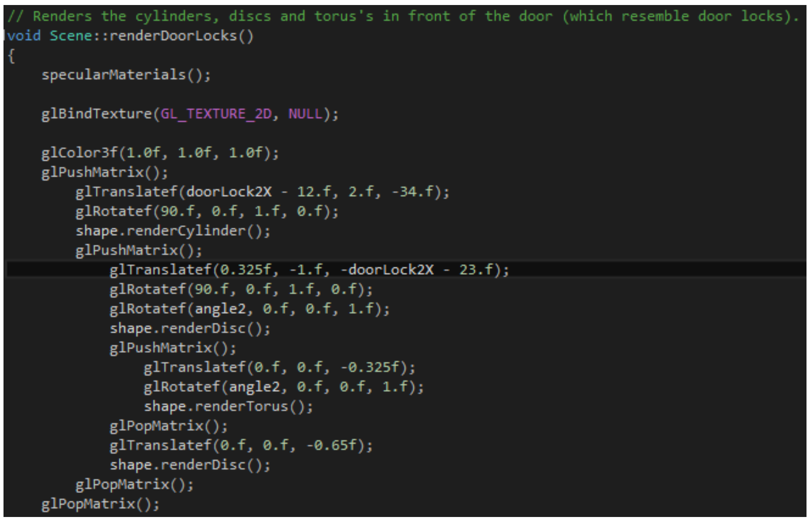
CMP203 – Submission

Fraser Barker - (1600196)

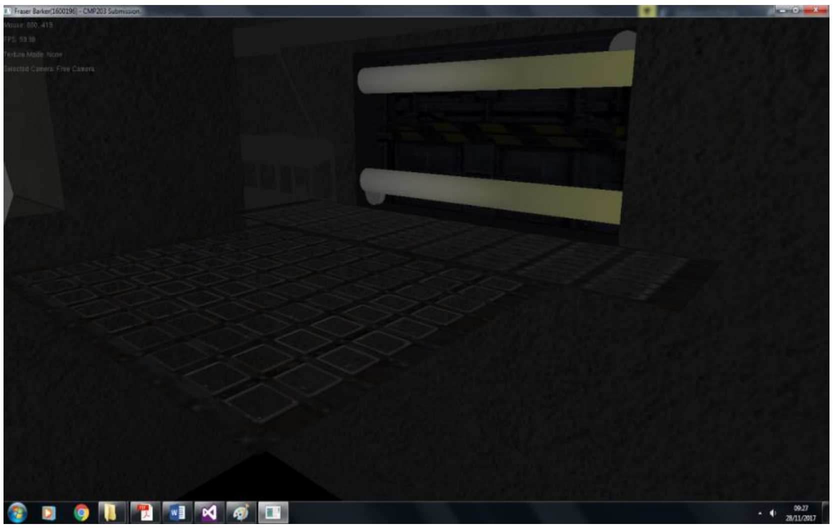
- Controls.
    - **1** – **4** to switch between various texture filtering modes.
        - **1** – Point Sampling
        - **2** – Bilinear
        - **3** – Trilinear
        - **4** – Point on near/Trilinear on far
    - **7** – **9** to switch between various cameras within the scene.
        - **7** – Default camera "Free Camera".
            - Can move via **W**,**A**,**S**,**D** keys, up with **Spacebar**, down with **C**.
            - Can rotate via the mouse.
        - **8** – Tram camera
            - Can rotate via the mouse.
        - **9** – Door camera
            - No movement/rotation allowed.
    - **W, A, S, D, Spacebar, C** are used to move the camera around the scene.
        - **W** – Moves the Free Camera forwards.
        - **S** – Moves the Free Camera backwards.
        - **A** – Moves the Free Camera left.
        - **D** – Moves the Free Camera right.
        - **Spacebar** – Moves the Free Camera up the Y-axis, rotation independent.
        - **C** – Moves the Free Camera down the Y-axis, rotation independent.
    - **I** and **K** to move the tram backwards and forwards within the scene.
        - **I** – Moves the tram in the negative X direction (left).
        - **K** – Moves the tram in the positive X direction (right).
    - **Q** and **E** to open/close the door in the scene.
        - **Q –** Opens the door once the door locks are retracted.
        - **E** – Closes the door.
    - **Mouse** movement is used to rotate a camera if it's designed to rotate.
    - **F** allows the user to turn wireframe mode on/off.
    - **R** resets every aspect of the scene.
- How I met each aspect of the coursework brief.
    - The scene must show lighting from multiple lights of different types, colours and some animated.
        - I have **7** lights within my scene. 2 are above the door, one on each side of the tram, one in each tram dock and another to give the scene a little light.
        - The two lights above the door are **spot** lights, which rotate and become enabled in tandem with the door opening (i.e. when the user presses and holds **E** or **Q**). They are both **red** in colour.
        - The lights on either side of the tram are also **spot** lights and are enabled when the user moves the tram (via pressing **I** or **K**). They are both **white** in colour.
        - The lights within each tram dock are **point** lights. They are always enabled. They are both **yellow** in colour.
        - The scene light is a **point** light. It is always enabled. It is **white** in colour.

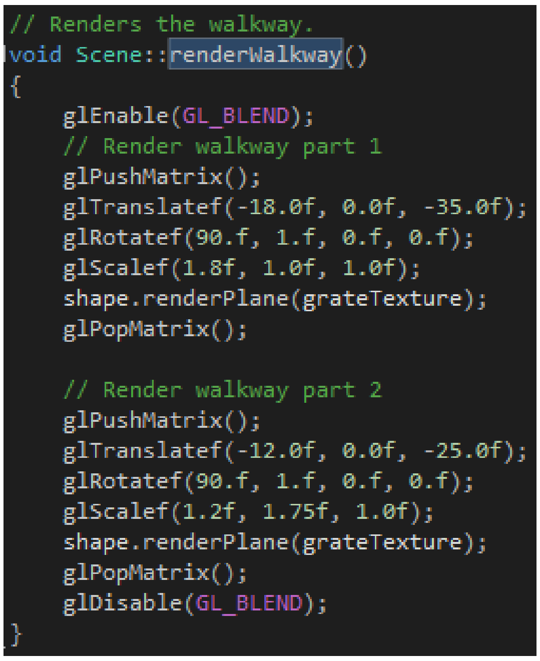
- The scene must show use of texturing. Additionally, demonstrating texture filtering.
  - I have **5** (technically 7) **different textures** within my scene.
    - **Hazard Texture** – Is the texture I apply to the tram rail.
    - **Wall Texture** – Is the texture I apply to each wall.
    - **Grate Texture** – Is the texture I apply to both parts of the walkway and also where I utilise some transparency/alpha blending effects.
    - **Door Top Texture** – Is the texture I apply to the top of the door. (There is also a flipped version of this I apply to the other side of the door.)
    - **Door Bottom Texture** – Is the texture I apply to the bottom of the door. (There is also a flipped version of this I apply to the other side of the door.)
  - I allow the user to switch between various texture filtering modes via the keys **1** – **4**. It's not very noticeable on the textures I have chosen however if you look closely at any one of the walls you will notice a slight difference between each selected texture filtering mode.
    - **1** – Point Sampling
    - **2** – Bilinear
    - **3** – Trilinear
    - **4** – Point on near/Trilinear on far
- A working camera. The user must be able to manipulate the view through using the mouse and keyboard to control the camera. Additionally, you should provide multiple cameras each with a different focus such as limited controls, fixed views, procedurally controlled views or different camera types.
  - I have **3** different cameras within my scene.
    - 1 – **Free Camera**. This is the default camera selected when the scene starts. The user can fly around the scene using the WASD, Spacebar and C keys. Rotation is controlled via the mouse.
    - 2 – **Tram Camera**. This is a camera which position is bound to the location of the tram (i.e. It can only be moved via **I** or **K** being pressed). It can be rotated via the mouse.
    - 3 – **Door Camera**. This camera can neither move nor rotate. It is purely a static camera which is focussed on looking into the reflective plane behind the door.
- A clear example of using the matrix stack for Hierarchical modelling and animation through hierarchical means.
  - I utilise the matrix stack in various places throughout my scene. A clear example of using it for hierarchical modelling/animation would be when I render my door locks as I render three shapes and translate/rotate them by different values.



```
// Renders the cylinders, discs and torus's in front of the door (which resemble door locks).
void Scene::renderDoorLocks()
{
    specularMaterials();

    glBindTexture(GL_TEXTURE_2D, NULL);

    glColor3f(1.0f, 1.0f, 1.0f);
    glPushMatrix();
        glTranslatef(doorLock2X - 12.f, 2.f, -34.f);
        glRotatef(90.f, 0.f, 1.f, 0.f);
        shape.renderCylinder();
        glPushMatrix();
            glTranslatef(0.325f, -1.f, -doorLock2X - 23.f);
            glRotatef(90.f, 0.f, 1.f, 0.f);
            glRotatef(angle2, 0.f, 0.f, 1.f);
            shape.renderDisc();
            glPushMatrix();
                glTranslatef(0.f, 0.f, -0.325f);
                glRotatef(angle2, 0.f, 0.f, 1.f);
                shape.renderTorus();
            glPopMatrix();
            glTranslatef(0.f, 0.f, -0.65f);
            shape.renderDisc();
        glPopMatrix();
    glPopMatrix();
```

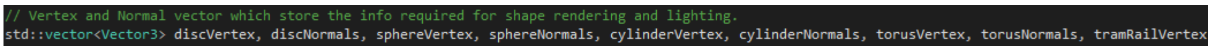
- Use of transparency effects / Alpha blending.
  - I use a transparency effect/alpha blending on my walkway within the scene. The reason they blend/are transparent is because I render the walls first so that they have something to blend against.


Fraser Barker(1600196) - CMP203 Submission.
Mouse: 800 ,419
FPS: 59.98
Texture Mode: None
Selected Camera: Free Camera



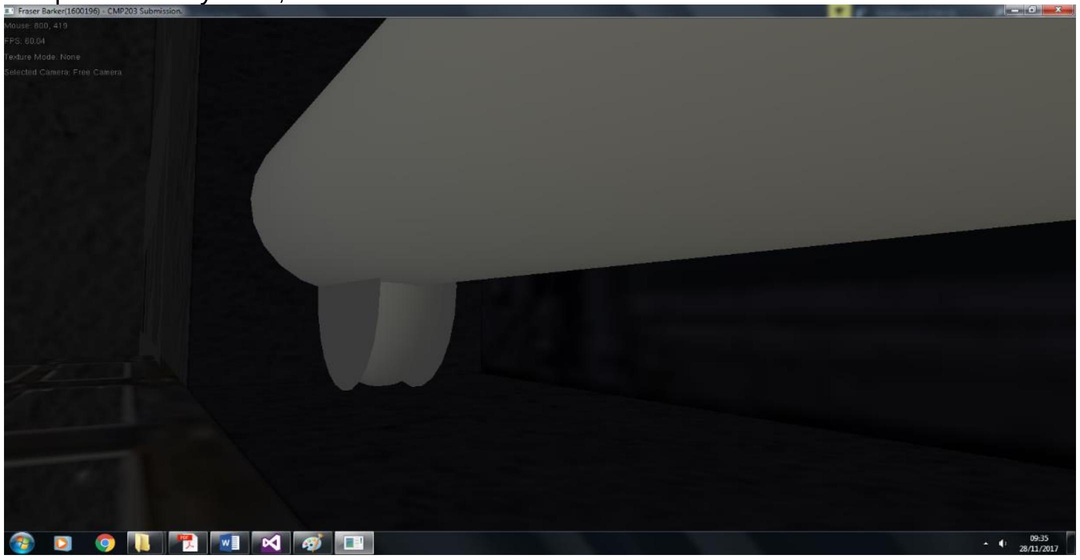
```
// Renders the walkway.
void Scene::renderWalkway()
{
    glEnable(GL_BLEND);
    // Render walkway part 1
    glPushMatrix();
    glTranslatef(-18.0f, 0.0f, -35.0f);
    glRotatef(90.f, 1.f, 0.f, 0.f);
    glScalef(1.8f, 1.0f, 1.0f);
    shape.renderPlane(grateTexture);
    glPopMatrix();

    // Render walkway part 2
    glPushMatrix();
    glTranslatef(-12.0f, 0.0f, -25.0f);
    glRotatef(90.f, 1.f, 0.f, 0.f);
    glScalef(1.2f, 1.75f, 1.0f);
    shape.renderPlane(grateTexture);
    glPopMatrix();
    glDisable(GL_BLEND);
}
```

- Use vertex arrays (not including model loading).
    - I utilise vector<Vector3> to store my procedurally generated shapes within the scene.



```
// Vertex and Normal vector which store the info required for shape rendering and lighting.
std::vector<Vector3> discVertex, discNormals, sphereVertex, sphereNormals, cylinderVertex, cylinderNormals, torusVertex, torusNormals, tramRailVertex
```

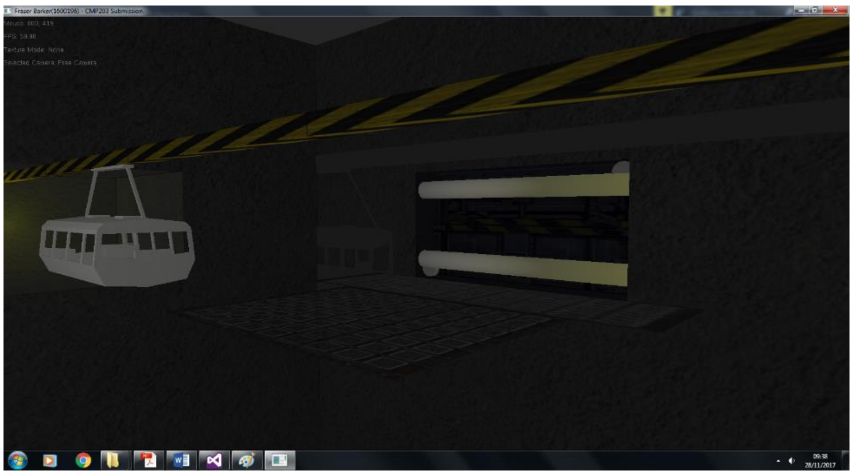
- Models loaded in from an external file.
    - I load two models in from an external file
        - 1 – The tram within the scene.
        - 2 – The crowbar in the door room.
- Examples of Procedurally generated shapes.
    - I procedurally generate the door locks which are in front of the door. They are composed of a cylinder, torus and two discs.



Fraser Barker(1600196) - CMP203 Submission

```
Mouse: 800, 419
FPS: 60.04
Texture Mode: None
Selected Camera: Free Camera
```

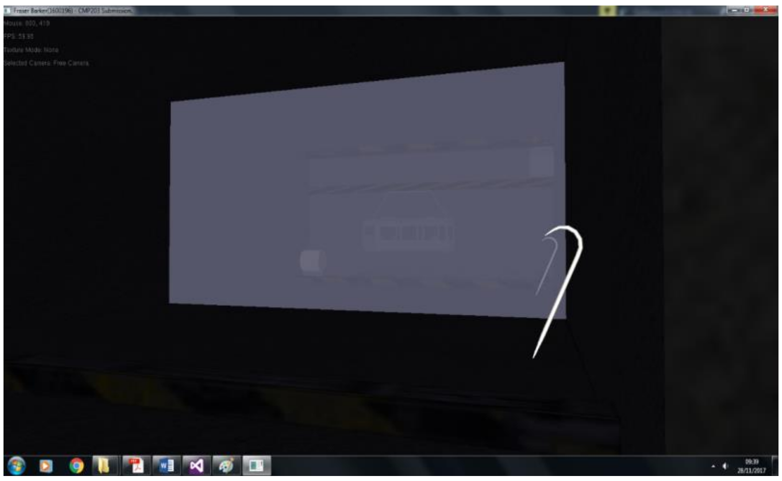
- User interaction (controlling objects in the scene other than the camera).
    - The user can control the tram (**I** or **K**).
    - The user can open/close the door (**Q** or **E**).
- A wireframe mode.
    - Pressing the **F** button allows the user to turn wireframe mode on/off.
- Advance features such as shadows and use of the stencil buffer.
    - I incorporate both shadows and reflection utilising the stencil buffer within my scene.
        - The tram and railway have a shadow.



Fraser Barker(1600196) - CMP203 Submission

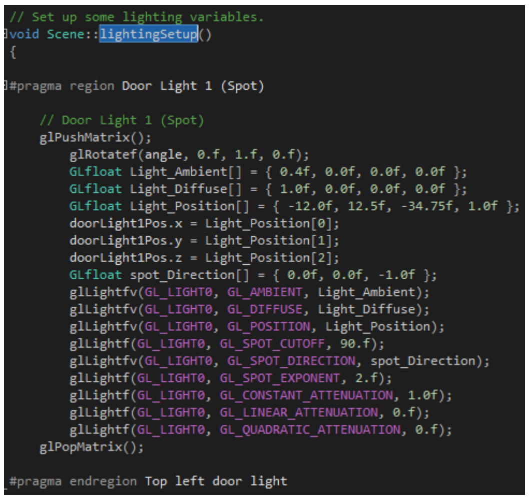
```
Mouse: 800, 419
FPS: 59.98
Texture Mode: None
Selected Camera: Free Camera
```

- The reflective plane in the door room behind the door reflects **7** objects.



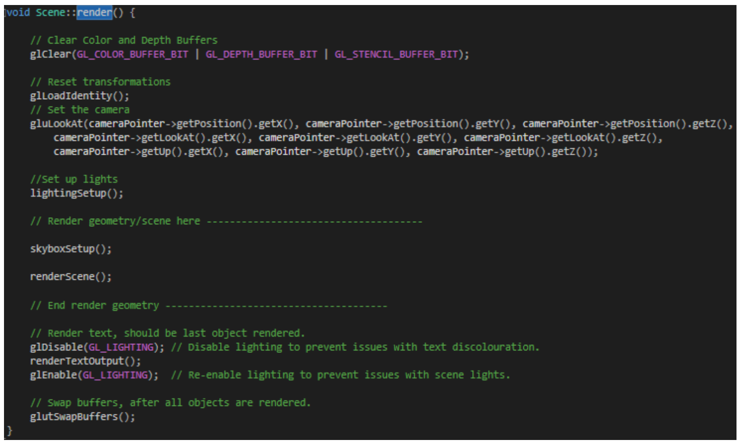
```
Fraser Barker(1600196) - CMP203 Submission
Mouse: 800, 419
FPS: 59.95
Texture Mode: None
Selected Camera: Free Camera
```

- The application should be carefully designed and constructed showing appropriate use of classes and well commented.
    - I comment and utilise tidying functions to make navigation around my program much easier. A function such as lightingSetup helps to keep the render function clearer as otherwise there would be values for 7 different lights in the render function.



```
// Set up some lighting variables.
void Scene::lightingSetup()
{

#pragma region Door Light 1 (Spot)

    // Door Light 1 (Spot)
    glPushMatrix();
        glRotatef(angle, 0.f, 1.f, 0.f);
        GLfloat Light_Ambient[] = { 0.4f, 0.0f, 0.0f, 0.0f };
        GLfloat Light_Diffuse[] = { 1.0f, 0.0f, 0.0f, 0.0f };
        GLfloat Light_Position[] = { -12.0f, 12.5f, -34.75f, 1.0f };
        doorLight1Pos.x = Light_Position[0];
        doorLight1Pos.y = Light_Position[1];
        doorLight1Pos.z = Light_Position[2];
        GLfloat spot_Direction[] = { 0.0f, 0.0f, -1.0f };
        glLightfv(GL_LIGHT0, GL_AMBIENT, Light_Ambient);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, Light_Diffuse);
        glLightfv(GL_LIGHT0, GL_POSITION, Light_Position);
        glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 90.f);
        glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_Direction);
        glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 2.f);
        glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0f);
        glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.f);
        glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.f);
    glPopMatrix();

#pragma endregion Top left door light
```

    - The end result.



```
void Scene::render() {

    // Clear Color and Depth Buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);

    // Reset transformations
    glLoadIdentity();
    // Set the camera
    gluLookAt(cameraPointer->getPosition().getX(), cameraPointer->getPosition().getY(), cameraPointer->getPosition().getZ(),
        cameraPointer->getLookAt().getX(), cameraPointer->getLookAt().getY(), cameraPointer->getLookAt().getZ(),
        cameraPointer->getUp().getX(), cameraPointer->getUp().getY(), cameraPointer->getUp().getZ());

    //Set up lights
    lightingSetup();

    // Render geometry/scene here -------------------------------------

    skyboxSetup();

    renderScene();

    // End render geometry --------------------------------------

    // Render text, should be last object rendered.
    glDisable(GL_LIGHTING); // Disable lighting to prevent issues with text discolouration.
    renderTextOutput();
    glEnable(GL_LIGHTING);  // Re-enable lighting to prevent issues with scene lights.

    // Swap buffers, after all objects are rendered.
    glutSwapBuffers();
}
```

- References:
    - Helped me to utilise the stencil buffer to cull the shadow once it had gone over the plane it was projected onto boundaries: http://artis.imag.fr/Recherche/RealTimeShadows/pdf/stencil.pdf
    - For the crowbar model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/622279
    - Gave me some insight into loading .mtl files although I have yet to have it fully implemented: https://xiangchen.wordpress.com/2010/05/04/loading-a-obj-file-in-opengl/ & https://www.youtube.com/watch?v=yfsVBh2AaA8
    - For the tram model: https://3dwarehouse.sketchup.com/model/fce48e6cd0f81c0c873433210e278bfe/Half-Life-Tram
    - HL Textures: http://25.media.tumblr.com/240d2e8c8ecc6e97807ffbfe06be1b28/tumblr_mwmqccBQJ81s1rufio1_1280.jpg
    - Grate Texture: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR10ZPSaY8cjLKK_Ek2QGGTfswKkg1YJOK-gnZ6edek_-Q_8QqC
    - Wall Texture: http://media.moddb.com/images/articles/1/103/102706/auto/dt_brick_cr.jpg