

Alien Racer -2.Abgabe

entwickelt wird mit C# und XNA für die Xbox 360

Katharina Krösl	0325089	e932	e0325089@student.tuwien.ac.at
Iliyana Zagralova	0226860	e532	e0226860@student.tuwien.ac.at

Kurzbeschreibung der Implementierung

Das Spiel läuft auf Windows7 und auf der Xbox360. (Steuerung in Beiden Fällen mit Gamepad)

Gameplay

Gespielt werden kann im Singelplayer oder Multiplayer mit insgesamt max. 4 Spielern.

Die Spieler fahren auf einem Track um die Wette. Wer vom Track abkommt verliert an Geschwindigkeit. Durch das einsammeln von Itemboxen erhalten die Spieler nützliche Items um sich einen Vorteil im Rennen zu verschaffen. Die gesammelten Itemboxen werden nach einer gewissen Zeit neu gesetzt, damit genug Power-Ups für die nächste Runde da sind. Wer als erstes seine 3.Runde beendet gewinnt.

Nichttriviale Objekte

- Auto: Das Auto kann vom Spieler durch das gesamte Level gefahren werden (-> auf der Rennstrecke hat das Auto die höchste Geschwindigkeit, wenn man den „falschen Weg“ nimmt, wird sie verringert).

Animierte Objekte

- Skydome: Der Skydome rotiert ganz langsam um das vorbeiziehen von Wolken zu simulieren.
- Itemboxen: Die Itemboxen rotieren um ihre eigene Achse und können vom Spieler „eingesammelt“ werden. Durch einen Zufallsgenerator bekommt der Spieler eines von 3 Items:
 - Kanister mit Nitro: erhöht die eigene Geschwindigkeit.
 - Hühnerei: behindert als Spiegelei die Sicht der Mitspieler.
 - Seifenblase: schließt den zum Fahrer nächsten andern Spieler in einer Seifenblase ein und hindert ihn so für eine gewisse Zeit an der Weiterfahrt.
- Seifenblase („Bubble“): Ein in eine Seifenblase eingeschlossener Fahrer schwebt in der Blase in die Luft, schwankt hin und her und dreht sich einmal um 180 Grad, bevor die Blase zerplatzt und er wieder auf die Fahrbahn fällt.
- Ei: Das Spiegelei auf der Scheibe ist eine animierte Textur, die verzerrt und verschoben wird um so den Eindruck zu erzeugen das Ei rinnt nach unten weg.

Beschleunigung der Sichtbarkeitsberechnung

View-Frustum-Culling von Objekten und Terrain wurde mit Hilfe einer Quadtree-Datenstruktur implementiert.

Transparenz-Effekte

- Itemboxen, Ziellinie, Seifenblasen: Alle diese Objekte werden mit Alphablending gerendert.
- Terrain: Beim Multitexturing für da Terrain wird auch vom Alpha-Kanal Gebrauch gemacht.
- HUD: Das HUD wird mittels Alpha-Blending über die Szene geblendet
- Ei-Item: Das „weiche Ei auf der Scheibe“ wird erzeugt indem eine Spiegelei-Textur mittels Alphablending mit Hilfe eines SpriteBatchs über die Szene geblendet wird. Anfangs ist es noch undurchsichtig, wird aber immer transparenter, bis es schließlich verschwindet.

Entwurf der Datenstrukturen

Die 3D-Objekte wurden in Maya modelliert und werden aus .fbx-Files geladen und als Models verwendet.

Texturen sind als .jpg oder .png -Dateien in Verwendung.

Das Terrain ist als Heightmap realisiert.

Alle zu zeichnenden Objekte werden von der selbst erstellten Klasse „SceneModel“ abgeleitet, die sich wiederum von „Microsoft.Xna.Framework.[DrawableGameComponent](#)“ ableitet.

Für View Frustum Culling und das Anwenden von LOD am Terrain , sowie zur Einschränkung der Collision Detection -Abfragen wird eine Quadtree-Datenstruktur verwendet.

Entwurf eines Kameramodells

Eine vom Auto unabhängig bewegbare Kamera würde in unserem Spiel keinen Sinn machen. Deshalb ist die Kamera als Third-Person-Kamera implementiert. Sie befindet sich etwas hinter und über dem Auto und folgt diesem. Um die Übersicht für den Spieler zu verbessern, blickt die Kamera nicht direkt auf das Auto, sondern etwas darüber hinweg (10 Pixel höher). Außerdem bleibt die Kamera immer über dem Terrain um auch dann eine gute Sicht auf das Auto und die Strecke zu ermöglichen, wenn der Spieler in eine Schlucht fährt.

Implementierung von Texture Mapping und Material:

Jedes Objekt im Spiel hat Texturkoordinaten und wird im Shader mit der passenden Textur (als png oder jpg) korrekt texturiert. Für das Terrain wird Multitexturing (bestehend aus 4 Texturen) verwendet.

Darüber hinaus wird je nach Material des Objekts eine Variable an den Shader übergeben, die die Intensität des „specular light“ bestimmt um das Objekt mehr oder weniger glänzend bzw matt erscheinen zu lassen.

Beleuchtung

Das Level wird von der Sonne, als direktionale Lichtquelle, beleuchtet. Die Beleuchtung wird durch Per-Pixel-Lighting (mit Phong Beleuchtung) realisiert. Dies wird in einem Shader mit Shadow Maps umgesetzt.

Die Beleuchtung aller Objekte funktioniert korrekt, Schatten sind auf Grund der Größe der Shadowmap, die das gesamte Terrain umfasst, und der Bewegung der Autos nicht 100% zufriedenstellend. (siehe flackernder Schatten unter den fahrenden Autos)

Steuerung

Zur Steuerung wird ein Xbox-GamePad verwendet. Gespielt wird in der Third-Person-Perspektive. Die Kamera folgt dem Spieler-Fahrzeug in einer festgelegten Höhe und Entfernung und ist nicht vom Spieler selbst steuerbar.

Steuerung mit GamePad

Right Trigger:	Gas
Left Trigger:	Rückwärtsgang
Left Thumb Stick:	Steuert die Richtung
A-Button:	Item aktivieren

Tastenbelegung am Keyboard

- F2: Framerateanzeige ein/aus
- F3: Wire Frame ein/aus
- F4: Texturqualität verändern
- F5: Gamepad / Tastatursteuerung umschalten
- F6: Bloom Effekt ein/aus
- F7: LOD ein/aus/off (teilweise fehlerhaft)
- F8: Viewfrustum-Culling ein/aus (teilweise fehlerhaft)
- F9: Transparenz ein/aus
- F10: Zeichnen der Bounding Volumes ein/aus
- F11: Zeichnen vom Quadtree ein/aus (zeichnet 1-3 Level)
- F: Fullscreen Modus ein/aus (vor dem Spielstart)

Steuerung mit Keyboard (nur zu Debug-Zwecken)

Steuerung mit Pfeiltasten, kein Rückwärtsgang

Effekte

Beleuchtung/Schatten:

- Shadow Maps:
realisiert mit einer direktionalen Lichtquelle (Sonne)
Es wird eine große Shadowmap verwendet, die das gesamte Terrain umfasst.

Visibility/Level of Detail:

- Quadtree:
Das Terrain wird in eine Quadtree Struktur unterteilt, durch die View Frustum Culling realisiert wird.

Screen Space-Effekte:

- Bloom:
Helle Regionen des Bildes überstrahlen dunklere Bereiche. Das Glühen wird über ein eigenes Rendertarget gelöst welches weichgezeichnet wird. Dieser Effekt verschlechtert allerdings die Framrate
- Motion Blur:
implementiert nach dem Motion Blur -Ansatz von Shawn Hargreaves:
Durch das wechseln der Rendertargets kann die gerade gerenderte Szene als Textur ausgelsen werden und mit Hilfe eines SpriteBatchs im nächsten Renderdurchgang mit einer gewissen Transparenz über die Szene geblendet werden. Bereits nach ein paar Renderdurchgängen hat man dadurch einen schönen Motion Blur – Effekt.

Sonstige Features

Multiplayer

- Auswahl von Playermodus (bis zu 4 Playern)
- Zwei verschiedene Fahrer-Auto-Modelle mit mehreren Texturen
- 2 Levels (Alien Desert, Green Planet). Die Levels werden in einer XML- Datei definiert, wodurch neue Levels immer wieder eingeschaltet werden können.
- Collision Detection zwischen den Fahrern und Itemboxen, sowie das Collisions-Verhalten wurde als minimalistische Physics-Engine selbst implementiert. Zur optimalen Durchführung der Intersection –Tests verschieden Bounding Volumes verwendet: für die nicht bewegten Objekte globale Bounding Spheres und AABB, und für die Autos eine hierarchische Kombination aus Globale Bounding Sphere, OBB und lokale Bounding Spheres um die Reifen.

HUD

- Aktuelle Runde und Rundenzeit
- Platzierung für jedes Level
- Endplatzierung und Gewinnerliste

- FPS

Antialiasing

- durch Multisampling

Levels of Detail für Terrain-Textur:

- Durch die Verwendung von MipMaps werden die Texturen je nach Entfernung zur Kamera in unterschiedlicher Detailstufe gerendert (besonders gut zu sehen am Terrain)

Textur Filter:

- Linear MipFilter
- Anisotropic MinFilter
- Anisotropic MagFilter

Sound

- Hintergrundmusik: ein Song wird in einer Schleife abgespielt

Quellen

msdn Library: How To: Implement Shadow Mapping

<http://msdn.microsoft.com/en-us/library/bb975671.aspx>

Gamasutra: Implementing Lighting Models With HLSL

http://www.gamasutra.com/features/20030418/engel_02.shtml

Shawn Hargreaves Blog: Motion Blur

<http://blogs.msdn.com/b/shawnhar/archive/2007/08/21/motion-blur.aspx>

GameDev.Net: Car-Car Collision

http://www.gamedev.net/community/forums/topic.asp?topic_id=193500#1213168

XNA Creators: Collision Series 5

<http://creators.xna.com/de-DE/sample/collision3dheightmapnormals>

XNA Creators: QuadTerrain

<http://forums.xna.com/forums/p/15397/80600.aspx#80600>

Euclideanspace.Com: Collision Detection und Physics

<http://www.euclideanspace.com/threed/games/examples/cars/collisions/index.htm>

Buch: Konerow Jens, 2009, Spieleentwicklung mit dem Microsoft XNA Framework, Frankfurt am Main, entwickler.press

Buch: Jones, Wendy. 2008, Beginning DirectX 10 Game Programming, Boston, Course Technology

Buch: Labao Alexandro, 2009, Beginning XNA 3.0 Game Programming: From Novice To Professional, Apress

Buch: Milligton Ian, 2007, Game Physics Engine Development, Morgan Kaufmann