Blocks

Christoph Winklhofer
MatNr. 0426461 Kennz. 033532 christoph.winklhofer@gmx.at

Thomas Weber
MatNr. 0526341 Kennz. 033532 weber.t@gmx.at

20. Juni 2007

1 Zum Spiel

Blocks ist ein 3D Plattformer mit Puzzle-Elementen. Der Spieler steuert eine Spielfigur durch eine Welt, die vollständig aus Holzbausteinen besteht. Dabei hat er die Aufgabe Murmeln aufzusammeln und den Ausgang des jeweiligen Levels zu erreichen.

Die Spielfigur kann frei durch die Welt laufen und springen. Außerdem kann sie Blöcke bis zu einer gewissen Größe schieben und ziehen.

Die Blöcke, aus denen die Welt besteht, folgen einer eigenen, vereinfachten, Physik. Im Ruhezustand sind sie immer an einem Raster ausgerichtet. Ausserdem lassen sie sich nur entlang den Achsen verschieben und sind nicht drehbar.

1.1 Steuerung

Die Spielfigur wird mit einem Game-Controller gesteuert. Idealerweise wird dabei ein XBox 360 Gamepad verwendet.

- Mit dem linken Control-Stick wird die Spielfigur bewegt. Die Richtung ist abhängig von der Kamera. So läuft die Spielfigur z.B. immer in Blickrichtung der Kamera, wenn der Spieler den Stick vorwärts bewegt.
- Mit dem rechten Control-Stick wird die Kamera gesteuert. Mit vorwärts/rückwärts kann man die Distanz der Kamera zum Spieler verändern. Mit links/rechts rotiert man die Kamera entlang der Y-Achse um den Spieler.
- Beim Betätigen von A springt die Spielfigur.
- Mit B werden Blöcke verschoben.
- Mit START kann das Spiel unterbrochen werden. Hier kann man auch das aktuelle Level neu starten, und zum Startbildschirm zurückkehren.

1.1.1 Springen

Wenn die Spielfigur Bodenkontakt hat und A betätigt wird springt sie nach oben. Dabei behält sie die Geschwindigkeit, die sie bereits hat. Die Spielfigur ist in der Luft weiterhin, jedoch eingeschränkter, mit dem linken Control-Stick steuerbar.

Neben dem normalen Sprung beherrscht die Spielfigur noch einen „Wall-Jump“. Dabei kann sich die Spielfigur, wenn sie mit genug Anlauf gegen eine Wand springt, durch rechtzeitiges Betätigen von A erneut von der Wand abstoßen. Zwischen zwei Wänden lässt sich das unbeschränkt oft wiederholen. So können sonst schwer erreichbare Bereiche erreicht werden.

1.1.2 Schieben

Wenn die Spielfigur direkt vor einem Block steht und B betätigt wird, greift sie den Block, solange B gehalten wird. Wenn man nun den linken Control-Stick in die entsprechende Richtung drückt, und der Block nicht zu groß ist, wird ihn die Spielfigur in diese Richtung schieben oder ziehen. Wenn der Spieler B wieder loslässt, lässt auch die Spielfigur den Block los.

1.1.3 Tastatur-Befehle

- Mit *ESC* kann man das Spiel beenden.
- Mit *F1* kann man sich die Szene in Wireframe-Mode ansehen.
- Mit *F5* kann man ein Level neu starten.

1.1.4 Einstellungen

Einstellungen zu Blocks werden in einer XML-Datei hinterlegt, das als Kommandozeilenargument übergeben werden kann. Folgende Einstellungen sind möglich

- input: Eingabegerät. Controller oder Maus/Tastatursteeuerung
- screen: Gibt die Fenstergröße an und ob das Spiel im Vollbildmodus gestartet wird. Default ist 1024x768 in Vollbild.
- level: Der Name des Levels, in dem man starten möchte. Optionen sind:
  - „Tutorial“
  - „Start“
  - „Level One“
  - „Level Two“
  - „Level Three“
  - „Level Four“
  - „Castle“
  - „Credits“
  - „Random“(Zufallsgeneriertes Höhenfeld)

  Default ist „Start“.

2 Umsetzung

2.1 Nichttriviale Objekte

Die Spielfigur ist ein komplexes nichtkonvexes Objekt, das, genauso wie ein Großteil der anderen Objekte im Spiel, gekrümmte Oberflächen besitzt.

Wir laden animierte 3D Modelle mit OgreXML.

2.2 Animierte Objekte

Die Spielfigur haben wir in Blender animiert und in das OgreXML Format exportiert. Als Animationstechnik haben wir Skeletal Animation [6] mit gewichteten Bones verwendet. Aus Performancegründen berechnen wir die Vertexpositionen der einzelnen Frames beim Laden der Spielfigur vor.

2.3 Beschleunigung der Sichtbarkeitsberechnung

Wir benutzen den Octree, um ganze Teilbäume auszuschließen. Dafür testen wir die Bounding Sphere eines Knotens auf Kollision mit dem Frustum der Kamera. Das Frustum wird durch seine Grenzebenen definiert.

2.4 Transparenz Effekte

Wir verwenden Transparenz zur Erstellung des animierten Himmels. Dabei werden Objekte wie Wolken, der Horizont oder die Sonne ebenenweise aufgetragen, und können frei über den Bildschirm bewegt werden.

3 Spezialeffekte

3.1 Light Space Perspective Shadow Maps

Zur Schatten-Berechnung benutzen wir Shadow-Maps. Da aber Levels relativ groß werden können, und unsere Lichtquelle global ist, kam es bei ersten Versuchen zu stark merkbarem Perspective Aliasing der Schatten.
Aus diesem Grund haben wir Light Space Perspective Shadow Maps implementiert. [5] [4]
Zur Bestimmung der sichtbaren Bereiches haben wir Clipping verwendet. Dabei clippen wir einerseits eine polygonisierte Version des Kamera-Frustums gegen die Bounding Box der Szene, und eine polygonisierte Version der Bounding box gegen das Kamera-Frustum. Die Eckpunkte der übrig gebliebenen Polygone sind die Extrema des sichtbaren Bereichs. Mit ihnen werden dann in weiterer Folge die clipping planes der zusätzlichen Transformation bestimmt.

3.2 Per Pixel Lighting mit Deferred Shading

Wir verwenden Deferred Shading zur pixelweisen Beleuchtung unserer Szene. [2]
Wir haben uns aus zwei Gründen für diese Methode entschieden. Einerseits, kann man so verhindern , dass für verdeckte Pixel die Helligkeit berechnet wird, ohne dass man einen weiteren vollständigen Pass benötigt. Andererseits ist so

Geometrie und Beleuchtung klar getrennt, wodurch unser Programm modularer und wartbarer wird.

3.3 Depth of Field

Wir implementieren Depth of Field als Postprocessing-Effekt mit einem variablen Filter-Kernel. In der ersten Implementierung haben wir die gerenderte Scene, nach Scheuermann [3] , mit einem vorgeblurrten Bild interpoliert. Die Tiefenunschärfe war in diesem Fall aber fast schon zu stark. Deshalb, und aus Performancegründen, haben wir die Tiefenunschärfe nur mit einem 8 Sample Filter-Kernel angewendet. [1]
Der Fokus wird dabei immer auf die Spielfigur gerichtet.

3.4 Octree

Sämtlich Blöcke sind in einem Octree verwaltet, in dem sie sich dynamisch bewegen können. Diese Struktur verwenden wir auch für View Frustum-Culling.

4 Externe Bibliotheken

- glfw: OpenGL Framework für die Erzeugung eines Anzeigefensters und Abfragen der Eingaben.
  http://glfw.sourceforge.net/
- GLee: Automatisches Laden von openGL-Extensions.
  http://elf-stone.com/glee.php
- IlmIf: Lesen von OpenEXR (HDR Dateiformat) Dateien
  http://www.openexr.com
- Imath: Mathematik Bibliothek
  http://www.openexr.com
- tinyxml: Parsen von XML Dateien
  http://www.grinninglizard.com/tinyxml
- Ogg Vorbis: Audio File Decoding.
  http://www.vorbis.org
- OpenAL: Sound Ausgabe.
  http://www.openal.org

5 Benutzte Tools

Zum Modellieren der Spielfigur wurde Blender benutzt.
Die Himmels-Texturen und die Textur auf der Spielfigur wurden per Hand mit Ölkreide und Wasserfarben gemacht, eingescannt und mit GIMP bearbeitet.
Das Original der Holz-Textur stammt von Wiki-Commons. Für unsere Zwecke haben wir die Orientierung angepasst, die Textur zugeschnitten und den Helligkeits-Gradienten des Original-Bildes ausgeglichen.

6 Kommentare zu den einzelnen Levels

6.1 Tutorial

Das Tutorial-Level ist als Einführung zu den Gameplay-Mechanismen des Spieles gedacht, und kann auch übersprungen werden.

6.2 Start

Hier muss man alle 3 Murmeln einsammeln und in das Ziel auf der Plattform gegenüber laufen.

6.3 Level One

Hier muss man Blöcke verschieben, um an die Murmeln und das Ziel zu gelangen.

6.4 Level Two

Hier muss man in das Ziel gelangen, bevor der herunterfahrende Block den Weg versperrt.

6.5 Level Three

Hier muss man Blöcke verschieben, um an die Murmeln und das Ziel zu gelangen.

6.6 Level Four

Hier muss man die drei Murmeln links und rechts der Bahn des roten Blocks aufsammeln, bevor man in das Ziel gelangen kann. Man muss aufpassen, dass man den Transport-Block dabei nicht verpasst.

6.7 SPOILER: Walkthrough durch Burg-Level

Das hier ist eine Schritt für Schritt Anweisung zum Lösen des großen Burg-Levels.

Erklärung: Richtungsangaben wie Links, Rechts, Vorne, hinten etc. sind alle von der Blickrichtung der Spielfigur zu Beginn des Levels aus angegeben.

- Laufe durch das Burgtor.
- Hier sollte Dir der eine Block in der Wand des großen Turmes auffallen, der etwas kleiner ist als die anderen und einen anderen Farbton hat. Ziehe ihn heraus, und hole Dir die blaue Murmel, die im Raum dahinter versteckt ist.
- Schiebe den Block jetzt nach rechts in die Lücke bei der Treppe.
- Statt die Stiege hinauf zu laufen, laufe nach vorne in den Garten hinter dem großen Turm.
- Entspann' Dich ein wenig beim Springbrunnen. ;)

- Bei der Wand hinter der linken Treppe im Garten ist ein Spalt, der zu einem Gang unter der Burgmauer führt. Folge dem bis zum Ende.
- Hier sind vier kleine Blöcke und ein großer Träger, der den kleinen Turm in der hinteren Burgmauer stützt.
- Schiebe den Träger von der Plattform. Der Turm wird nun herunterstürzen und auf den kleinen Blöcken aufliegen, ohne Dich zu erschlagen.
- Jetzt ist es Zeit auf die Burgmauer zu gehen. Lauf dazu die Treppe hinauf, die Du im dritten Schritt vervollständigt hast.
- Laufe jetzt durch den Großen Turm durch und auf den Eckturm hinten links.
- Spring jetzt vom Turm auf die hintere Burgmauer und laufe über den, nun um 4 Blöcke verkleinerten Turm, auf den Eckturm hinten rechts. Hier findest Du die grüne Murmel.
- Lauf jetzt wieder auf den Eckturm hinten links und spring auf die bewegliche Plattform, die sich zwischen den beiden Ecktürmen links hin und her bewegt.
- Von der Plattform kommst Du zum vorderen linken Eckturm, auf dem Du einen roten Block findest.
- Dieser Block ist ein Schalter. Wenn Du auf ihn springst, werden die die Zinnen der vorderen Burgmauer nach oben schweben und einen Pfad zum Eckturm rechts vorne formen.
- Tu' das und hol Dir die rote Murmel.
- Jetzt hast Du alle drei Murmeln, die Du benötigst, um das Level abzuschließen.
- Laufen nun in den großen Turm zum Thron und springe auf ihn.
- Der Thron wird nun mit Dir nach oben fahren.
- Hier musst Du jetzt von Plattform zu Plattform springen, bis Du ganz oben am Turm angelangt bist.

6.8 Random

Dieses Level wird zufallsgeneriert, und besteht aus 64x64 unterschiedlich hohen Blöcken, auf denen zufällig Murmeln und Ziele verteilt sind. Von hier aus kann man in viele unterschiedliche Levels gelangen. Dieses Level ist eher eine Zugabe. Es wird nicht garantiert, dass es mehr als 20 FPS hat.

Literatur

[1] Natalya Tatarchuk Guennadi Riguer and John R. Isidoro. Real-time depth of field simulation. In Wolfgang Engel, editor, *ShaderX2: Shader Programming Tips and Tricks with DirectX 9.0*. Wordware, Plano, Texas, 2003.

[2] Shawn Hargreaves. Deferred shading. Vortragsfolien GDC 2004.

[3] Thorsten Scheuermann. Advanced depth of field. Vortragsfolien GDC 2004.

[4] Michael Wimmer and Daniel Scherzer. Robust shadow mapping with light space perspective shadow maps. In Wolfgang Engel, editor, *ShaderX 4 – Advanced Rendering Techniques*, volume 4 of *ShaderX*. Charles River Media, March 2006.

[5] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light space perspective shadow maps. In Alexander Keller and Henrik W. Jensen, editors, *Rendering Techniques 2004 (Proceedings Eurographics Symposium on Rendering)*, pages 143–151. Eurographics, Eurographics Association, June 2004.

[6] Ryan Woodland. Filling the gaps - advanced animation using stitching and skinning. In *Game Programming Gems 1*, volume 1 of *Game Programming Gems*. Charles River Media, August 2000.