Andreas Brunner
Thorsten Korpitsch

Starman Documentation

Implementierung (Compulsory)

- **HUD**
  The HUD was implemented using Freetype.
- **Light Mapping (+ Textures)**
  A lightmap was baked in Blender for a static object and is combined with the diffuse texture during runtime.
- **Physics Engine**
  Bullet-Physics is used as Physics Engine.
- **Frustum Culling**
  The frustration of culling is realized by means of bounding spheres.
- **Hierarchical Animation**
  Around the pickups (yellow + euro) a smaller version of the pickup is moving around them. (They get the translation, etc. from the big pickup that you can catch and have their own transformation around their parent).
- **Complex Models from Files**
  Models created with blender are loaded with the assimp framework.
- **Animation via Physics Engine**
  All objects are moved via physics, the transformation/rotation values are extracted from the World transformation and used for drawing.

Features

In the game you can fly around with the spaceship, trying to avoid colliding with asteroids, picking up the yellow euro pickups for additional points (and health) and shoot down the white enemies. You can also ram into enemies, but this will cost you health points. If you crash into an asteroid it is over. If you drop below 0 health points the game is over too. You win when all enemies are destroyed.

Lightning

The game is lighted with 1 Sun. All objects are colored from textures.

Additional Libraries

- Assimp
- Freetype
- BulletPhysics

Effects

- **Lightmapping (+ Texture)** 0.5 Points
- **GPU Particle System (Compute Shader)** 1.0 Points
- **Cell Shading + Contours (Edge Detection)** 1.5 Points

Andreas Brunner
Thorsten Korpitsch

- **Procedural Generated Textures** 1.0 Points

Lightmapping (+ Texture)

Reference: Repetitorium Slides 2018

The textures were baked with blender into a seperate texture. At runtime both textures are loaded into the shader and combined there.

GPU Particle System (Compute Shader)

Reference: Repetitorium Slides 2018

We followed the instructions of the slides, as well as the tips of our tutors.
Shader-Buffer-Objects for Position and Velocity are created and filled with the data of 2 initial particles.
The compute Shader is updating and spawning particles and returning the new values to the ping-pong Shader-Buffer-Objects (which are used for input and output alternating).

A geometry Shader is used to spawn vertices at the position of the particles. The Vertex shader is basically a pass through shader. The fragment shader then colors the “particle” through a texture.

Cell Shading + Contours (Edge Detection)

Reference: https://raptor.developpez.com/tutorial/rendering/celshading/
Reference: https://en.wikipedia.org/wiki/Cel_shading

The cell shading clamps the light values to 4 different stages. It happens directly in the shader used to draw the colors of the objects. When the Phong calculation is done, the values get adjusted if cell shading is enabled. For the Contours the the colors and depths are rendered into a Framebuffer, which we use for post processing. A Sobel filter is used on the depth map, revealing edges. The edges are then combined with the color value resulting in outlines of objects.

All objects except for the space station are cell shaded and contour shaded. The station is only contour shaded.

Procedural Generated Textures

Reference: https://solarianprogrammer.com/2012/07/18/perlin-noise-cpp-11/

A perlin noise function is used to generate a texture. The texture is then used to color asteroids.

Models

Andreas Brunner
Thorsten Korpitsch

Models and Textures were created by ourselves in Blender.

Controls

- Escape - Closes the game
- F2 - enables/disables Debug-Mode
- F3 - enables/disables Wire-Frame-Mode
- F6 - enables/disables Cell-Shading
- F7 - enables/disables Post-processing (Contour Shading)
- F8 - enables/disables Face-Culling
- F9 - enables/disables debug camera (switch from player view to debug camera or back again)
- F10 - enables/disables Frustum Culling

Player Controls:

- W, S - Accelerate/Decelerate
- A, D - Strafe left/right
- Left-Click/Right-Click - Shoot with left/right weapons
- Mouse-Movement - Change Direction (look around)

Debug Camera Controls:

- W, A, S, D - Move forward/backward/left/right
- Mouse-Movement - Look around