El Crustachio

A Steamy Getaway

Daniel Pichler 01627759
Oskar Perl 01625757

"El Crustachio - A Steamy Getaway" is a vertical Singleplayer platformer. You control the crustacean *"El Crustachio"* and try to not get cooked in a pot by jumping on platforms and reach the top of the pot.

Controls:

[w]	move forward
[s]	move backward
[a,d]	strafing left and right respectively
[space]	jump
[mouse]	move the camera around and change the characters rotation
[middle mouse button]	hold down the middle mouse button to look around without rotating the character model
[F1]	toggle wireframe
[F2]	toggle backfaces
[ESC]	close the game
[ENTER]	proceed from the win/lose screen

Development Status:

Gameplay:

The characters movement direction can be controlled using the w,a,s,d keys. The rotation of the character as well as the camera can be controlled with the mouse. The player can jump between the platforms using the space key.

In the beginning the player is greeted with a start screen. Pressing enter in the start screen will start the the first level.
The game currently consists of three levels. The levels feature multiple platforms in the form of carrot slices and a wooden platform. When the wooden platform is reached the player is safe and finishes the level by jumping one more time while on the wooden platform. When the level is completed a splash screen will be shown via a GUI and the next level will be loaded. All levels feature a lose condition. The lose condition in all levels is reached when the player drops in the water. The level of the water will rise at a steady rate as time passes in the level. When the lose condition is triggered the current a splash screen will be rendered in the scene via a GUI and level will be reset. When all Levels are completed an additional win Screen will be displayed, pressing enter will return the player to the start screen.

The player model is retrieved from an obj file. We load the model to the scene, using OBJ-Loader(https://github.com/Bly7/OBJ-Loader). What is more, the player features a custom texture we created using Adobe Substance.

In the settings.ini file several parameters can be changed. The most important ones are: width and height of the window, the refresh_rate, set fullscreen, brightness of the scene, fov and the mouse sensitivity.

The game utilizes Nvidia PhysX(https://github.com/NVIDIAGameWorks/PhysX) to simulate collision between the player and the objects in the scene.

Lighting:

The Scene is lit using a point light above the level and a directional Light.

Effects:

- -Cell Shading:
    - Our main character is rendered using our implementation of cell shading.
- -Contours via edge detection:
    - We rendered the scene and the actor as a depth map to an FBO. We applied a sobel filter to the texture retrieved from the FBO. We applied a threshold on the resulting image containing the edges so that the resulting image only contains black and white values and added it to the the scene.
- -Shadow maps with PCF:
    - We render the shadow of the main character using shadow mapping, to give the player visual feedback on where the player model is located in 3d space. To achieve his we rendered the character from the lights point of view to a depth map. Wie used the depth map to determine which fragment of the platforms are lit and which are in the shadow of the character. Additionally we used Percentage Closer Filtering to smoothen the shadows edges.

- -CPU Particle System
    - We use a Particle system to simulate small cartoonish clouds of steam on the surface of the water plane. The particles are spawned randomly in the proximity of the water surface. The Particle System consists of Parts
        - Particles, are the base form, they consist of information regarding the life-length, vertices, the elapsed time and other geometry relevant information.
        - The Particle Manager, manages as the Name suggests all Particles, it has a member with all active Particles and updates that list every frame by calling the Update Method of each Particle. Is the resulting boolean false, the Particle will be marked as dead and will be removed subsequently.
        - The Particle System, is responsible of Randomly spawning Particles on the map in a given interval.

- -Planar Reflections:
    - We use planar reflections to render the water plane to look like actual water. We split the scene into two parts:
        - the Refraction, which is everything below the water
        - and the Reflection, which is everything above the water, but turned upside down. This is achieved by inverting the camera Position and the pitch of the camera in relation to the Water Surface when rendering to the FBO.

      Both are rendered to FBOs, the resulting texture is then passed onto the fragment shader along with a DuDv map (consisting of Red and Green values) and a moveFactor. This factor is updated on the CPU and moves the DuDv map with which we create the wave effect by distorting the Texture. The final Texture is then rendered onto the Water plane.