MA48 Unlimited - Documentation

Implementation

1. Scene Objects

Es gibt Folgende Arten von Szenen-Objekte:

- Mesh
- Movable_Mesh
- Linked_Mesh
- Camera
- Light

a. Static Objects (Mesh)

Das ist die Hauptklasse für alle anderen Objekten. (ja, ich weiß, die Bennenung ist nicht sehr aussagekräftig/Korrekt in allen Fällen :)). Direkte Instanzen von Mesh sind die Müllobjekte in Weltall und die Basisstation.

Alle Instanzen von Mesh Besitzen einen Pointer auf eine Instanz eines *Collisions Objects (Bullet-Framework).* Dadurch wird ein Tracking für Kollisionen und die Kollision selbst ermöglicht.

b. Moving Objects (Movable_Mesh)

Die Objekte die durch's All fliegen sind Instanzen von Movabe_Mesh, die von Mesh (und implizit Scene_Object) erben. Die Bewegung passiert in der update-Funktion durch die Modifikation der Modelmatrix. Die entscheidung, wie die Modelmatrix geändert wird, hängt von den User-Inputs (Keyboard-Keys und Mouse). Dafür werden Berechnungen gemacht, die eine Flüssige und weiche Bewegung des Objekts ermöglichen (z.B bleibt das Objekt nicht gleich stehen, nach dem man die Bewegungs-Tasten los lässt, sondern es wird weiter bewegt während die "Kraft" nach und nach lässt bis das Objekt dann still steht. In der aktuellen Implementierung gibt es nur ein Hauptobjekt das eine Instanz von Movabe_Mesh ist. Das ist das Schiff mit den man herum fliegt (Akteur).

c. Linked Objekts od Childs(Linked_Mesh)

Das sind Objekte die an andere Szenen-Objekte verknüpft sind. Bei jedes Update übernehmen sie die Position des Parents mit ein bestimmtes Positions-Offset. Weiter können diese auf beliebige rotationen eingestellt werden. In die aktuelle Implementierung befinden sich vier solche Objekte: 1. Satelitenschüssel (verknüpft and Basisstation mit eine Rotation um die eigene y-Achse), 2. Station_Propeller (verknüpft and Basisstation mit eine Rotation um die eigene y-Achse), 3. Orientierungs-Pfeil (verknüpft an das Akteur-Schiff), 4. Scheinwerfer(verknüpft an das Akteur-Schiff).

d. Cameras

Kameras werden auch als Szenen Objekte behandelt, sie haben eine Modelmatrix (für position und Orientierung) jedoch haben diese keinen Körper und werden auch nicht gezeichnet.

- Es gibt zwei Kameras im Spiel; eine Positioniert hinter den Raumschiff, eine im Raumschiff hinter der Windschutzscheibe.
- Durch drücken und halten des Rechts-Clicks, wird die aktuelle Kamera vom Schiff ab gekoppelt und mit den W,A,S,D tasten zusammen mit der Maus Bewegung (für die Neigung) wird die Kamera in Raum bewegt.

e. Lights

Die Lichtquellen sind ebenso Szenen-Objekte die nicht gezeichnet werden. Es Gibt zwei arten von Lichtquellen.

i. Pointlight
ii. Spotlight

Es gibt in der Szene drei Lichtquellen:

- Eine Punkt-Lichtquelle, das ist die Sonne.
- Zwei Rampenlichter, das sind die Scheinwerfer des Raumschiffes. Diese können von User ein- und ausgeschaltet werden (mit der L-Taste). Sie werden von einem Linked_Mesh repräsentiert (siehe c. 4.)

2. Texture Mapping

Texturen werden über den sogenannten TextureHelper erstellt, der Bilder mithilfe der FreeImage-Library lädt.
Die Texturkoordinaten werden gemeinsam mit dem Mesh Object (.obj Files) importiert. Diese werden mit Hilfe des Assimp-Importers ausgelesen und einen Databuffer gespeichert.

3. Lighting and Materials

Falls ein Objekt keine Textur hat, wird die Materialfarbe aus dem .obj-File ausgelesen und für den Shader gespeichert. Das Specularfaktor wird auch ausgelesen und zur renderzeit an Shader für Specular Lighting weitergegeben.
Die Szene wird mit hilfe des BlinnPhong-Shader beleuchtet. (wegen Lichtquellen siehe auch 1. e.)

4. Controls

a. Keyboard

Das Schiff wird über die Pfeiltasten nach oben, nach unten, nach rechts, links, w,a,s,d gesteuert. Im Spiel wird die Keyboardsteuerung (in Help Menu) so erklärt:

Press W/S Keys to move Forward/Backwards
Press A/D Keys to move To Left/To Right
Press UP-DOWN Keys to pitch
Press LEFT-RIGHT Keys to roll
Press 'R' to go home (reset to start)

Andere Steuerungen der Szene sind folgende:

Press 'F1' for Help / Pause
Press 'F3' for Line/Wire-Frame
Press 'F4' for SpaceWaste InfoScreen
Press 'F8' for Viewfrustum-Culling on/off
Press 'F9' for Transparency on/off (ship-windshield)
Press 'F10' for SkyBox animation on/off
Press 'B' for Bloom-Effect
Press 'F' for turning Lensflare Effect ON/OF
Press 'L' for turning spotlights ON/OF
Press 'C' for changing Camera View

b. Mouse

Mit der Maus kontrolliert man die Orientierung des Schiffes (ähnliches Ergebnis wie die UP-DOWN, LEFT-RIGHT Keys). Klickt und hält man die linke Maustaste wird das Schiff um die eigene Y-achse rotiert, wenn man die Maus nach links und rechts bewegt.

Die freie Kamerafahrt wird mit den W,A,S,D erreicht, solange die rechte Maustaste gedrückt ist. Die Orientierung der Kamera ändert sich dabei durch die Mausbewegung.Wenn man die rechte Maustaste los lässt kehrt die Kamera wieder zum Raumschiff zurück.

Im Spiel wird die Maussteuerung (in Help Menu) so erklärt:
Right-Click and hold for free moving around with W;S;A;D!"
Move the mouse to change the Ship direction !"
Left-Click and hold for yaw-rotation with the mouse !"

5. Effects

Es wurden 3 Effekte implementiert:

- Spotlight
- Bloom
- Lensflare

Spotlight (Taste L)

Wird für die Scheinwerfer des Schiffes verwenden, wenn man sich an einem Objekt in der Szene annähert, merkt man leicht erhöhte beleuchtung des Objekt. Am großen Flächen wie Basisstation, sieht man leicht die runde fläche des Scheinwerfers.

Bloom (Taste B)

Beispiel: Wenn man mit der Kamera (und Raumschiff), in Richtung Sonne schaut, werden die Ränder des Schiffes leicht rötlich, das funktioniert selbstverständlich bei alle anderen Objekte.

Lensflare (Taste F)

Beispiel: Das Effekt ist leicht zu erkennen. Es tritt verstärkt bei helle Objekte die in der MITTE der Szene sich befinden. Das Effekt verwedentet (unter anderem) ein ScaleBiasShader und eine 1-Dimensionale Textur für die Linsenfarben.
Sowohl Bloom- als auch Lensflare-Effekt, werden bei niedrigen Auflösungen (mithilfe von Mipmaps) berechnet.

6. Basic Gameplay

Es werden im Raum zufällige Objekte an zufälligen Orten erstellt, diese werden von dem Schiff geladen. Beim Fliegen durch's All sinkt das Leben um eine bestimmte Menge. Man Startet mit einem Leben von 100 Punkte. Ist man bei 0 Lebenspunkte, stirbt man. Erreicht man die Basis noch bevor das Leben auf 0 Punkte sinkt, bekommt man wieder mindestens 100 Lebenspunkte. Beim Sammeln eines Abfallobjekts, steigt das Leben des Schiffes um eine bestimmte Menge.

"Features" of the game

- transparent Textures
- Automatic generation of waste, you never get the same waste :)
- Realistic movements of the ship (like flight simulation)
- multiple control possibilities through mouse and keyboard
- lensflare effect
- bloom effect
- Animated Skybox
- endless loop (for big highscores)
- ship-headlights
- background music (spacelike-Theme), and GameOver - music
- Fullscreen possibility
- Framerate independency
- Help Screen
- SpaceWaste - InfoScreen (Facts about waste in space)
- High precision collision detection through morecomplex collisionshapes
- Animated Objects
- better render-performance through Frustum Culling
- Orientation arrow that points to the base-station, to find the way back home
- 2 Camera Views

URLs

Der Blur Effekt (verwendet für bloom & lens flare) basiert auf den Vorschlag von nVidia:
http://http.developer.nvidia.com/GPUGems3/gpugems3_ch40.html
LensFlare Effekt implementiert aus:
http://john-chapman-graphics.blogspot.co.at/2013/02/pseudo-lens-flare.html
Spot-Light Effekt: http://www.mbsoftworks.sk/index.php?page=tutorials&series=1&tutorial=20

Skybox:
http://www.keithlantz.net/2011/10/rendering-a-skybox-using-a-cube-map-with-opengl-and-glsl/
Text-Anzeige:
http://zenpandainteractive.blogspot.co.at/2013/02/rendering-text-in-opengl-using-freetype.html (Code ist aber sehr fehlerbehaftet)
http://en.wikibooks.org/wiki/OpenGL_Programming/Modern_OpenGL_Tutorial_Text_Rendering_01
http://hacksncodes.blogspot.co.at/2013/08/efficient-text-rendering-system-with.html

Additional libraries

- FreeImage
- Assimp
- Bullet
- Freetype
- Boost

3-D Models

Ich habe für die Modellierung von 3D-Modelle Blender & Maya verwendet. Ich habe keine komplexe 3D-Modelle erstellt, sondern habe ich im internet nach frei verfügbare Modelle gesucht und diese nach meine Bedürfnise ‘manipuliert’.

Collision detection

Das ich die Kollision eher für den letzten Schritt im Projekt geplant habe, war es bitter fest zu stellen, das es sich ziemlich alles was mit logik in Spiel zu tun hat, auf eine Physics engine baut die unter anderem dann auch Collision detection anbietet. Ich habe daher viel umprogrammieren müssen.
Verwendet habe ich Bullet.
Waste - Objects und Basis Station sind einfache rigid bodies die auf einen user callback eingestellt sind, das Schiff ist ein Kinematic object für den ich alle Bewegungen (Flugsimulation) selbst programmiert habe.

Collision Shapes

Für eine präzise collision detection, habe ich für komplexe Modelle compound collision shapes verwendet (berechnet mit eine App. die HACD verwendet) die eine Approximierung des komplexen model darstellt.