CSU44052 Final Project Report

Name:	Merlin Prasad
Student ID:	19333557
Declaration:	I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: http://www.tcd.ie/calendar I have also completed the Online Tutorial on avoiding plagiarism 'Ready, Steady, Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write **Signed:** Merlin
Youtube link 1: Required Features **Google drive link 2:** Required Features:	https://youtu.be/pG-nE3JvJuE https://drive.google.com/file/d/1Il2MHtBylhJeOaq4Z_9U1XJENOK-iJy8/view?usp=sharing
Youtube link 2: Final Demo **Google drive link 2:** Final Demo	https://youtu.be/KPW7YEITzYw https://drive.google.com/file/d/1162c5MckfK7Pz1D8znPXK0ubhXNZthzQ/view?usp=sharing

Required feature 1: crowd of animated snow-people/reindeer/robins etc.

Screenshot(s) of feature:

Crowd of hierarchical snow-people

Describe how you implemented it: I implemented a crowd of animated snowmen. Each snowman is a hierarchical character making this also an advanced feature. First I loaded in a snowman base model and 2 arm models so I could move them later. There is a main snowman model that acts as the parent object. It translates up and down across the scene based on a cos wave of the time and has 2 child arm objects connected to it that rotate. These arms make the snowman appear like it is waving. The rest of the crowd of snowmen are children to this main parent snowman. This hierarchy means all the snowmen

move up and down following the main snowman model in a realistic manner. Child snowmans arms also rotate like the main snowmans arms so all the snowmen are waving in this hierarchical animated crowd.

Pseudocode:

```
//the main snowman body that has its translation for moving up and down the scene applied to it
lightingShader.setMat4("model", modelBody);
snowManBasic.Draw(lightingShader);
//right arm has rotations applied to it already moves along with the body in hierarchy
lightingShader.setMat4("model", modelBody* rightArm);
armRight.Draw(lightingShader);
Animating left arm is similar to right arm code

//Animating hierarchical child snowmen that follow the parent snowman
lightingShader.setMat4("model", modelBody * modelSnowman2);
snowManBasic.Draw(lightingShader);
lightingShader.setMat4("model", modelBody * modelSnowman2* leftArm2 * rightArm);
armLeft.Draw(lightingShader);
Animating left arm is similar to right arm code

Similar steps for the rest of the child snowmen in crowd to move in the hierarchy
```

Credits (e.g., list source of any tools, libraries, assets used):
Assimp (model loading library) : https://github.com/assimp/assimp/blob/master/Build.md
Snowman model: https://sketchfab.com/3d-models/matt-the-snowman-f769c1ea1ce84566a3a9c0b5f4388c0f

Required feature 2: texture-mapping your scene and creatures using an image file

Screenshot(s) of feature:

Snowman texture image and snowman with texture mapped

Tree textures and tree

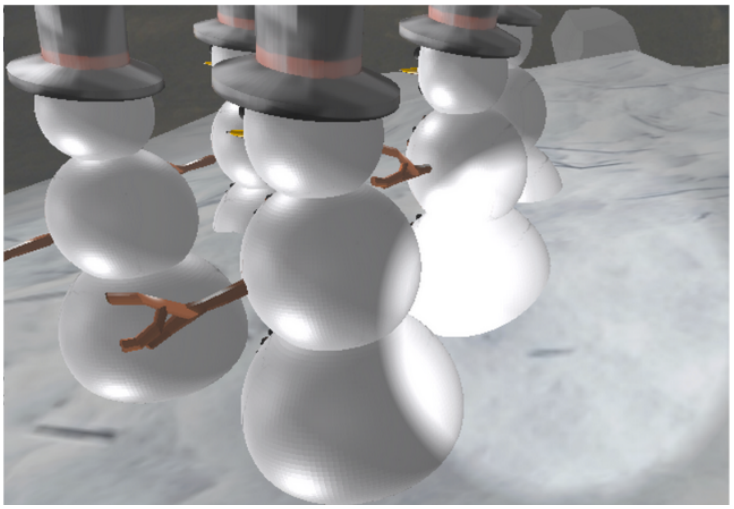
Describe how you implemented it: To implement textures I followed the learnopengl tutorial to load a model with assimp and stb.image library by creating a Model class. I passed the texture coordinates from vertex shader to fragment shader so that lighting and other changes were applied correctly. On blender I attached image files to colour my objects in the shading tab. All the models in this scene are texture mapped with images I found online or made from scratch on blender. These textures like for the snowmen and the foxes were handmade by me using texture paint. Finally I exported my models from Blender as an obj file with an accompanying .mtl that linked to the textures which I loaded in .

Credits (e.g., list source of any tools, libraries, assets used):
Image loading library stb : https://github.com/nothings/stb/blob/master/stb_image.h
Tree trunk texture : https://unsplash.com/photos/4u629vFD3vI
Tree leaves texture : https://unsplash.com/photos/K0wseIXrs3I
Snow grass texture : https://opengameart.org/node/34741
Snow texture : https://opengameart.org/content/seamless-snow-texture-0
Model loading class tutorial : https://learnopengl.com/Model-Loading/Model
Texture tutorial: https://learnopengl.com/Getting-started/Textures
Blender to hand make the snowman texture,fox and all other textures for scene

Required feature 3: implementation of the Phong Illumination model

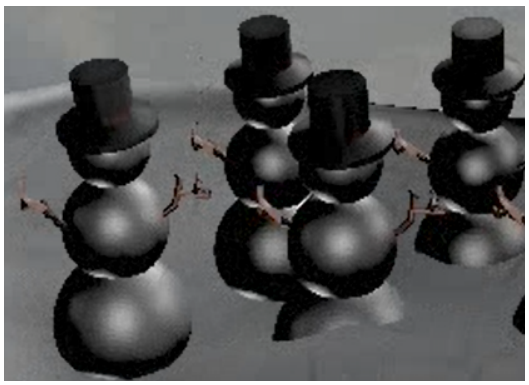
a. Multiple light sources (at least 2, can be point, directional, or spotlight)
b. Multiple different material properties (at least 5 on 5 different objects)
c. Normal must be transformed correctly
d. Shading must use a combination of ambient, diffuse, and specular lighting

Screenshot(s) of feature:



Specular highlights of the point, directional light and spotlight visible

Material properties visible on the present models (Silver,ruby,gold,emerald and bronze)



Low Ambient Value

Low Diffuse Value

Low Specular Value

High Ambient Value

High Diffuse Value

High Specular Value

Describe how you implemented it: I implemented 3 types of lighting (directional,point and spotlight). The directional light was added to make the dark scene brighter. 4 point lights were dotted in the sky to represent the stars and a spotlight was added to traverse the foggy scene. I followed learnopengl tutorial to make edits to the fragment shader. The fragment shader takes in the material properties of the object which can be manually specified which I did to show the 5 different material properties on the present objects. (ruby,gold,emerald,silver and bronze) or as image files with specular and diffuse values mapped on it. It also takes in the light position, ambient, diffuse and specular properties.I calculated the ambient, diffuse and specular contributions of each fragment and also used the normalize function to make sure normals are transformed correctly. I then added ambient, diffuse and specular values together to make a result which I multiplied by the texture to figure out the fragment colour in the shader.

The lighting my scene has implemented is also Blinn-Phong lighting as it looks more natural. To demo the changes of what happens when ambient,diffuse and specular terms are changed I mapped these variables to the keyboard as follows.The I key increases ambient value while J decreases it. O increases diffuse value while K decreases and P increases specular value while L decreases it.

Pseudocode:

Code for blinn-phong directional lighting*

```
// ambient
vec3 ambient = light.ambient * material.ambient;

// diffuse
vec3 norm = normalize(normal);
vec3 lightDir = normalize(light.pos - fragPos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = light.diffuse * (diff * material.diffuse);
```

```
// specular
vec3 viewDir = normalize(viewPos - fragPos);
//blinn phong
vec3 halfwayDir = normalize(lightDir + viewDir);
float spec = pow(max(dot(norm, halfwayDir), 0.0), material.shininess);
vec3 specular = light.specular * (spec * material.specular);
vec3 result = (ambient + diffuse + specular) ;

//direct light
fragColour = texture(texture_diffuse1, texCoord) * vec4(result, 1.0) ;
```

**Note :* Specific lighting code for spotlight etc can be found in manyLights.fs

Code for specifying ruby material property on presents:

```
//ruby
matShader.setVec3("material.ambient", 0.1745f, 0.01175f, 0.01175f);
matShader.setVec3("material.diffuse", 0.61424f, 0.04136f, 0.04136f);
matShader.setVec3("material.specular", 0.727811f, 0.626959f, 0.626959f);
matShader.setFloat("material.shininess", 0.6f);
matShader.setMat4("model", modelPrez2);
prez.Draw(matShader);
```

Credits (e.g., list source of any tools, libraries, assets used):
Material properties reference sheet : http://devernay.free.fr/cours/opengl/materials.html
Lighting tutorial : https://learnopengl.com/Lighting/Multiple-lights

Advanced feature 1

Description/name of feature: Exponential squared fog

Screenshot(s) of feature:

Fog is making trees and house models obscured in fog as they are further away from camera

Describe how you implemented it: Objects further away from the camera are not shown as clearly as objects nearby to demonstrate fog .Instead of using the basic linear fog implementation which has a min/ max fog value I did an squared exponential fog version as it looked more realistic in opengl.I also connected it to the F key on the keyboard so you can easily enable and disable it.

Pseudocode to calculate exponential squared fog:

```
//distance from pixel to camera
float dist = length(fragPos.xyz - viewPos.xyz);
//exponential squared fog
float distRatio = 4.0 * dist/fogMax ;
float fogFactor = exp(-distRatio * fogDensity * distRatio * fogDensity);
fragColor = mix(fogColor, fragColor, fogFactor);
```

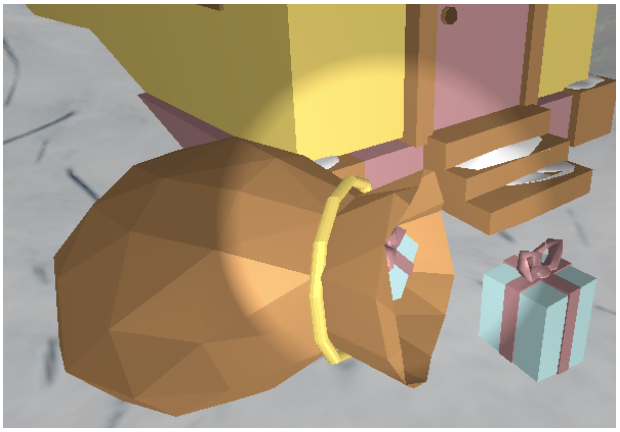
Credits (e.g., list source of any tools, libraries, assets used):
Exponential squared fog logic source : https://www.youtube.com/watch?v=oQksg57qsRA

Advanced feature 2

Description/name of feature: Great Handmade models

Screenshot(s) of feature:

Terrain with handmade models (ground.obj)



Presents sack

Presents wrapped in ribbon

House front view

House side view

Low poly fox

Forest

Describe how you implemented it: I used blender to hand make all the models for the scene that wasn't the snowman. The tree was made by editing multiple cylinder objects together. The house involved a combination of cubes and planes that I scaled and extruded . The present is made of a cube and curved objects for the bow. I manipulated a cylinder for the sack. The boulder is a cube that I cut up using the bisect tool until it looked right. I used a reference image for the fox model and mapped a plane mesh to fit it. I used the real snow add-on in blender to add snow to my models for realism. Finally I coloured all my models using image textures I mostly made myself using blender such as for the house model or presents model.

Credits (e.g., list source of any tools, libraries, assets used):
Blender to create models : https://www.blender.org/
Real snow-add on Blender : https://docs.blender.org/manual/en/latest/addons/object/real_snow.html
Fox reference : https://www.shutterstock.com/image-photo/side-view-red-fox-looking-two-2187373047
Youtube videos to help make my models:
Basics of blender : *https://www.youtube.com/watch?v=VYN9g-U7uco*
Animals in blender: https://www.youtube.com/watch?v=6mT4XFJYq-4
Texture sources for any non hand-made texture assets were already listed in the texture feature above.

Advanced feature 3

Description/name of feature: Music

Describe how you implemented it: I used the irrklang audio library to load background music and sound effects into my program. I added some positional music to the scene as well for the sound of snowmen walking on snow and birds chirping in the forest. This sound plays based on the listeners position which is the camera so the closer you are to the sound position the louder it gets. Finally I implemented sound effects like when I press the “F” key on the keypad for fog a beep sound plays.

Pseudocode:

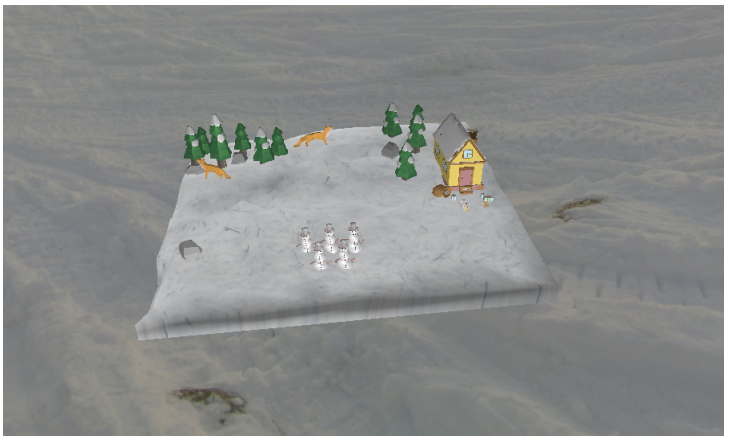
```
//positional crowd moving through snow sound
ISound* snowSound = musicEngine->play3D("music/snow.mp3", crowdPosition,true);
if (snowSound){
snowSound->setVolume(.006f);
snowSound->setMinDistance(0.05f);
}
//set listeners position to camera position
musicEngine->setListenerPosition(vec3df(camera.Positions), vec3df(0, 0, 1));
```

Credits (e.g., list source of any tools, libraries, assets used):
Irrklang (audio loading library) : https://www.ambiera.com/irrklang/downloads.html
Morning music : https://www.fesliyanstudios.com/royalty-free-music/download/morning-magic/423
Button sound effect : https://pixabay.com/sound-effects/beep-6-96243/
Walking on snow sound effect:
https://pixabay.com/sound-effects/people-walking-in-the-snow-footsteps-in-the-snow-in-the-snow-in-the-park-16589/
Birds chirping sound:https://pixabay.com/sound-effects/birds-in-the-morning-24147/

Advanced feature 4

Description/name of feature: Skybox

Screenshot(s) of feature:



Skybox bottom face viewed from above

Skybox top face viewed from above

Skybox front face

Skybox back face

Describe how you implemented it: I used a cubemap to make the skybox by using 6 separate textures images of a snowy scene and then combining them by following the learnopengl tutorial. I created skybox shaders that contained a samplerCube uniform to colour the scene correctly.

Pseudocode:

```
for (i = 0 ; i < 6; i ++ ){
    generate texture for each face of the cubemap using the stbi_load and the loaded in skybox pics
}
Wrap and filter methods
```

Credits (e.g., list source of any tools, libraries, assets used):
Skybox image : https://www.humus.name/index.php?page=Textures&ID=72
Cubemap guide : https://learnopengl.com/Advanced-OpenGL/Cubemaps

Advanced feature 5

Description/name of feature: Advanced Lighting Effects (Blinn-Phong Lighting and Gamma Correction

Screenshot(s) of feature:

Gamma correction enabled

Gamma correction disabled

Blinn-Phong Lighting on Snowmen

Blinn-Phong on House

Describe how you implemented it: I edited the lighting shader to change how the specular value is calculated based on learnopengl. I enabled gamma correction by enabling GL_FRAMEBUFFER_SRGB. To contrast between the difference of when gamma correction is enabled and disabled I mapped it to the F key along with fog. This way I have both a late evening with fog scene and a bright early morning with no fog scene.

Pseudocode:

```
//enable gamma correction disable for a darker scene
        if (!fog){
        glEnable(GL_FRAMEBUFFER_SRGB);
        }
        else {
        glDisable(GL_FRAMEBUFFER_SRGB);
        }
```

Credits (e.g., list source of any tools, libraries, assets used):https://learnopengl.com/Advanced-Lighting/Advanced-Lighting

Advanced feature 6

Description/name of feature: Hierarchical animated crowd that moves realistically which is already detailed in the required features 1 crowd of animated snow-people.