# Sphere

Computergrafik UE

186.831

SS2015

# Team

| Name | Matr.Nr. | eMail |
|---|---|---|
| Hufler Andreas | 0626345 | e0626345@student.tuwien.ac.at |
| Moser Valentin | 0757786 | moser.valentin@gmail.com |

# Table of Contents



# Changes (!!)

The following changes have been made since the original submission 2:

- **The game works only on AMD for now.**
- The spotlight has an inner and outer angle between which the light will have a falloff.
- Frustum Culling should be working but has no positive effect on FPS (see details below).
- Glow: in addition to Bloom, certain meshes will have a similar effect that makes them seem to glow. This also counters the hard edges of bloomed parts of these meshes (see details below).
- Glow also has a sort of flickering effect (see glow below).
- Shadows will now consider the opacity of the object they are drawn onto. Shadow on glass will obviously be less strong than on a metal surface.

- The level has been completely reworked.
- When the player dies or wins, a message will appear on the screen to inform the player.
- FPS should have increased tremendously.
- Memory leaking should not happen anymore.
- Additional key bindings have been added (see below).
- Key G activates Godmode which will allow you to leave the sphere. Also NPCs will ignore you.
- Many other minor fixes of various things (too many/can't remember exactly).

# Limitations

- Playing the game while AMD Gaming Evolved is running will result unforeseen consequences.
- The launcher will not work if the path to the current directory contains any space characters.
- The game only displays correctly with AMD cards.

# Hardware

We tested the game using an AMD Radeon HD 7950 and the VisLab computer.

## Launching the Game

You can launch the game by either using the Launcher (Launcher.jar) or starting the game directly.

The Launcher will let you chose the command line options that are available with a GUI, but requires Java 8 to be installed.

Launching the game directly will use the following default settings:

*width = 1280*
*height = 800*
*frameRate = 60*
*fullScreen = false*

When launching the game via command line you can use these options:

*Sphere.exe (int)width (int)height (0/1)fullScreen (double)frameRate*

**Note**: All four options have to be present or defaults will be used! Values must be positive.

# Implementation Details

The following effects were implemented:

| Effect | Points |
|---|---|
| Deferred Shading | 1 |
| Shadow Maps | 1.5 |
| Spotlights | 0.5 |
| Bloom | 1 |
| | **4** |

## Free movable camera:

You can move around the scene as you like unless you collide with an object or try to leave the arena. There is a limit though on the maximum angle you may pitch the camera up and down.

## Moving objects / complex objects:

There are computer controlled “drones” in the arena which you’ll have to fight. They try to move around and catch you. You will notice that they have an antenna attached to them that is rotating. (sort of)

## Lighting / Shadow:

We are able to render one Ambient light and “infinite” numbers of directional-, point- or spotlights. Directional lights and point lights also cast shadows.

The depth map for shadow calculations is generated in two steps.
**Step A:** Each light has its own FBO for caching the depth information of all static objects that will never move. This is generated once after loading the scene.
**Step B:** When actually calculating shadow and lighting, the cached depth map gets copied over to the actual depth map. Then, only the dynamic objects that can move get rendered to further populate the depth information.
This technique saves a lot of power since most of our scene is static anyway and therefore the depth information will never change.

The spotlight has an inner and outer angle between which the light will have a falloff as requested by the submission 2 feedback.

## Transparency:

Most of the sphere that resembles the arena is made of transparent “glass”.

## View-Frustum-Culling:

Frustum Culling is working but has no positive effect on FPS. We think this is caused by the huge amount of meshes that get generated by Maya when exporting the whole level as one object.

## Bloom / Glow:

Bloom will kick in, when the light calculation finds that a pixel has a brightness that is over a certain threshold. These pixels are also written into the bloom buffer FBO. After that, the blooming parts get blurred using a gaussian blur filter and are then blended with the actual picture on the screen.

Glow is implemented in a way that makes use of the already ongoing bloom. In the geometry pass, pixels (diffuse texture) that belong to glowing meshes get stored in a separate texture of the FBO. After that, this texture gets copied over into the bloom buffer. So we see, the bloom buffer will not be empty when it gets filled by the actual bloom, though the glowing parts will be missing their lighting and be less bright than blooming parts. Gauss does the rest.

Another thing you may notice, is, that glow on our obstacles will sometimes flicker. We think this effect looks quite good apart from the fact that we can only enable/disable flickering on a per mesh base. This is due to the level hierarchy and would need a redesign of the 3d models. To individually handle the glowing “triangles” we’d have had to redo the whole level in Maya.

## Controls:

| | |
|---|---|
| W | Move forward |
| A | Strafe left |
| S | Move backward |
| D | Strafe right |
| Left Shift / X | Lose height |
| Spacebar | Gain height |
| Mouse (move) | Move camera |
| Mouse (left button) | Fire |

| | |
|---|---|
| ESC | exit game |
| G | toggle GodMode |
| | |
| F2 | toggle FPS |
| F3 | toggle Wireframe |
| F4 | toggle Texture-Sampling |
| F5 | toggle Mip Mapping |
| F8 | toggle Frustum-Culling |
| F9 | toggle Transparency |
| | |
| numpad + / . | increase ambient light |
| numpad - / , | decrease ambient light |
| L | toggle directional lights |
| | |
| ENTER | display current position |
| | |
| 1 | render final picture |
| 2 | bloom buffer with copied glow |
| 3 | gbuffer: world positions |
| 4 | gbuffer: diffuse |
| 5 | gbuffer: normals |
| 6 | gbuffer: glow |

### Gameplay:

Welcome to the arena where gravity is… well.. off. Use your gun to defeat the vicious drones that roam around. But be careful, they might fire back!

### Collision detection:

We use Bullet for collision detection between NPCs/ the Player and the arena's obstacles. To keep the player inside the arena we simply calculate the distance from the center. Bullets fired from the NPC/the Player will do their own collision detection to determine if they hit.

### Artificial intelligence aka. enemies:

Enemies are not really smart. They will go from waypoint to waypoint until they find a player. When they do, they will try to chase and shoot him.

## Libraries & 3rd party content

Bullet

collision detection

FreeImage

loading images for use as textures

Assimp 3

loading model data (meshes / materials)

GLM

class implementation and functions for 3d math

GLEW

wrapper library for opengl

Model/Mesh loaders are based on tutorials from:

http://www.learnopengl.com/

Models are created by us in Autodesk Maya, Textures are selfmade but might contain downloaded texture parts (e.G. metal brush).