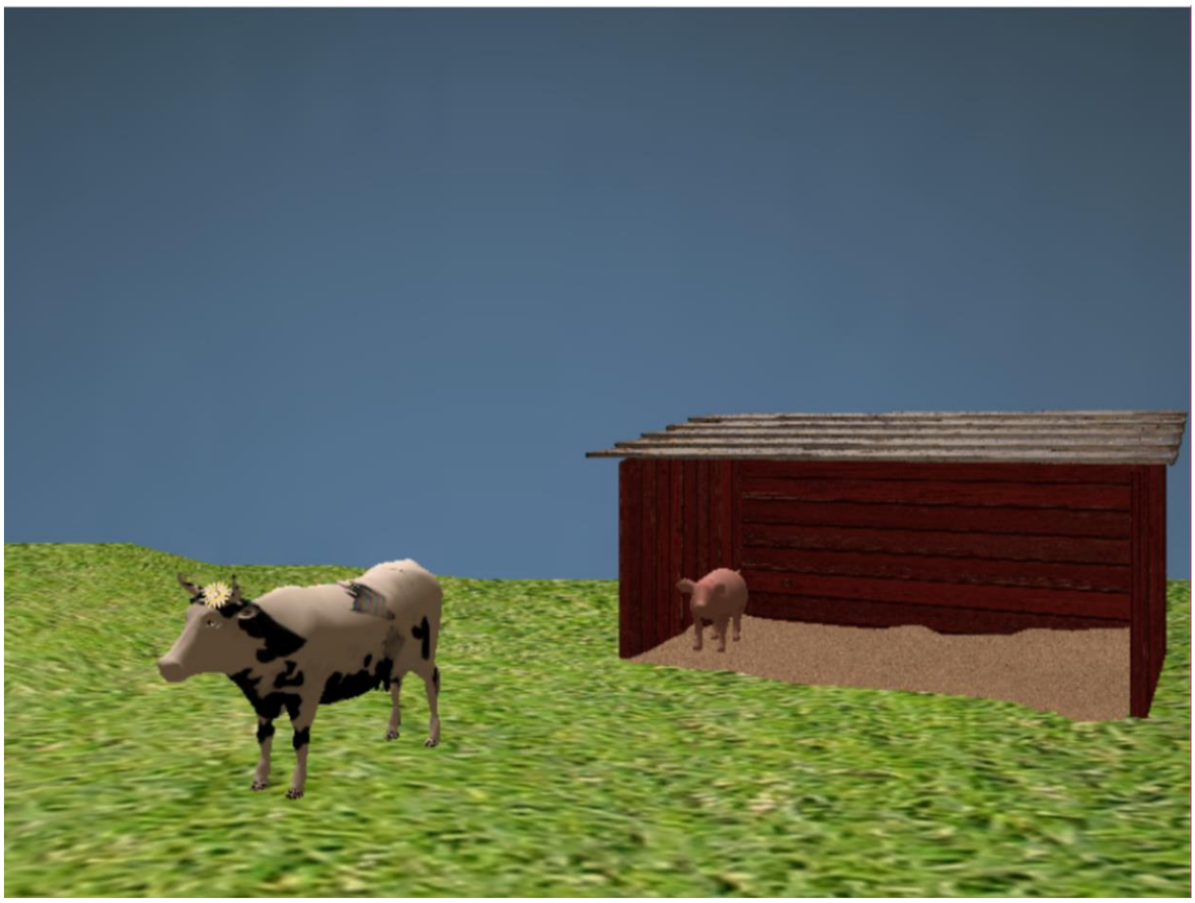
Final Lab

Eva Nolan

14335043

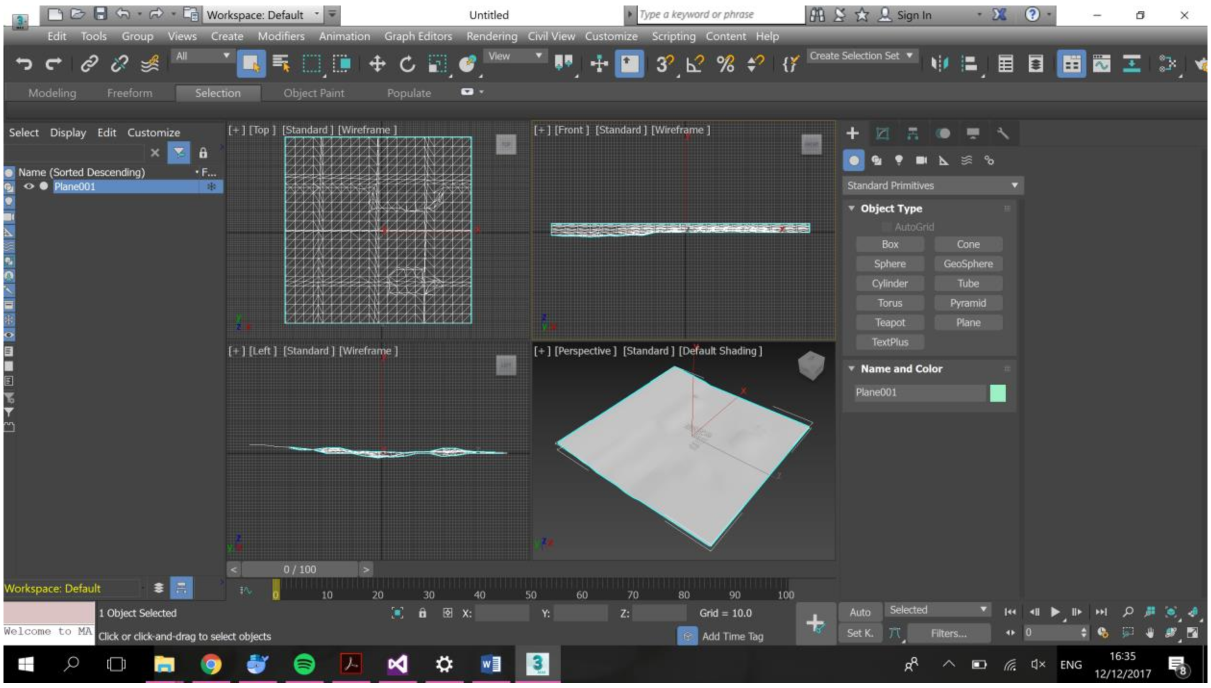


My final lab consisted of a farmyard scene. It consists of two models downloaded from the internet and four modles I made myself on 3DS Max. Camera control is done by keyboard and mouse and can scroll left, right, forwards and backwards as well as rotate about the yaxis of the scene. A video is included with this submission and is also available at https://youtu.be/_L5jDomGHgo showing displaying the functionality of the program but the low frame rate makes the movement look very jumpy.

Models

Ground

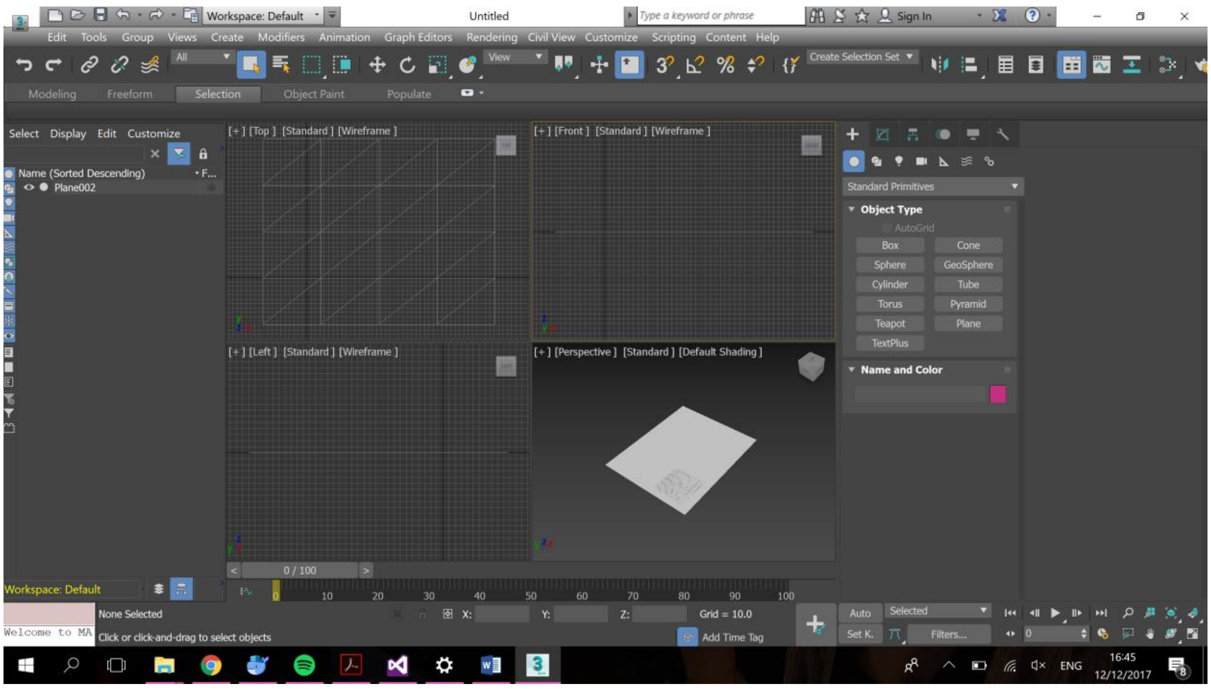
The ground model was made from a plan on 3DSMax. I found that a flat plane looked very unrealistic and so added raised and dipped areas to give the appearance of a hilly plain.


Workspace: Default
Untitled
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Civil View Customize Scripting Content Help
Modeling Freeform Selection Object Paint Populate
[+] [Top] [Standard] [Wireframe]
[+] [Front] [Standard] [Wireframe]
[+] [Left] [Standard] [Wireframe]
[+] [Perspective] [Standard] [Default Shading]
Standard Primitives
Object Type
Box Cone Sphere GeoSphere Cylinder Tube Torus Pyramid Teapot Plane TextPlus
Name and Color
Plane001
1 Object Selected
Click or click-and-drag to select objects
Grid = 10.0

```
//ground
mat4 model3 = identity_mat4();
mat4 Smat3 = { 10.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 10.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 10.0f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
model3 = model3*Smat3;
model3 = rotate_y_deg(model3, 90);
model3 = translate(model3, vec3(-1000.0, -400.0f, 1600.0f));

mat4 global3 = model3;
// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global3.m);
glBindTexture(GL_TEXTURE_2D, GROUND_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(GROUND_ID);
glDrawArrays(GL_TRIANGLES, 0, ground_count);
```

Sky

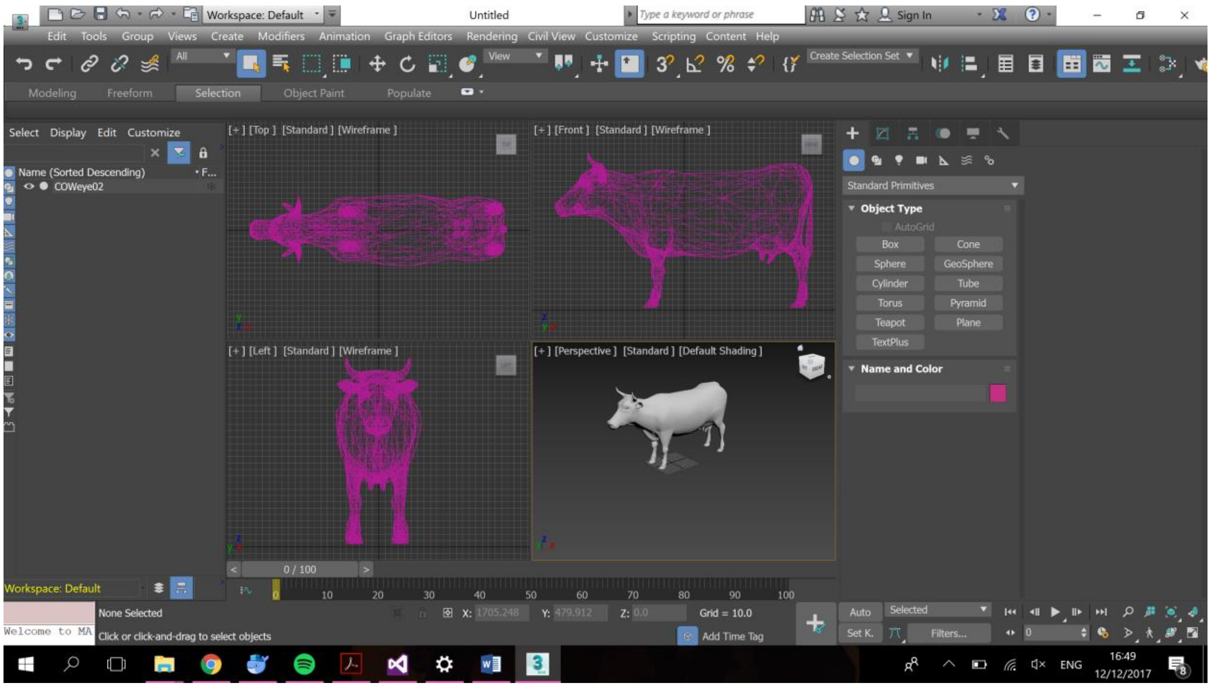
I made the sky as a flat plane in open gl. It was drawn at a right angle to the ground plane.


Workspace: Default
Untitled
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Civil View Customize Scripting Content Help
Modeling Freeform Selection Object Paint Populate
Select Display Edit Customize
Name (Sorted Descending)
Plane002
[+] [Top] [Standard] [Wireframe]
[+] [Front] [Standard] [Wireframe]
[+] [Left] [Standard] [Wireframe]
[+] [Perspective] [Standard] [Default Shading]
Standard Primitives
Object Type
Box
Cone
Sphere
GeoSphere
Cylinder
Tube
Torus
Pyramid
Teapot
Plane
TextPlus
Name and Color
0 / 100
None Selected
Click or click-and-drag to select objects
Grid = 10.0
Add Time Tag
16:45
12/12/2017

```
//sky
mat4 model7 = identity_mat4();
model7 = model7*Smat3*Smat3;
model7 = rotate_x_deg(model7, 90);
model7 = rotate_z_deg(model7, 90);

model7 = translate(model7, vec3(-1500.0, -200.0f, 1350.0f));

mat4 global7 = model7;

// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global7.m);
glBindTexture(GL_TEXTURE_2D, SKY_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(SKY_ID);
glDrawArrays(GL_TRIANGLES, 0, sky_count);
```

Cow

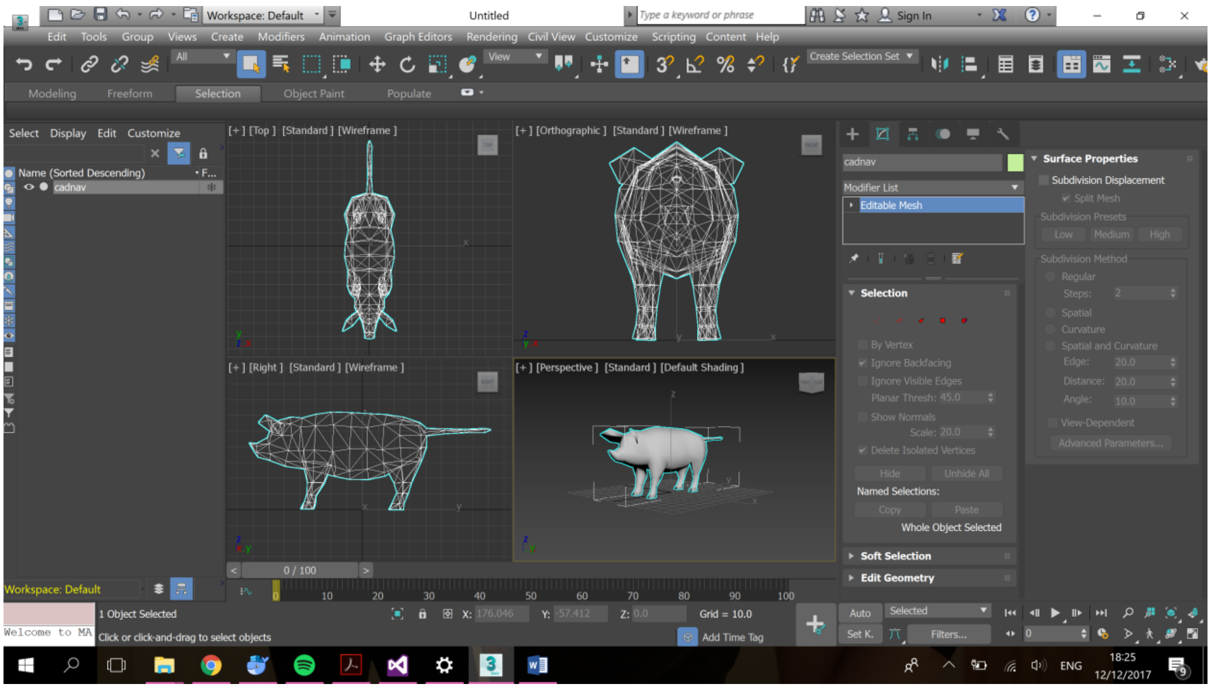
The cow model was downloaded from archive3d.net. The texture came with the model.


Workspace: Default
Untitled
Type a keyword or phrase
Sign In
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Civil View Customize Scripting Content Help
Create Selection Set
Modeling Freeform Selection Object Paint Populate
Select Display Edit Customize
Name (Sorted Descending)
COWeye02
[+] [Top] [Standard] [Wireframe]
[+] [Front] [Standard] [Wireframe]
[+] [Left] [Standard] [Wireframe]
[+] [Perspective] [Standard] [Default Shading]
Standard Primitives
Object Type
AutoGrid
Box Cone
Sphere GeoSphere
Cylinder Tube
Torus Pyramid
Teapot Plane
TextPlus
Name and Color
0 / 100
Workspace: Default
None Selected
Click or click-and-drag to select objects
Welcome to MA
Grid = 10.0
Add Time Tag
Auto Selected
Set K. Filters...
16:49
12/12/2017
ENG

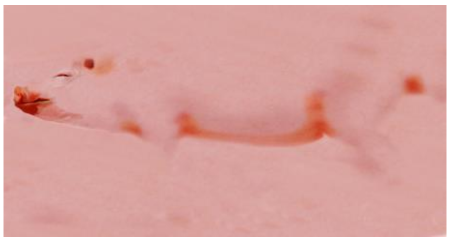
```
//cow
mat4 model2 = identity_mat4();
mat4 Smat6 = { 0.75f, 0.0f, 0.0f, 0.0f,
    0.0f, 0.75f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.75f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
model2 = model2*Smat6;
model2 = rotate_y_deg(model2, -120);
model2 = translate(model2, vec3(100.0, -340.0f, -700.0f));
mat4 global2 = model2;
// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global2.m);
glBindTexture(GL_TEXTURE_2D, COW_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(COW_ID);
glDrawArrays(GL_TRIANGLES, 0, cow_count);
```

Pig

The pig model was downloaded from cadnav.com. The texture came with another pig model downloaded from the same website as it fit quite well.

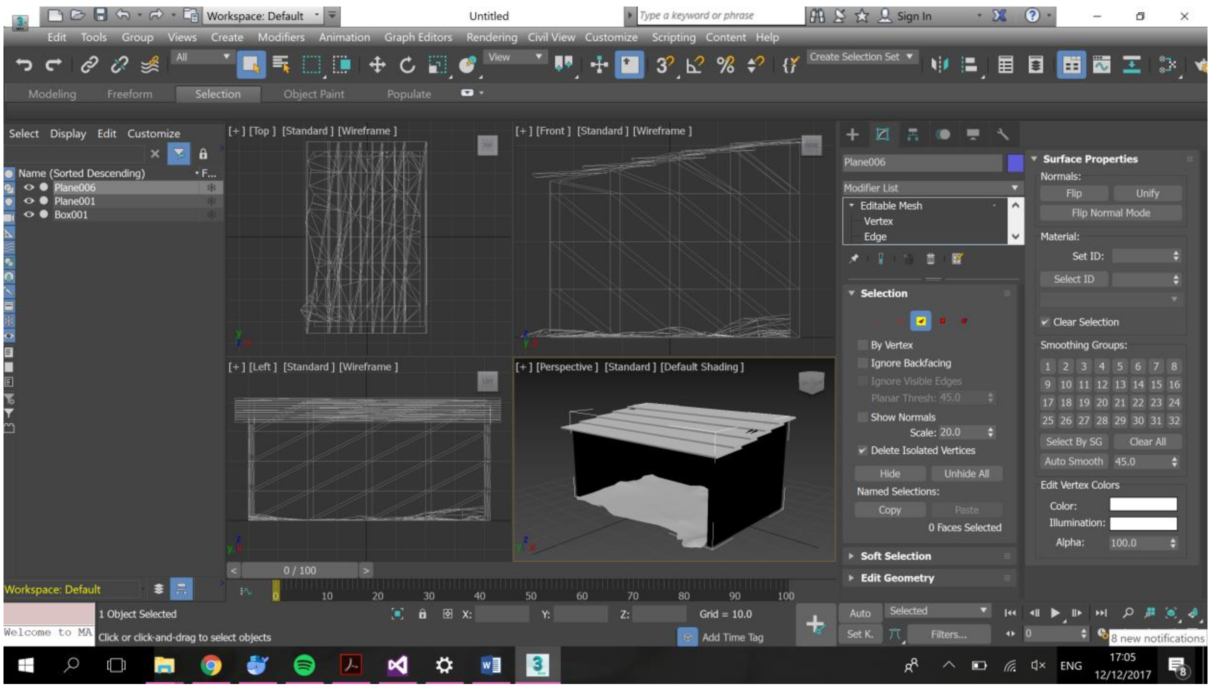

Untitled
Workspace: Default
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Civil View Customize Scripting Content Help
Modeling Freeform Selection Object Paint Populate
[+] [Top] [Standard] [Wireframe]
[+] [Orthographic] [Standard] [Wireframe]
[+] [Right] [Standard] [Wireframe]
[+] [Perspective] [Standard] [Default Shading]
cadnav
Modifier List
Editable Mesh
Selection
Surface Properties
1 Object Selected
Click or click-and-drag to select objects

```
//pig
mat4 model1 = identity_mat4();
model1= rotate_y_deg(model1,240);
mat4 Smat = { 02.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 02.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 02.0f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
model1 = model1*Smat;
model1 = rotate_y_deg(model1, -80);
model1 = translate(model1, vec3(-400.0, -340.0f, 50.0f));
mat4 global1 = model1;

// update uniforms & draw
glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj1.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view1.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global1.m);

glBindTexture(GL_TEXTURE_2D, PIG_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(PIG_ID);
glDrawArrays (GL_TRIANGLES, 0, pig_count);
```



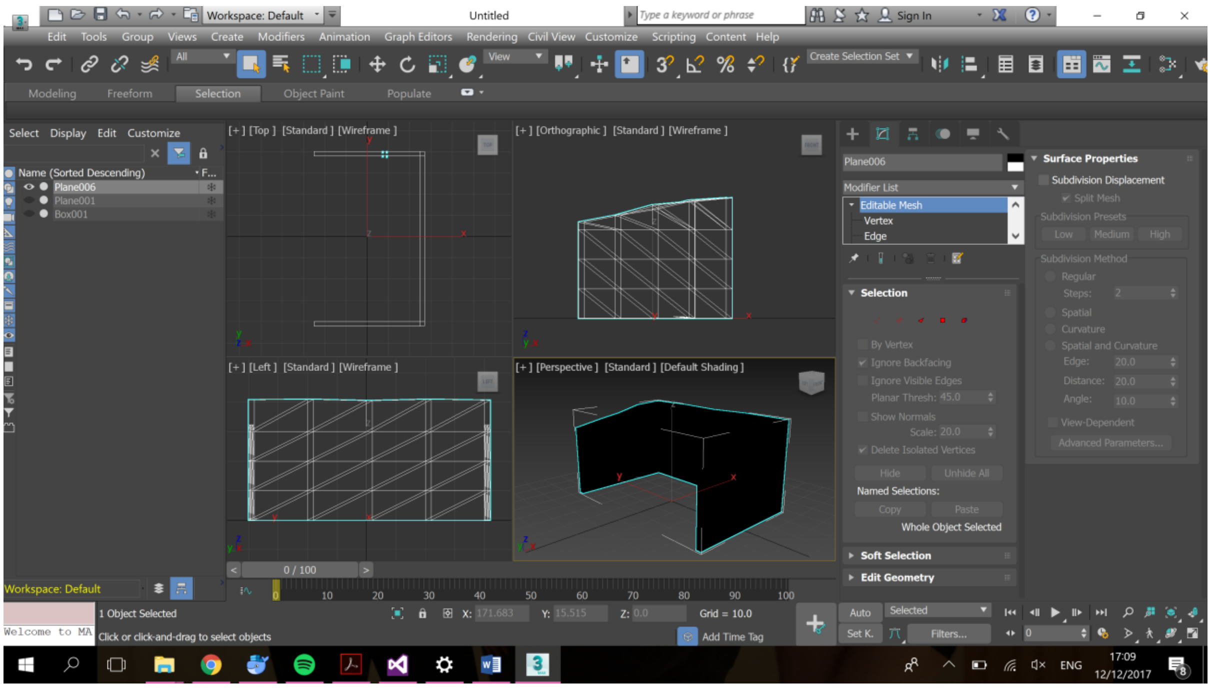
Shed

I built the shed model on 3DSMax. The shed hierarchy is made up of three separate meshes; the roof, the walls and the floor. This was done so that they could all be textured separately.



Shed Walls

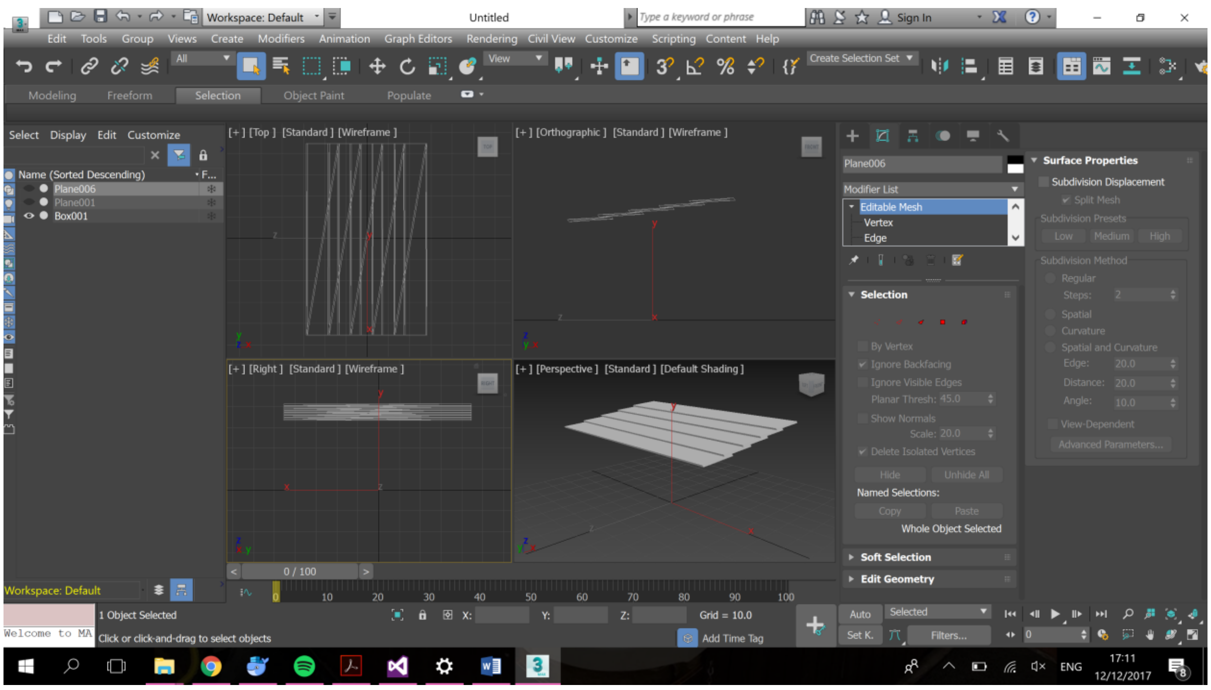
The walls acted as the root of the barn hierarchy.



```
//BARN
mat4 model4 = identity_mat4();
mat4 Scale_Barn = { 10.00f, 0.0f, 0.0f, 0.0f,
    0.0f, 10.00f, 0.0f, 0.0f,
    0.0f, 0.0f, 10.00f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
vec3 barnpos = { -700.0f, -370.0f, 100.0f };
model4 = model4*Scale_Barn;
model4 = rotate_y_deg(model4, 240);
model4 = translate(model4, barnpos);

mat4 global4 = model4;
// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global4.m);
glBindTexture(GL_TEXTURE_2D, BARN_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(BARN_ID);
glDrawArrays(GL_TRIANGLES, 0, barn_count);
```

Barn Roof

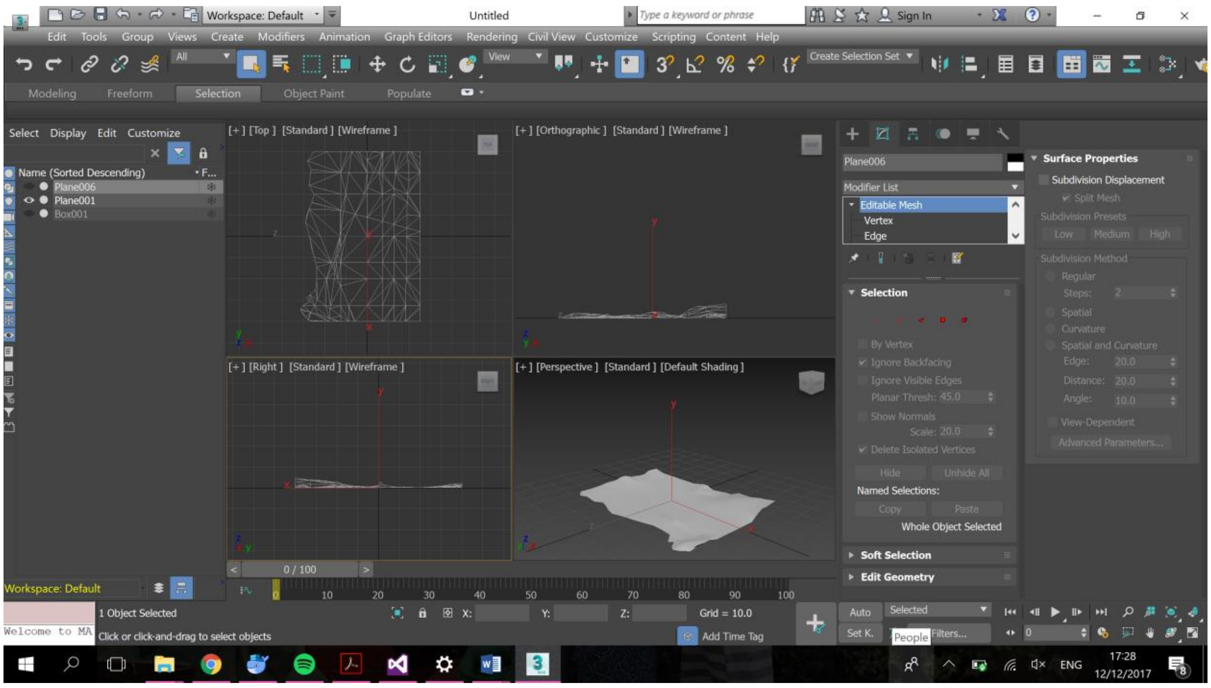
The roof was made up of four boxes layered the give the effect of a roof made of sheets of metal



```
//barn roof
mat4 model6 = identity_mat4();
mat4 global6 = global4* model6;

// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global6.m);
glBindTexture(GL_TEXTURE_2D, BARN_ROOF_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(BARN_ROOF_ID);
glDrawArrays(GL_TRIANGLES, 0, barn_roof_count);
```

Barn Floor

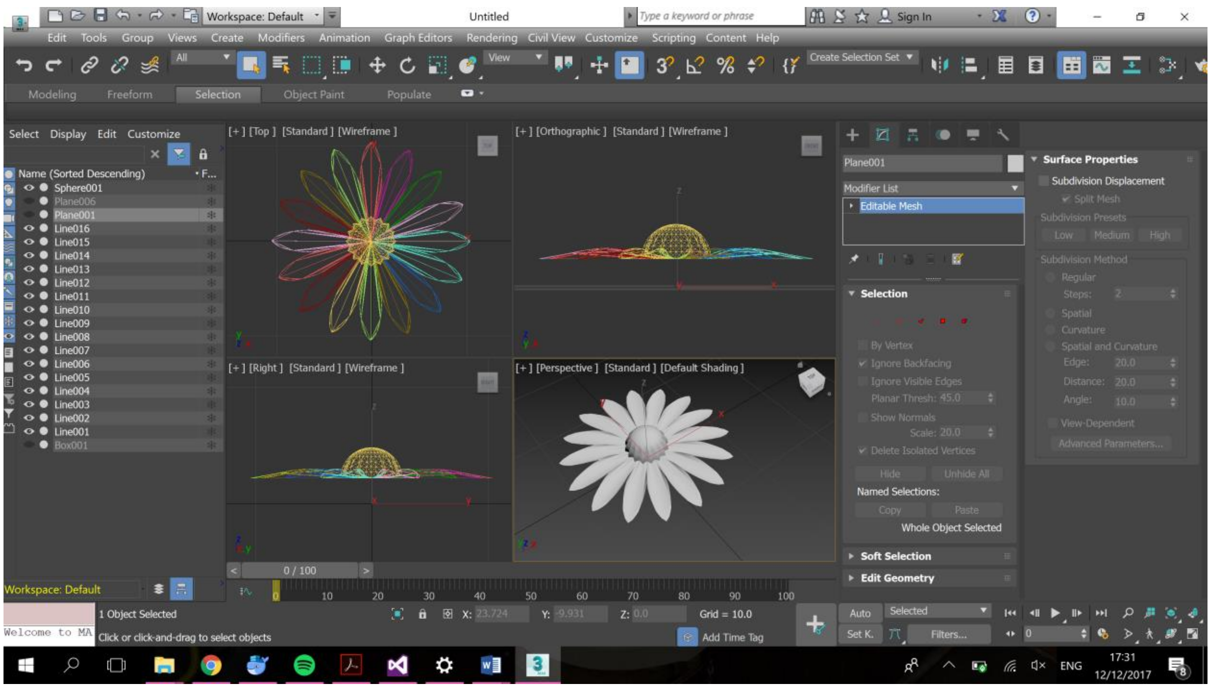
The bar floor was also a child of the barn walls.

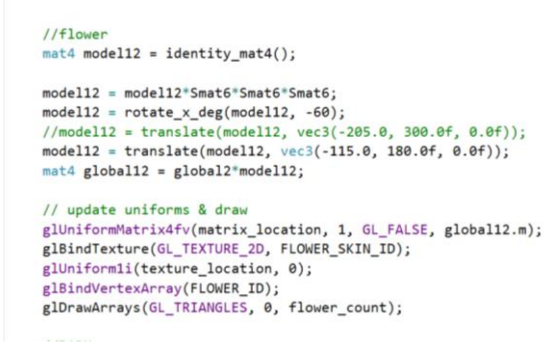

Workspace: Default
Untitled
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Civil View Customize Scripting Content Help
Modeling Freeform Selection Object Paint Populate
Select Display Edit Customize
Name (Sorted Descending)
Plane006
Plane001
Box001
[+] [Top] [Standard] [Wireframe]
[+] [Orthographic] [Standard] [Wireframe]
[+] [Right] [Standard] [Wireframe]
[+] [Perspective] [Standard] [Default Shading]
Plane006
Modifier List
Editable Mesh
Vertex
Edge
Selection
Soft Selection
Edit Geometry
Surface Properties
Whole Object Selected
1 Object Selected
Click or click-and-drag to select objects
Grid = 10.0
0 / 100

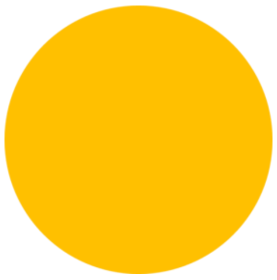
```
//barn floor
mat4 model5 = identity_mat4();
mat4 global5 =global4*model5;

// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global5.m);
glBindTexture(GL_TEXTURE_2D, BARN_GROUND_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(BARN_GROUND_ID);
glDrawArrays(GL_TRIANGLES, 0, barn_ground_count);
```

Flower

I made the daisy in 3DSMax. It is a child of the cow hierarchy as it sits behind the cows ear




```
//flower
mat4 model12 = identity_mat4();

model12 = model12*Smat6*Smat6*Smat6;
model12 = rotate_x_deg(model12, -60);
//model12 = translate(model12, vec3(-205.0, 300.0f, 0.0f));
model12 = translate(model12, vec3(-115.0, 180.0f, 0.0f));
mat4 global12 = global2*model12;

// update uniforms & draw
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, global12.m);
glBindTexture(GL_TEXTURE_2D, FLOWER_SKIN_ID);
glUniform1i(texture_location, 0);
glBindVertexArray(FLOWER_ID);
glDrawArrays(GL_TRIANGLES, 0, flower_count);
```

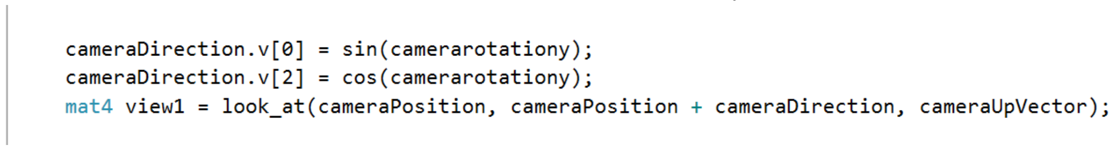


Camera Control

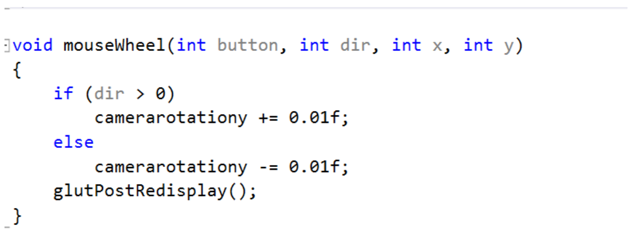
Camera control is achieved through the use of two functions, one for the mouse wheel input and one for special key input.

<u>Rotation</u>

The camera is rotated clockwise and anticlockwise about the y axis when the mouse wheel is rolled.



```
cameraDirection.v[0] = sin(camerarotationy);
cameraDirection.v[2] = cos(camerarotationy);
mat4 view1 = look_at(cameraPosition, cameraPosition + cameraDirection, cameraUpVector);
```



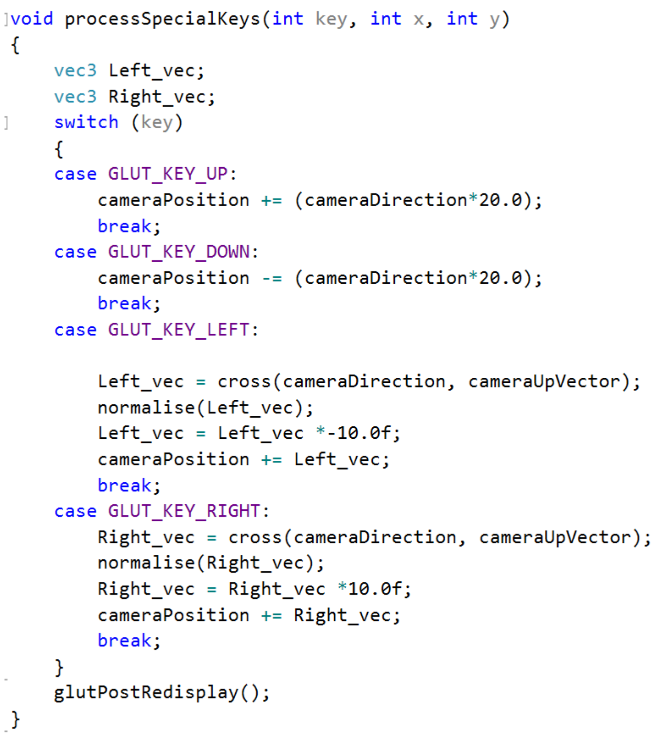
```
void mouseWheel(int button, int dir, int x, int y)
{
    if (dir > 0)
        camerarotationy += 0.01f;
    else
        camerarotationy -= 0.01f;
    glutPostRedisplay();
}
```

<u>Special Keys</u>

On a special key press the camera moves forward, backward, left and right.

Forward and backward were simple to implement, simply increasing the camera position by the camera direction times a small increment.

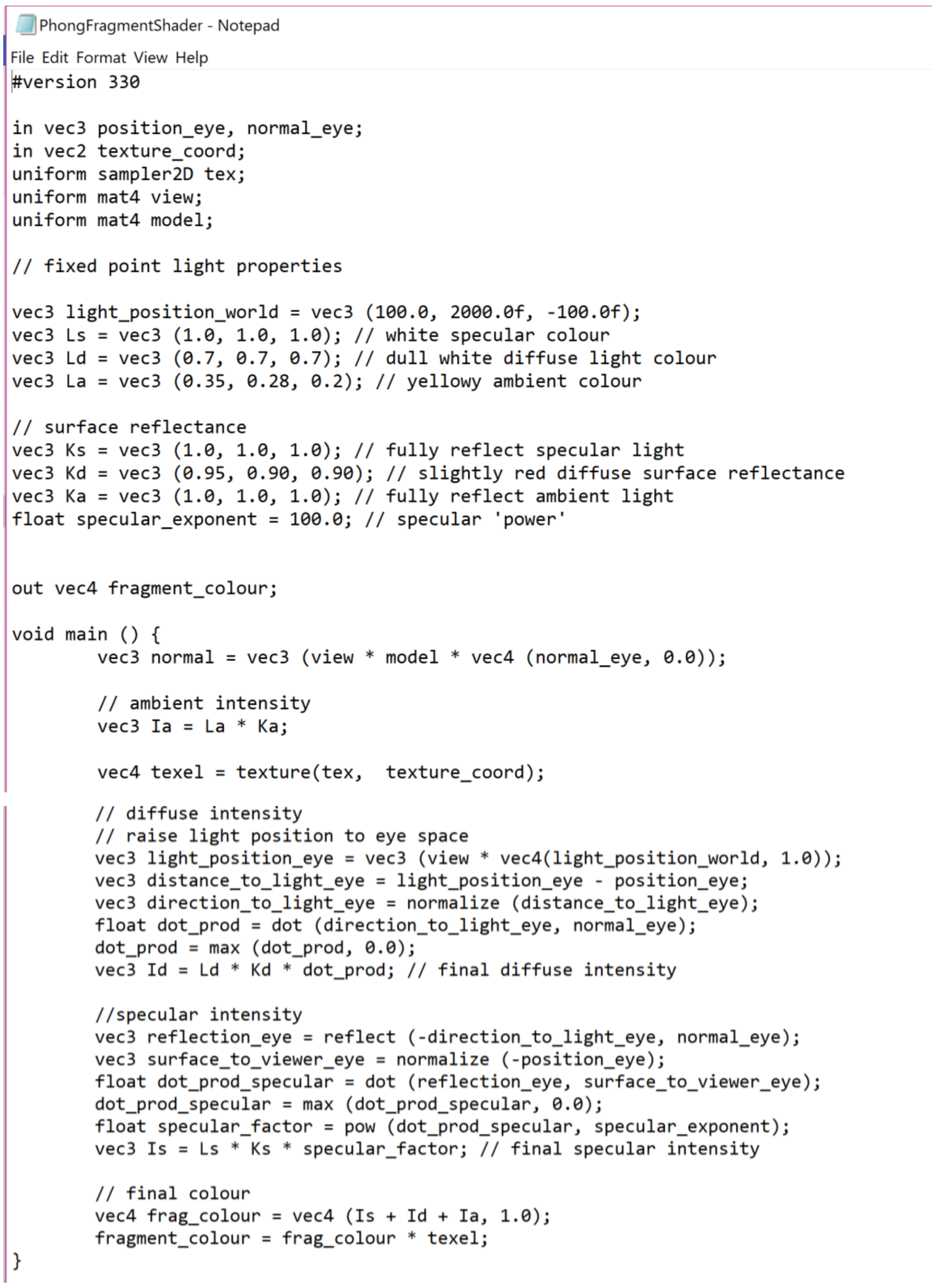
The left and right directions were more complex. It was necessary to get the cross product of the camera direction and the up vector and then normalising the resultant vector this gives the direction that is perpendicular to the direction in which the camera is currently pointing. Then in the same way as in the forward and backward calculations this vector is multiplied by the increment we wish to move by and added or subtracted from the camera position.



```
void processSpecialKeys(int key, int x, int y)
{
    vec3 Left_vec;
    vec3 Right_vec;
    switch (key)
    {
    case GLUT_KEY_UP:
        cameraPosition += (cameraDirection*20.0);
        break;
    case GLUT_KEY_DOWN:
        cameraPosition -= (cameraDirection*20.0);
        break;
    case GLUT_KEY_LEFT:

        Left_vec = cross(cameraDirection, cameraUpVector);
        normalise(Left_vec);
        Left_vec = Left_vec *-10.0f;
        cameraPosition += Left_vec;
        break;
    case GLUT_KEY_RIGHT:
        Right_vec = cross(cameraDirection, cameraUpVector);
        normalise(Right_vec);
        Right_vec = Right_vec *10.0f;
        cameraPosition += Right_vec;
        break;
    }
    glutPostRedisplay();
}
```

Lighting

The lighting implemented is the phong lighting model. The light source position is above the scene so it acts as the sun in the scene



```
#version 330

in vec3 position_eye, normal_eye;
in vec2 texture_coord;
uniform sampler2D tex;
uniform mat4 view;
uniform mat4 model;

// fixed point light properties

vec3 light_position_world = vec3 (100.0, 2000.0f, -100.0f);
vec3 Ls = vec3 (1.0, 1.0, 1.0); // white specular colour
vec3 Ld = vec3 (0.7, 0.7, 0.7); // dull white diffuse light colour
vec3 La = vec3 (0.35, 0.28, 0.2); // yellowy ambient colour

// surface reflectance
vec3 Ks = vec3 (1.0, 1.0, 1.0); // fully reflect specular light
vec3 Kd = vec3 (0.95, 0.90, 0.90); // slightly red diffuse surface reflectance
vec3 Ka = vec3 (1.0, 1.0, 1.0); // fully reflect ambient light
float specular_exponent = 100.0; // specular 'power'


out vec4 fragment_colour;

void main () {
        vec3 normal = vec3 (view * model * vec4 (normal_eye, 0.0));

        // ambient intensity
        vec3 Ia = La * Ka;

        vec4 texel = texture(tex,  texture_coord);

        // diffuse intensity
        // raise light position to eye space
        vec3 light_position_eye = vec3 (view * vec4(light_position_world, 1.0));
        vec3 distance_to_light_eye = light_position_eye - position_eye;
        vec3 direction_to_light_eye = normalize (distance_to_light_eye);
        float dot_prod = dot (direction_to_light_eye, normal_eye);
        dot_prod = max (dot_prod, 0.0);
        vec3 Id = Ld * Kd * dot_prod; // final diffuse intensity

        //specular intensity
        vec3 reflection_eye = reflect (-direction_to_light_eye, normal_eye);
        vec3 surface_to_viewer_eye = normalize (-position_eye);
        float dot_prod_specular = dot (reflection_eye, surface_to_viewer_eye);
        dot_prod_specular = max (dot_prod_specular, 0.0);
        float specular_factor = pow (dot_prod_specular, specular_exponent);
        vec3 Is = Ls * Ks * specular_factor; // final specular intensity

        // final colour
        vec4 frag_colour = vec4 (Is + Id + Ia, 1.0);
        fragment_colour = frag_colour * texel;
}
```