen lässt, landet man in einer Röhre. Dort ist ein zweites Fass und das transparente Fass befindet sich im hinteren, dunkeln Bereich der Röhre. Wenn man nun an das andere Ende der Röhre fährt, kann man aus ihr heraus fahren und landet im 3. Level Abschnitt. Dort sind noch zwei weitere Fässer zu platzieren. Über die Rampen, bzw mittels sich an einem Abgrund fallen lassen, kann jede Wand erreicht werden.

Um das letzte Fass im Runden Raum zu platzieren, ist ein Weg über die dicke Säule zu nehmen. Wenn man einmal auf ihr drauf ist, kann man durch den kleinen Durchgang in der

Ecke des Raumes zum Runden Raum gelangen. Wenn man dort das letzte Fass richtig platziert, hat man gewonnen und die Zeit wird gestoppt.

Um das Spiel neu zu starten, muss es über ESC zuerst verlassen werden.