Start the game by pressing ENTER.
Then use WASD to move around.
Find the odd-one-out among the trees (The one which is moving VERY slightly.)
If successful, you are redirected back to the game menu = darkness.
Why is my game menu just black? Find an explanation below.

I implemented the following tasks:

- **3D Geometry:** I implemented an Assimp model loader. In my game, it succesfully handles .obj and .dae files. Everything in my environment is retrieved via the object loader.
- **Playable + Advanced Gameplay:** I implemented a game which is very different to my proposal. In my game, the player is a little cube thrown into a forest. All the trees are static - except one which moves slightly. If the player touches this tree, the game is won. The game can be started by pressing ENTER. Then use WASD for controlling the cube. I actually had a user interface implemented (with text output). However, it clashed with Assimp and I could not resolve the issue. That is why I removed it again.
- **Min. 60 FPS and Framerate Independence:** The speed of the player and the animated tree trunk are adapted to the current framerate, making the game framerate independent. FPS >= 60.
- **Win/Lose Condition:** The player wins when he finds the odd one out among the trees.
- **Intuitive Controls:** WASD + ENTER = very commonly used.
- **Illumination Model:** I added materials and a directional light source. However, as I am aiming for a comic-style look, I commented out the code in the shader. In animation.frag simply comment out the code and delete line 48 to display an illuminated scene.
- **Textures:** Even though very simple (one color), all objects are imported and rendered with their textures.
- **Moving Objects:** Only one tree moves slightly. That’s the point of the game, though.
- **Documentation:** You are currently reading it.

- **GPU Vertex Skinning:** One of the trees is animated. Followed the tutorial on learnopengl.com.
- **Contours via Edge Detection:** Followed the tutorial referenced below the assignment. I rendered the normals into a separate Renderbuffer. Then I extracted the edges with a Sobel operator.