Hexagonum

Tom Lautenbach, 12044805
Johannes Trippl, 00826563

We focused on building the render engine first. Once the engine was completed, we designed the game and implemented the game logic.

Resources used

- https://learnopengl.com/
- Home (opengl-tutorial.org)
- OpenGL 4 Shading Language Cookbook (978-1789342253)

Libraries used

- assimp
- SDL
- PhysX
- FreeImage
- glew
- freetype

Mandatory, Gameplay

Playable (3)
The exe file is executable. Once one starts the game, the first level (LEVEL 1) gets loaded.

Advanced Gameplay (3)
Game logic and game design is finalized. At the beginning of each level, the player can have a look at the level for some seconds and needs to memorize the path. Then the goal is to move through the level taking the memorized path to reach the next level to add up the highscore.

Win/Lose condition (3)
In each level, the player is allowed to commit two mistakes. A third one would lead into a game over state. If the player passes all fields shown in the beginning, a level is completed and the very next level gets loaded. The higher the level, the bigger the points a player gets by passing a field.

Intuitive Controls (2)
We decided on intuitive third person controls (WASD and mouse). You also need to make use of the character's jetpack (SPACEBAR).

Intuitive Camera (2)
Camera follows the player (third person perspective).

Moving Objects (2)
Besides the character moving you can discover a moving hexagon in level 3 and a moving wall from level 2. Further the barrels are moving once the character collides with them.

Framerate independency (3)
You can check this by adjusting refresh rate to 10 e.g.

Lighting (2)
A directional light (Sun) is used.

3D (6) & Textures (2)
Models + materials are loaded via assimp.

Changeable Parameters (1)
You can adjust game settings in bin/settings.ini.

Documentation (1)
This pdf.

Optional Gameplay

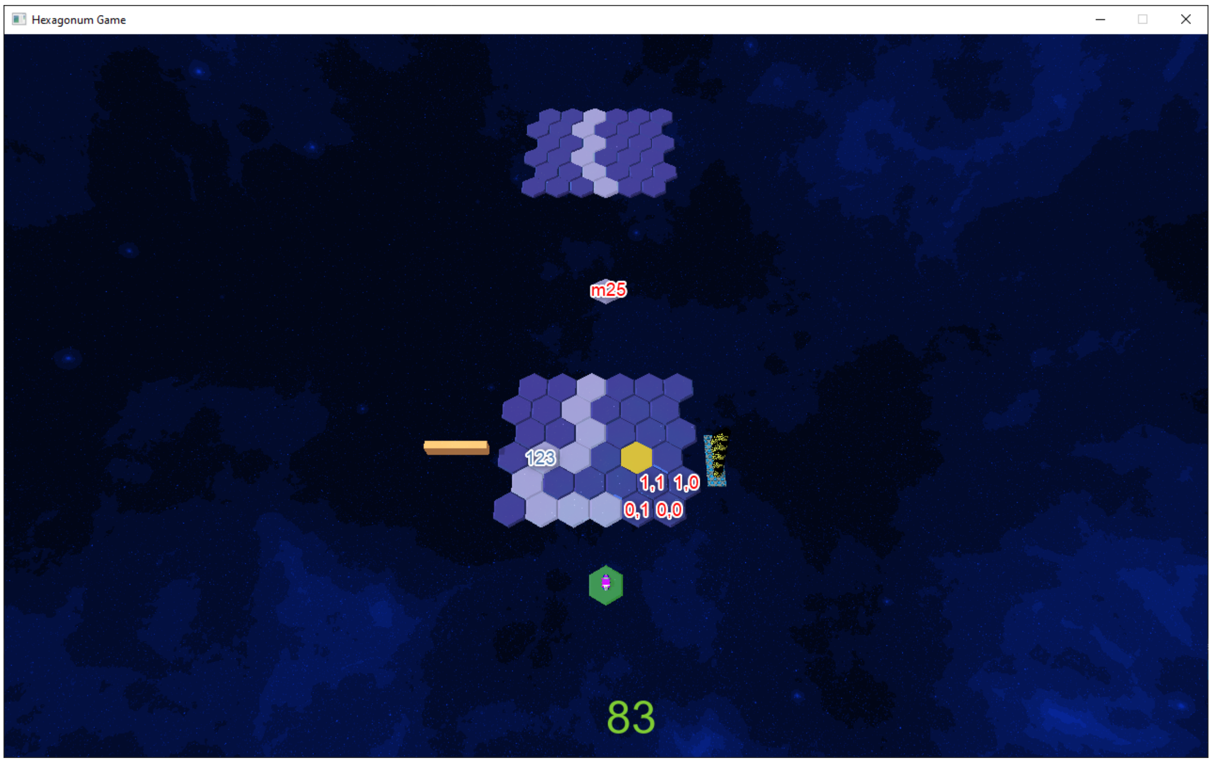
Hud (4)
You can enable the high score list by pressing TAB. You trigger a highscore save as soon as your game is over. Highscore is persisted to a file. Highscore is calculated levelnumber times passed fields. Once done entering your name, save your high score by pressing ENTER. Hit TAB again and you will see that your value and name are now part of the list. If you load up the game again, they are still in the list, since they've been persisted to the high score file. Note that only the top 5 values are shown in the TAB-view.

Advanced Physics (6)
The Character Model is a PhysX Character Controller. Once the player hits the "dangerous" field (marked in yellow in the level 1 preview), this field will fling them away, which results in the Character Controller being transformed into a Dynamic Rigid Body with normal physics properties. The way the dangerous field flings the player away is that the player flies straight into the barrels (which are Dynamic Rigid Bodies as well). The collision is automatically calculated and simulated by PhysX.

Collision detection (4)
The collision detection can be observed e.g. when the player moves onto a new hexagon and its color changes - we catch collisions in the character model and then test for what exactly the player has moved against.

Scripting Language Integration (6) - not defined in design document
One can simply add or adjust levels to the game without recompiling. This works by adding a simple file to /res/levels. The file has to be specified as follows:

- Level<levelnumber>.txt as a filename
- First two lines: width and length of the playfields to be generated (max. 9)
- Third line: Number of playfields to be generated (max. 2)
- Empty line in between
- Definition of the path through the playfield in all following lines
    - First digit: playfield number
    - Second and third digits: coordinates of the path hexagon in the playfield (see figure below)
- Specify a moving hexagon between two playfields with an “m” and a number for the amplitude of the movement it will receive. 25 has proven to be a good value for this.



Screenshot 1: Level design reference

Effects

CPU Particle System (8)
Whenever the player steps on a platform, some sparks around it are spawning.

Environment Map (8)
You can immediately see a skybox as soon as the game is loaded. We also verified the correctness of the reflections by drawing some arrows to the skybox.

Shadowmap with PCF (16)
Press “M” to render the depthmap to a quad at the lower left corner of the screen.

Cel Shading (4)
The whole game is rendered using cel shading.

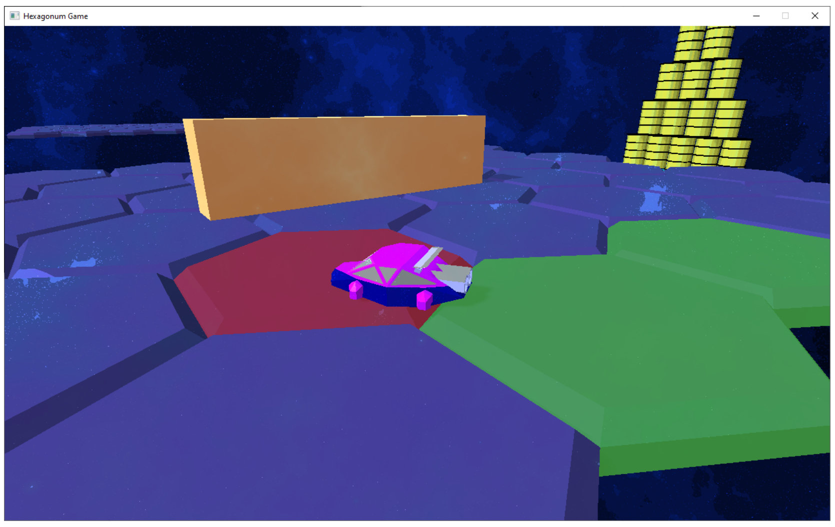
Controls

Key	Action
WASD + Mouse	Movement Controls
M	render depthmap quad
TAB	Show Top 5 Highscore
ENTER	After entering name save to highscores
ESC	Leave Game

Settings (settings.ini file)

Key	Value
width	resolution width in pixels
height	resolution height in pixels
fullscreen	0 = window mode 2 = fullscreen borderless
refreshrate	frames per second
gamma	gamma correction (1.0 is default)

Screenshots



Screenshot 2: Gameplay