

# RePeter

## Submission 2

Jana Sokol (09971444)

Christoph Winkler (11701292)

## Documentation:

### Implemented Compulsory Gameplay:

- **3D Geometry (6 Points):** Our game contains non-trivial 3D-Objects (Wall around the door, pressure plate, RePeter). All used objects were created using Blender and are being loaded into our project via the ASSIMP library. The file format of our choice is wavefront (.obj). The objects are being loaded using our `LoadModel()` function and are being rendered using the `RenderModel()` function.
- **Playable (3 Points):** Our game is playable. In contrast to our first submission, we now have our three planned levels implemented. Even though the cloning mechanics are still acting up a little bit sometimes, there is now a game mechanic in place to deal with such issues, so that it is now possible to play the whole game through, without having to restart the game.
- **Advanced Gameplay (3 Points):** As per the requirements, we added sophisticated gameplay. As mentioned above, the cloning mechanics, especially the handling of the RePeter has been updated. Also with the implementation of the additional levels, each level has now specific unique features (Lava, lights out).
- **Min. 60 FPS and Framerate Independence (3 Points):** There is now a `glfwWindowHint(GLFW_REFRESH_RATE)` in place, which specifies the minimum number of FPS. The variable can be set in the settings.ini file in the assets folder. Our testing has shown us a standard framerate of about 200+ FPS. The object movement is dependent upon a delta time variable.
- **Win/Lose Condition (3 Points):** As they say, winning is easy, losing is hard. Winning in each level means to open the door and pass onto the next level. Losing however, is intentionally level dependent. As the first level serves as a tutorial/introduction which is why we decided to leave losing here as “not being able to pass on”. The second level makes use of the more classic interpretation of losing, which is dying. In our case either by falling into the Lava or running into the Laser. The last level, as it is the lights out level, is hard enough as is, so the player plays against themselves using the timer as a win/lose condition.
- **Controls (2 Points):** The game is being played using the standard WASD-SPACE key configuration to move and jump, the mouse is used to control the camera and

our cloning mechanism can be triggered using the key “C”. Furthermore, as stated earlier, the “R” key has been implemented to reset the clones or start the level again after dying. It is also very useful to regain control over a rough clone if it should decide to act up. Last but not least you can use the “P” key to push boxes around the scene.

- Intuitive Camera (2 Points): The game uses a camera model to generate the correct transformation matrices. It can be controlled by using the mouse.
- Illumination Model (2 Points): There are point lights, spotlights and a directional Light implemented in the game, although spotlights have not made it onto the stage. As of now, the game supports two types of materials (shiny and dull). The textures are being displayed using the normal vectors which come with the models themselves.
- Textures (2 Points): All of our objects are textured. All but one (plain, default) textures come with the objects themselves.
- Moving Objects (2 Points): To complete a level, the player has to open the doors to the next room. The opening of the doors is done by moving the doors. The mechanic which opens the door, the RePeter, are also moving objects.
- Adjustable Parameters (1 Points): Screen settings (screen size, fullscreen, refresh rate etc.) are being read from an .ini file via the Inireader. Using this, these screen settings can be changed without recompiling the game.

## **Implemented Optional Gameplay:**

- Collision Detection (4 Points): As our physics engine of choice, we’re using PhysX. Every Object in our scene has a rigid body attached to it, including the player, which allows for game mechanics like gravity, collision detection, etc.
- Advanced Physics (6 Points): One of the core elements of our game is the pressure plate. In game they are used to open doors, control lasers and win the game. The pressure plates, as well as the Lava and the Laser are PhysX Trigger Objects, which set a callback whenever they are triggered to inform the game of what has happened. The Player Character itself is controlled not just by the player, but also by a PhysX Character Controller. This helps mostly with the interaction with other RePeter. Furthermore we simulated physics with pure dynamic objects (making no use of kinematic functions/bodies), which can be seen in Level two, where the stepping plate makes use of gravity and you are able to push a box using linear velocity.
- Heads-Up Display (4 Points): Our game makes use of HUD’s in two forms. The first one comes in form of the timer, which can be found at the right upper corner of the screen. The second one can be seen at the beginning of the first level as the tutorial. Here are again, two are implementations at play, the first one being in form of text on screen (Press T for tutorial) and the second one as 2D-billboards hovering over the pressure plates (Stand here to activate).

## Features:

The feature that makes up the whole concept behind the game is the cloning mechanic. It should be noted that the kind of cloning we chose to go for, was temporal cloning – meaning that each time the player decides to clone, it resets time. The current implementation of the cloning mechanic works on the premise of the observing and recording mode. The player starts out in the observing mode, where it can navigate the world freely without effecting the first clone. Pressing the “C” key will switch the player from observing to recording mode. In recording mode, the player starts out at the start point of the clone’s path and any further movement is being tracked as a path for the clone to follow. The clone mechanic is depended upon the timer in the upper right corner of the screen, as each cloning call will reset time, meaning set it back to zero. This is important when considering the clone’s lifetime as a clone will stay alive if the player keeps staying in recording mode. At the time when the player decides to press “C” again, the player is being switched back into observing mode and the clone starts to follow the path, the player has tracked. Pressing “C” again will start the recording for the second clone. If the player starts recording the second clone while the first clone is still alive, the player will be able to see and interact with clone. Up to 4 clones can be created this way. Should the player die along the way, it will be asked to press the “R” key to reset the current level. This will also delete all current clones and their paths. The function can also be called while the player is still alive, to help with deleting clones who seem to follow a path incorrectly. This way, the whole game can be mastered.

## Additional libraries:

- ASSIMP (<https://www.assimp.org/>) - for object loading
- FREETYPE (<https://www.freetype.org/index.html>) – for text on screen
- GLEW (<http://glew.sourceforge.net/>) - extension wrangler
- GLFW (<https://www.glfw.org/>) - for window creation
- GLM (<https://glm.g-truc.net/0.9.9/index.html>) - for mathematical calculations (vectors, world transformations via matrices, etc.)
- PHYSX (<https://developer.nvidia.com/physx-sdk>) - for in game physics (collision detection, usage of trigger, character controlling etc.)

## Implemented Effects:

- Lighting: Shadow Map with PCF (16 Points): In the game shadows are being displayed using shadow maps. For example shadow acne is being handled by using a bias and shadow edges are being smoothed out by using pcf (see `shader.frag CalcDirectionalShadowFactor()`).
- CPU Particle System (8 Points): The main resource for the implementation was the [opengl-tutorial.org: Particles/Instancing](http://opengl-tutorial.org/Particles/Instancing). The particle system can be found in the class particles and is implemented in a way, that provides the ability to create multiple different particle systems with varying textures, sizes, positions, speed and lifetime. The most important components of the system are the methods `Init()`, `FindUnusedParticle()`, `SortParticles()` and `RenderParticle()`. Particle systems are on every pressure plate and on the lava.
- Video Texture (8 Points): For the implementation of this effect no resources were used. We first generated a video of the laser of level two in Adobe Animate by designing a grid and spinning it by 90 degrees. We exported this video as a png

sequence with 72 frames with a showing speed of 24 frames per second in mind. The png sequence is being loaded into the engine as level two is being generated. The implementation can be found in the main.cpp file. The framerate of the game is kept independent of the framerate of the video by multiplying the deltatime with the increment of the loop that chooses the next picture.

- Simple Normal Mapping (4 Points): For the implementation of this effect the resource [learnopengl.com: Normal Mapping](https://learnopengl.com/Normal%20Mapping) was used. A normal map of the door texture was generated with the online tool <https://cpetry.github.io/NormalMap-Online/>. In accordance with the tutorial the normal map is being loaded into the fragment shader (shader.frag) where the normals are being chosen from the normal map instead of the object. The effect can be toggled with the “N” key. The effect can be seen on doors.
- Bloom/Glow (8 Points): The implementation on bloom was done with the help of [learnopengl.com: Bloom](https://learnopengl.com/Bloom). The code can be found in the class Bloom.h and Bloom.cpp. We updated our shaders for bloom to work properly so that they write into two color buffers so we can use their information for the blurring effect. The blurred fragments are chosen via a threshold in the fragment shader. The bloom effect can be seen around the lights on the ceiling and can be toggled via the “H” key. To help with bloom, HDR has been implemented to enable more intense lights.

### Step by step instruction:

- To pass the first level of the game, the player must activate one pressure plate via clone and the other one itself. A pressure plate is active when a player or RePeter stands on it. The player starts by pressing “C” to switch into recording mode and walks towards a chosen pressure plate. After a short wait, to make sure both pressure plates are activated long enough for the door to open, the player can hit “C” again to switch back to observing mode, and head for the other pressure plate. The clone will follow its path, both pressure plates will be activated, the door opens, and the level is complete.
- The second level requires a bit more steps. First the player has to push the box to the edge of the lava pool and switch in to recording mode to record a path jumping onto it. Then the player must proceed to the stepping plate that fell from the ceiling. From this height the player is able to jump on top of the first RePeter and onto the second floor. On the top floor the player activates the pressure plate. Once this is done, a second pressure Plate will appear on the ground floor. Pressing “C” will start a new recording for a clone to remain standing on it, for as long as this one is active, the laser on the upper floor will remain turned off. Switching back into observing mode, the first clone will again take its position as a platform in between and the second one will activate the ground floor pressure plate. All that is left to do, is to make it to the upper floor and through the door before the clones die again.
- Level three is the last level and follows a parkour style. After 5 seconds the lights get turned off and the player must navigate the parkour to the pressure plates with the help of the particles.