Documentation

Group/Game Name: Getting Home
Brief description of implementation: Getting Home is a 3D platformer game, which was implemented using OpenGL.
Additional libraries: GLFW, GLEW, Assimp model loading library, Bullet physics library

Gameplay:

Mandatory:

- 3D Geometry: We made our own models for the rocket, rocket parts, and stone platforms, which are loaded using the Assimp library. There are platforms with cube geometry that are generated programmatically.
- Playable: We implemented a 3D platformer game, where the player can jump on floating platforms, which are both static and moving.
- Advanced Gameplay: There are 6 rocket parts scattered all over the level that have to be collected to win. There is a time limit, and if it runs out the player loses, however every collected item adds some extra time.
- Min 60 FPS and Frame Rate Independence: The game runs at 60 fps and all movement (player, platform movement, animations) use delta time to calculate movement.
- Win/Lose Condition: The player loses the time runs out (shown in the top right corner of the HUD). They win if they collect all 6 rocket parts in time.
- Intuitive controls: We use typical 3D game controls: WASD for movement and space for jumping. The camera angle can be controlled by moving the mouse.
- Intuitive Camera: The camera always points in the direction the player is moving. While standing still, the camera can be moved around the player to get a view of the surroundings.
- Illumination model: We use directional lighting as well as multiple point lights so every part of the level is illuminated properly.
- Textures: Our game supports loading models with textures given by .mtl files, manually attaching a loaded texture to a model, and attaching a shader texture to a model.
- Moving Objects: Some of the platforms are moving between two given points. The movement affects the player when standing on them.
- Documentation:
- Adjustable Parameters:
    - F1: Wireframe mode
    - F2: Backface culling
    - F3: Fullscreen
      In settings.ini:
    - Refresh rate
    - Brightness multiplier (gamma)

Other keymappings:

- F4: Bullet Debug Shader (causes framerate drop, only for demonstration)
- F5: Reset player position to 0,0,0
- F6: Turn off time limit
- F7: Toggle HUD
- Resize the window while in windowed mode

Optional:

- Collision Detection (Basic Physics): All items in the levels have collision, which is implemented using the Bullet library. The player can stand on and get moved by the platforms. The collectibles are removed from the world when the player collides with them.
- Heads-up Display: A heads-up display shows the items that have been collected (top left corner) and the time left (top right corner). Win and lose screens are also rendered using the heads-up display.

Effects:

Advanced Modelling:

- CPU Particle System: Over the rocket in the middle of the level, black smoke can be seen, which is implemented using CPU particles. We mostly used this tutorial for the particles/instancing, and changed it to render textures as particles. https://levelup.gitconnected.com/how-to-create-instanced-particles-in-opengl-24cb089911e2

Animation:

- Hierarchical Animation has been implemented where any animation applied to a mesh also transforms its child meshes. Animations can be set by providing different transformations as keyframes. This can be seen on the player, which bounces up and down while the legs also individually move. Also on the rocket, which bounces and also has a rotating animation on the legs.

Texturing:

- Procedural Texture: A procedural texture is used on the ground of the level, to create the effect of a strange moving planet surface. Specifically, we used a shader that implements Fractal Brownian Motion. For this we found an existing shader on Shadertoy and adapted it to look nice with our game: https://www.shadertoy.com/view/llsSzB

Shading:

- Physically Based Shading: All platforms and some of the collectible objects use physically based shading using the Cook-Torrance model, where the result is calculated from 5 different given textures (albedo, normal, metallic, roughness, ao). The shader used for this was found from LearnOpenGL's lighting/PBR tutorial: https://learnopengl.com/PBR/Lighting and the textures were taken from https://freepbr.com/ which was also recommended in the tutorial.

Walk-through:
The game starts with the player in the middle of the level, with floating platforms located all around you. You can see that there are objects on some of the platforms at the top. You will have to jump upwards on the platforms in order to reach the collectibles. The controls are WASD to move and SPACE to jump. On the top right corner, you can see the remaining time. There is not really a linear walkthrough here as the items can be collected in any order, but you have to find out what order the platforms can be jumped on. Items that have been collected are shown in the top left corner. You also restore some time when you collect an item. When all 6 collectibles have been collected, a win screen is shown. If the time runs out, a game-over screen is shown. On the win/game-over screen, you can quit the game with ESC.