

Bermuda Island: Arrival - Documentation

Controls

Commands	Action
W, A, S, D	Move character
Mouse	Control view
Right Mouse	Zoom

There are a total of 6 objects to collect. After every one of them if collected the game is won, however there is also a timer. If the time runs out before every object is collected, the game is lost. A special object (the telescope & map) allows the player to zoom.

Camera

We are using first person view with a FOV of 65. The view is limited to + and - 90 degrees up, so the camera cannot turn upside down.

Collision Detection

For collision detection we are using Bullet. Our Player collides with the terrain and the water (jesus mode).

- Requirements
 - Effects
 - Shadow Mapping (1.5): For Shadow Mapping we are rendering a depth texture from the light source's direction using orthogonal projection to the origin and are drawing shadows using that information. Our far plane is very far away, therefore a high shadow map resolution of 4096 is used. The PCF feature is therefore quite minimized, but still noticeable if one looks hard enough.
 - Environment Mapping (1): For the environment map we are reusing the cube map generated for the skybox and are sampling pixels from it using a shader. This shader is merged into the shadow mapping shader using a boolean flag to determine if an object is supposed to use environment mapping. For our pickup-objects and our water plane we are using environment mapping.
 - Cel-Shading (0.5): After color calculation the color adjusted based on intensity in a fragment-shader.
 - Depth of Field (1.5): Everything is rendered into a color- and a depth-texture. The color texture is then manipulated based on the depth texture in the fragment-shader (post-processing).
 - Complex Objects
 - Terrain, Palm trees, Object following the player (look up)

- All models are self-made
 - Object and Texture loaders are implemented (FreeImage, Stbimage)
- Animated Objects
 - Sphere moving with the player while rotating
- View-Frustum-Culling
 - Since all our trees are in one mesh (:/) and our terrain has way too many vertices for our current algorithm to work efficiently (currently checking if points are inside frustum), we are only checking the collectable objects. Terrain and trees therefore never get culled.
- Experimenting with OpenGL
 - Several FrameBuffers were used
 - Shadow Mapping
 - Depth of Field
 - Drawing outlines in Post-Processing (removed because they did not look fitting - DEPRECATED)
 - Mip Mapping
 - Implemented with Nearest filtering + Anisotropic filtering
- Features
 - Interaction with models to collect them
 - Depth of field shader includes autofocus
 - Collision detection
 - Startup parameters, Fullscreen flag can be set
- Illumination
 - We only have one light source, which represents the moon, even though it is very bright because of projector problems

Tools:

Blender <https://blenderartists.org/forum/>

Gimp <https://www.gimp.org/>

Paint

Libraries:

FreeImage <http://freeimage.sourceforge.net/>

GLM <http://glm.g-truc.net/0.9.8/index.html>

GLEW <http://glew.sourceforge.net/>

GLFW <http://www.glfw.org/>

Bullet <https://github.com/bulletphysics/bullet3/releases>

Tutorials / Sources:

[https://blenderartists.org/forum/showthread.php?237488-GLSL-depth-of-field-with-bokeh-v2-4-\(update\)](https://blenderartists.org/forum/showthread.php?237488-GLSL-depth-of-field-with-bokeh-v2-4-(update))

<https://r3dux.org/2014/10/how-to-load-an-opengl-texture-using-the-freeimagelibrary-or-freeimageplus-technically/>

<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-6-keyboard-andmouse/>

<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/>

<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading/>

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-9-vbo-indexing/>

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-14-render-to-texture/>

https://developer.mozilla.org/enUS/docs/Games/Techniques/3D_collision_detection

<https://computergraphics.stackexchange.com/questions/3646/opengl-glsl-sobel-edge-detection-filter> (later removed)

<https://learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping>

<https://learnopengl.com/#!Advanced-OpenGL/Cubemaps>