

CG – Astrofighter Submission 2

Requirements:

1. Gameplay
2. Effects
3. Animated Objects
4. Frustum Culling
5. Controls
6. Experimenting with OpenGL

Requirements for Experimenting with Opengl:

1. Buffer-Objects: FBO (Frame Buffer Object) or UBO (Uniform Buffer Object)
2. Blending: Use hardware blending (glBlend*) somewhere in your code. Make sure it can be toggled on/off
3. Mip Mapping (on/off)
4. Texture-Sampling-Quality (Bi/Trilinear Filtering)

Used Effects:

1. Bloom (1p)
2. Contours (1p)
3. Normal Mapping (1p)
4. Particles Effect (CPU) (0.5p)
5. Cel Shading (0.5p)

Description of the implementation

GAMEPLAY: This game is a multiplayer game (2 players). Each player has his own area (the camera view) in the game and each one is controlled independently. The controls keys are separated at the keyboard, so that each player has easy access to his control keys. The players are controlling spaceships in the space. The goal of the game is to hit the other player with a laser and finally destroy the other spaceship. The ships are freely moveable in to directions. There is only on light source (a sun) in the space (the light of the earth which is visible). The ships also have to avoid asteroids, which are in space and rotating. Each player has a life bar with the life points and an enemy direction arrow which shows, where the other spaceship is. If one player destroyed the other spaceship, the ship explodes. Then this destroyed ship is not moveable anymore and cannot shoot.

ADDITIONAL FEATURES: We implemented the feature, that you can see on an arrow, where the other spaceship is - like a radar. The bullet of a spaceship which shoots is a red laser, which disappears by time. The explosion (when the other spaceship got hit) is shown as an explosion of small particles.

ANIMATED OBJECTS:

There is a gun under each spaceship which is loaded separately and is moving forward when the player shoot and moving back afterwards => shooting animation/animated gun.

BLENDING:

We use glBlend for the particles. The disappearing functionality of the particles is actually the blending. The particles are going from alpha value 1 to 0.

VIEW FRUSTUM CULLING:

When the objects move out of the camera view (projection view) of one player, the objects are getting culled. This improves the performance of the game. You can see how much objects are culled by enabling the frustum culling with KEY F8.

GAME OBJECTS:

<i>Object</i>	<i>Lighting settings/ effect settings</i>	<i>Texture</i>
2 spaceships	Blinn-Phong illumination + Cel Shading + Bloom + Contours + Explosion particles	Standard UV Mapping
100 asteroids	Blinn-Phong illumination + Cel Shading + Bloom + Bump mapping	Standard UV Mapping + Normal Map

EFFECTS: We implemented 5 effects (listed above). We use the bloom effect for general highlighting objects/textures/specular areas. The cel shading is also used for all objects. If an enemy is visible in one's player's camera view, it has a contours effect so you can see the enemy better in the space as well as behind asteroids. The particles effect is implemented as an explosion of particles when one player hits the other spaceship. If the other spaceship is destroyed the particles change the color from red/orange to black/gray. The asteroids have a bump mapping to display more structure (craters) and to improve the game performance by avoiding too complexed asteroids.

LIBRARIES AND SOURCES:

Library	Function	URL
ASSIMP	For model loading	http://assimp.sourceforge.net
OpenGL	3D engine	https://www.opengl.org
GLFW	Creating windows	http://www.glfw.org
GLEW	OpenGL extensions	http://glew.sourceforge.net
FreeImage	Texture loading	http://freeimage.sourceforge.net
Sources	Function	URL
Sourceforce	Loading the texture with FreeImage	https://sourceforge.net/p/freeimage/discussion/36111/thread/e0565591
Lighthouse3D	Frustum culling library and code	http://www.lighthouse3d.com/tutorials/view-frustum-culling

HOW THE EFFECTS ARE IMPLEMENTED

EFFECT	Implementation	Source URL
Cel Shading	This shader is directly implemented in the shader programs	http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/toon-shader-version-ii/ http://sunandblackcat.com/tipFullView.php?l=eng&topicid=15
Bloom	For bloom you take the 2D texture of the scene which is stored in an FBO and gets blurred by an own shader program	https://learnopengl.com/#!Advanced-Lighting/Bloom
Contours	Contours is actually an edge detection shader. You store the scene in a 2D texture and do an edge kernel on this texture with an own shader program	https://learnopengl.com/#!Advanced-OpenGL/Framebuffers
Particles effect	The particles are small 2D textures which are facing to the camera. The origin of the explosion is the point where the ship got hit. From there the particles are going in all directions. All particles have the same vertices but have different position colors and directions.	https://learnopengl.com/#!In-Practice/2D-Game/Particles http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing

Normal Mapping	The normal mapping uses a 2D normal map and tangents and bitangents vectors to calculate a more complexed structure (bumps) on an image. The normal mapping is done with an own shader program	https://learnopengl.com/#!Advanced-Lighting/Normal-Mapping
----------------	--	---

MODEL LOADING: We do the model loading by assimp and are loading DAE (Collada) files. These DAE files are generated with Blender (<https://www.blender.org>). Currently we implemented a very easy loading where we group the meshes in the model and loading full connected models. The hierarchical structure of the animated objects is done in the game.

Cameras: We implemented a split screen for each player of the game. The camera is “sitting” in the “cockpit” of the spaceship but is able to move around in the space. The ship follows this movement.

Skybox: We are using a cube as skybox for the scene as space. It has a skybox texture which provides smooth transitions at the edges so that the edges are not visible. The skybox texture is separated in the different faces and loaded with the texture loading.

Controls: There are several control elements implemented. Each player has its own keys to move the camera and the spaceship. To provide additional functionalities there some other control keys.

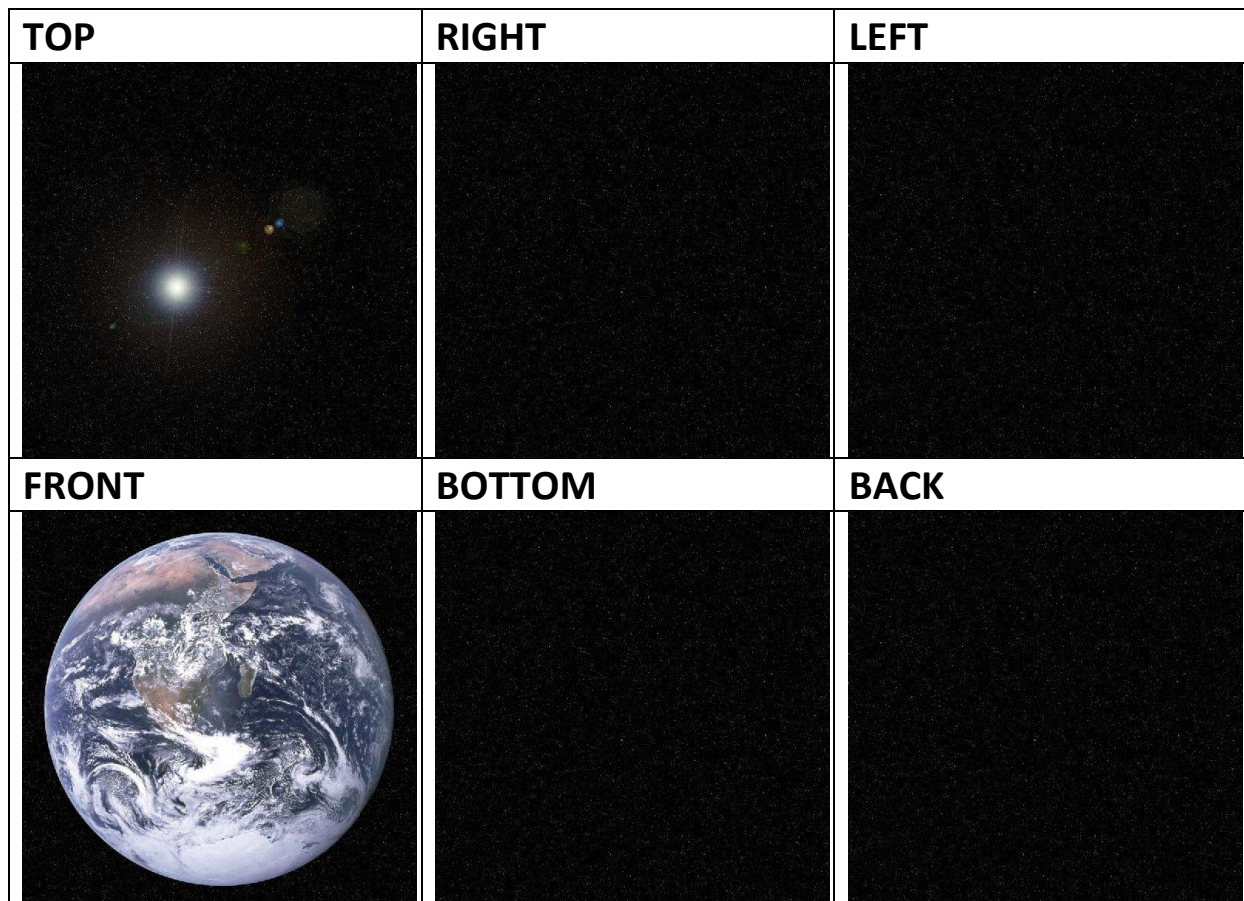
Keyboard Player 1	Action
A, W, D, S	rotate left, go forward, rotate right, go backwards
T,G	rotate up, rotate down
TAB	toggle freely moveable camera and spaceship camera
Keyboard Player 2	Action
LEFT, UP, RIGHT, DOWN	rotate left, go forward, rotate right, go backwards
NP 8, NP 5	rotate up, rotate down
Additional Controls	Action
F1	Help
F2	Frame time on/off
F3	Wire frame on/off
F4	Texture sampling quality (off/nearest neighbor/bilinear)
F5	Mip mapping quality
F6	EMPTY
F7	EMPTY
F8	View frustum culling on/off
F9	Blending on/off
KB 1	Bloom effect on/off
KB 2	Contours effect on/off

KB 3	Normal mapping effect on/off
KB 4	Particles effect on/off
KB 5	Cel shading effect on/off
ENTER	Restart game if one player died
SHIFT	Increase the speed of the first player
P	Toggle camera rotation with mouse of player 1 on/off

Lighting: We are using sun light for the spaceships and the asteroids. The lighting model is Blinn-Phong. The skybox has no lighting dependence – it just uses the color of the texture to avoid specular effects. The additional effect is, that you do not see the edges of the skybox if you do not use lighting for this model.

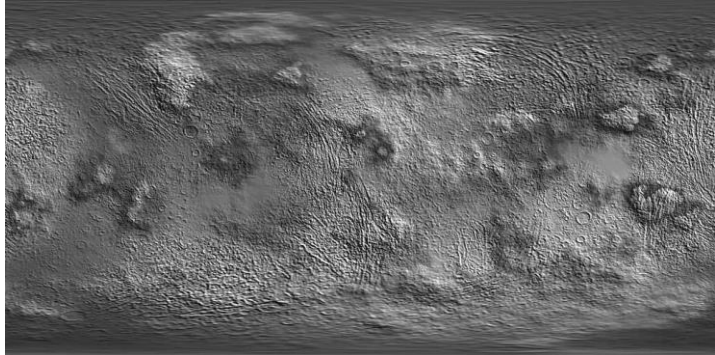
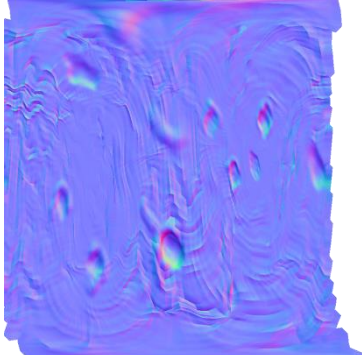
Textures:

Skybox:



URL: <http://www.custommapmakers.org/skyboxes.php>

Asteroid

Asteroid (asteroid.jpg)	NormalMap (asteroidNormalMap.png)
	

URL: https://commons.wikimedia.org/wiki/File:Generic_Celestia_asteroid_texture.jpg

Spaceship:

ShipUVMap.jpg
