Computergraphik UE SS2013

Matthias Karan 1027663

Hansjörg Hofer 1026632

Controls:

KEY	EFFECT
W, A, S, D	Move Character
Spacebar	Jump
Shift	Sprint
Mouse	Aim Color Gun / Look around
Left Mouse Button	Fire Color Ball
Num 1	Change Guncolor to WHITE
Num 2	Change Guncolor to RED
Num 3	Change Guncolor to GREEN
Num 4	Change Guncolor to BLUE
F1	Help
F3	Wireframe Mode On/Off
F8	ViewFrustrumCulling On/Off

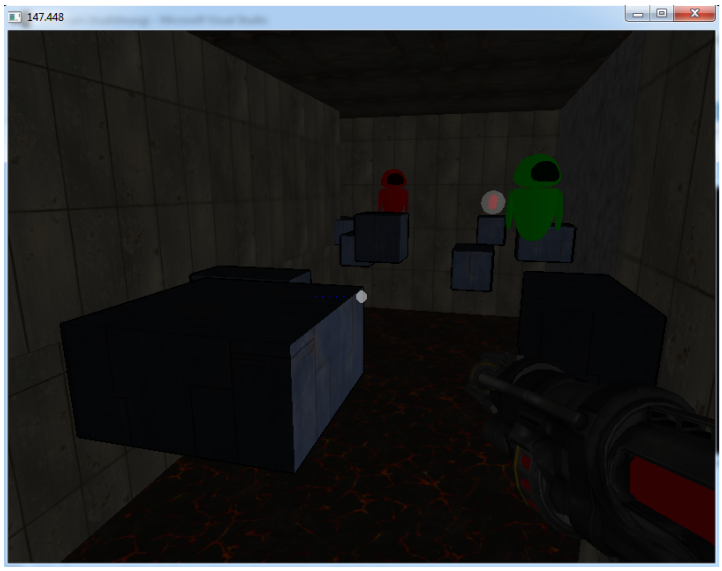
Change Resolution / Fullscreen Mode:

In our folder there is a file called “ScreenRes.ini”. In this file the first line stands for FullScreenMode (0 = OFF, 1 = ON), the second line for Screen-WIDTH and the third line for Screen HEIGHT. Just change the values as you desire and start the game.

Effects:

Toon Shading / Cel Shading

To give our game a comic-style we implemented Cel Shading. The shading of our toon-objects is done by our „toon_shader“. There we create 3 gradiations, depending on the dot product of N and L. Outlines of our toon-objects are achieved by the „inverted backface“-method, where backfaces are drawn slightly larger before front faces.[1]



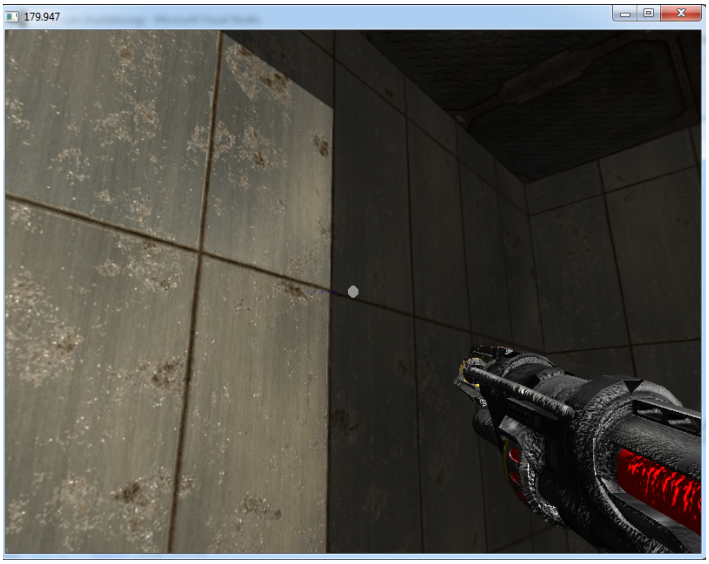
Normal Mapping

To create bumpy surfaces in our game, we implemented Normal Mapping. Therefore an additional Texture, the normal map, is used to „change the surface normal vectors according to some virtual bumps“.[2]

---

[1] http://www.informatik.uni-oldenburg.de/~trigger/content/opengl/opengl_course/slides/2010-JOGL-11-Toon-Shading.pdf

[2] http://en.wikibooks.org/wiki/GLSL_Programming/Unity/Lighting_of_Bumpy_Surfaces#Normal_Mapping



Shadow Mapping

To create realistic shadows from a point light, we created a shadow cubemap. The cubemap represents the distance to all the objects around the point light. So if a fragments distance to the light ist greater than the value in the cubemap, the fragment is in the shadow[3].

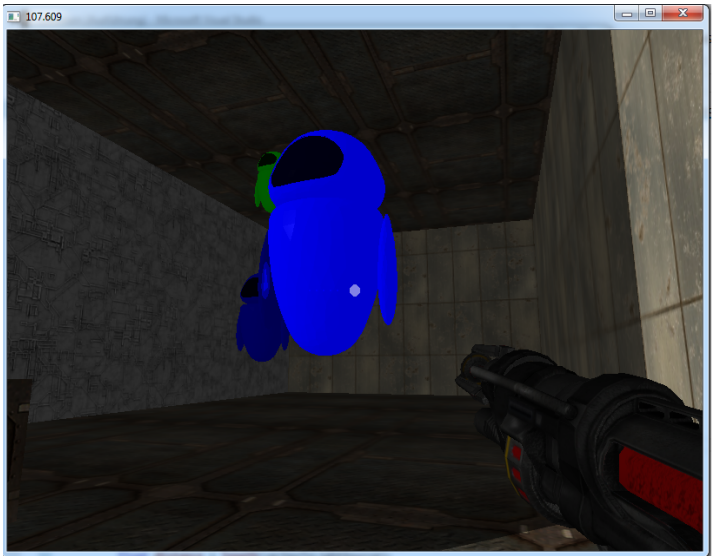
Features:

Complex Objects:

To load complex objects into our game we use the Assimp Library and create Meshes in our

[3] http://www.cg.tuwien.ac.at/courses/Realtime/repetitorium/2011/OmnidirShadows.pdf

Mesh-Class[4].



Animated Objects:

The heads of our enemies are animated, e.g they rotate. We achieved this by creating a simple child-structure. A parent object, in our case the Enemy Class, holds several child-meshes. The model matrix of our child-meshes is multiplied with the model matrix of the parent mesh.

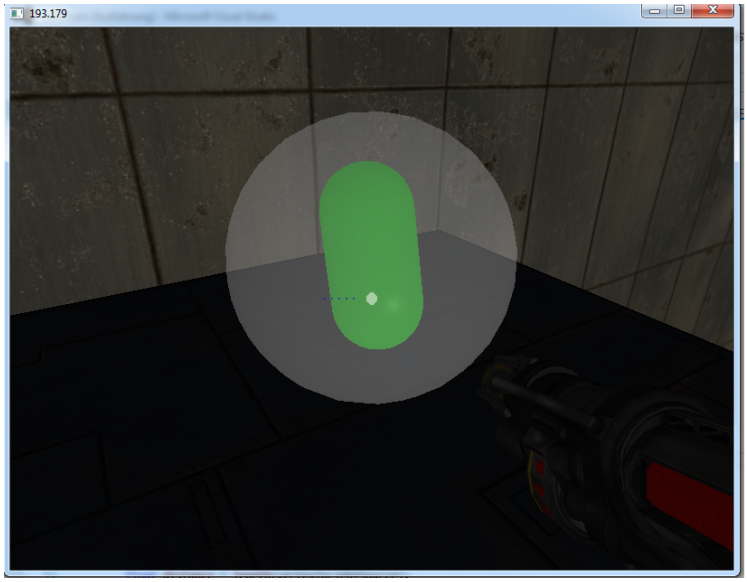
View Frustrum Culling:

All objects in our game have a „bounding box“. The object will be drawn only if its bounding box is inside of the „View Frustrum“. In can be activated/deactivated by pressing the F8-key.

Transparency:

In some parts of our game we included transparency on some objects. Our collectable color-items, that increase the amount of color-balls, are coated by a transparent hull (sphere). The crosshairs in the HUD is also transparent.

[4] http://assimp.sourceforge.net/



HUD:

To show the user how much red/blue/green color-balls are available, we created a dynamic HUD around the crosshairs.

Collision

Detection / Physics:

To make the Level as fun as possible we integrated the Library Bullet. Our complete Level is equipped with collision detection, so the player can jump on obstacles and can’t move outside

the level. To make the game more realistical and fun, physics were also integrated.[5]

Color Walls:

In our game you can color some walls with any color you want. To do so, just shoot towards the wall and it will change its color. This can be very useful to kill enemies ;-).
If you shoot white color-balls instead, they will be reflected from the wall.

[5] http://bulletphysics.org/wordpress/