

# Subway Escape Documentation

## Gameplay

Subway Escape is an escape/puzzle game. You are somehow locked in a tunnel system underground, where you have to find your way out, back to the surface. Because of many riddles a full walkthrough is provided in a second PDF.

## Developer Tools

To switch between different tunnels without actually playing the game you can use the “tab” and “backspace” keys. *Attention:* This was simply implemented for easier navigation through the tunnels while testing the gameplay, this feature sometimes (pretty often) throws exceptions if not used correctly.

To find out in which tunnel you are currently in hit “t” – it will print out a specific number in the console.

## Effects

We implemented three effects from the provided effect-page: Shadow mapping (1.5), Portal Rendering (2) and Spotlights (0.5).

For *shadow mapping* the following tutorials were used:

<https://www.youtube.com/watch?v=o6zDfDkOFIc>

<https://www.youtube.com/watch?v=9sEHkT7N7RM>

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/>

<http://www.learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping>

As an extension, shadow mapping for multiple light sources where added. When only stepping through the tunnels with “tab”, shadow mapping can be seen in tunnel 18.

There is also a moving train that throws shadow. To see that, do the following: Use “tab” to go to tunnel 4, take the door at the end of the tunnel (the right one, if there are already multiple visible). You should be able to see the train now. The train is also animated. (see Animated Objects)

For the *portal rendering* following tutorials were used:

<https://th0mas.nl/2013/05/19/rendering-recursive-portals-with-opengl/>

[https://en.wikibooks.org/wiki/OpenGL\\_Programming/Mini-Portal](https://en.wikibooks.org/wiki/OpenGL_Programming/Mini-Portal)

The portal rendering includes an oblique clipping in order to prevent drawing objects that are behind the portal: <http://www.terathon.com/lengyel/Lengyel-Oblique.pdf>

Once the player clicks on a normal door, the door has been opened and you can see and also walk through the portal. The two tunnels are not linked together in the game world. This is visible when multiple doors on the same wall have been opened.

For *spot lights* the following tutorial was used:

<http://www.lighthouse3d.com/tutorials/gls-tutorial/spotlights/>

Spot lights can be seen in every tunnel.

### **Complex Objects**

Objects are loaded with assimp. Following tutorial was used in order to implement the loader: <http://www.learnopengl.com/#!Model-Loading/Assimp>

Some models were from opengamearts.org. Other models are drawn by us. We used blender for that.

### **Animated Objects**

Hierarchical animation can be found on every train. The train itself is moving back and forward in the tunnel. Take a closer look at the wheels and you will notice that they are moving too, but also rotate. In order to see the train without playing the game, use following steps:

Use “tab” to go to tunnel 4, take the door at the end of the tunnel (the right one, if there are already multiple visible). You should be able to see the train now.

### **View-Frustum-Culling**

View Frustum Culling was implemented with the help of the following tutorial: <http://www.lighthouse3d.com/tutorials/view-frustum-culling/> and the use of AABB. Since we use separate tunnels, low poly models as well as vSync no performance improvement could be seen. However the console will output how many objects were culled.

### **Transparency**

Half-transparency can be seen in tunnel 5 in the colored view. In colored view a “glass wall” should prevent the user from taking the key.

### **Libraries**

SDL: <https://www.libsdl.org/>

SDL\_image: [https://www.libsdl.org/projects/SDL\\_image/](https://www.libsdl.org/projects/SDL_image/)

Assimp: <http://assimp.sourceforge.net/>

GLM: <http://glm.g-truc.net/0.9.7/index.html>

Glew: <http://glew.sourceforge.net/>

Bullet: <http://bulletphysics.org/wordpress/>