Mir

Tobias Winter | CGUE22

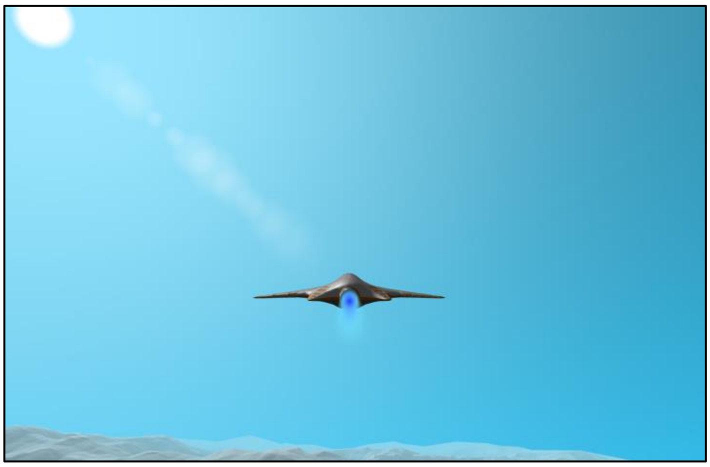
Features

Sky shader

Shader using ray direction and simple distance function to estimate atmosphere color. Atmosphere assumed curved with uniform density.

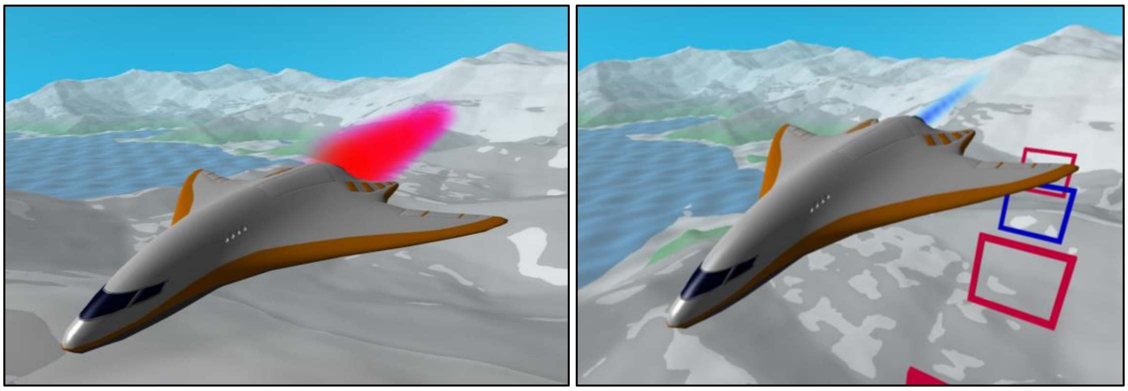
Lens flare

Ray direction and camera direction is calculated and used to give the visual appearance of a lens flare.



Particle system

Particle system where vertexes consisting of position and time are sent to shader where the particles are animated and billboarded to always face the player. The effect can be seen on the aircraft’s engine plume. Note that the plume changes color when the aircraft is boosted by a blue objective.

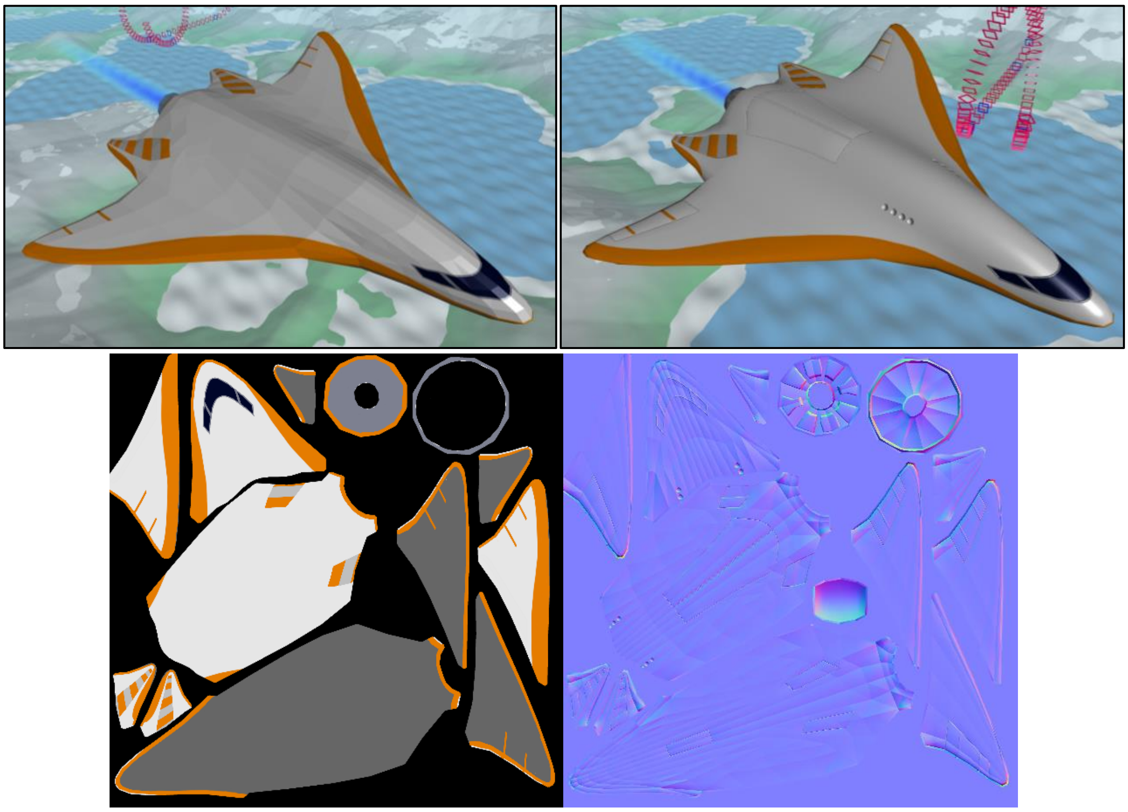


Mesh loader

A mesh loader that loads obj files and calculates tangent / bitangent as well as barycentric coordinates.

Tangent space normal mapping

Normals from a high-poly model was baked onto the normal-texture of a low-poly version of the same aircraft. The tangent and bitangent are then used to correctly map the normals of the low-poly model using the normal-texture.



Objective shader

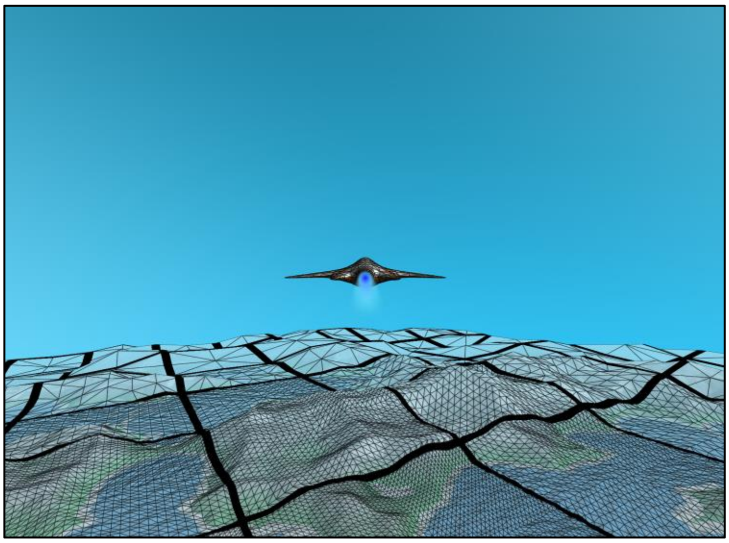
The objective mesh is constructed on the GPU using geometry shader, only positions and objective type are being sent to shader. Note that the objectives are animated. Therefore, a large number of objectives can be drawn without any significant loss in performance.

Text engine

Strings being sent as vertex data, and quads being constructed on the fly. Samples from font map texture.

Terrain shader

Uses GPU terrain tessellation to generate geometry from a set of patches being sent from the CPU. The vertices are then curved and displaced by a function using an interpolated texture sampling of Worley noise (made using a python script) to generate fractal noise. The diffuse colors are then calculated from another function before the terrain is shaded using a numerical approximation to get the normals. The water shader is simply a superposition of a couple of sine waves as this makes for cheap normal-functions. Note that the water is animated. The displacement function is being mirrored on the CPU-side by the player entity such that collision is detected between player an terrain.



Gameplay

The game has two modes, either you can try to hit all objectives as fast as possible, or you can fly around freely to explore. Blue objectives give the player a boost.

Controls

W,S	Pitch
A,D	Roll
Q,E	Yaw

Debug controls

Mouse	Control camera (in debug mode)
1,2	Turn on / off tangent space normal mapping*
3,4	Turn on / off debug mode*
5,6	Turn on / off HUD
TAB	Turn on / off wireframe mode**

* use these two setting to observe the tangent normal mapping.
** use this setting to observe terrain tessellation.

Libraries used

GLFW	-
GLEW	-
GLM	Math framework
STB	Loading textures