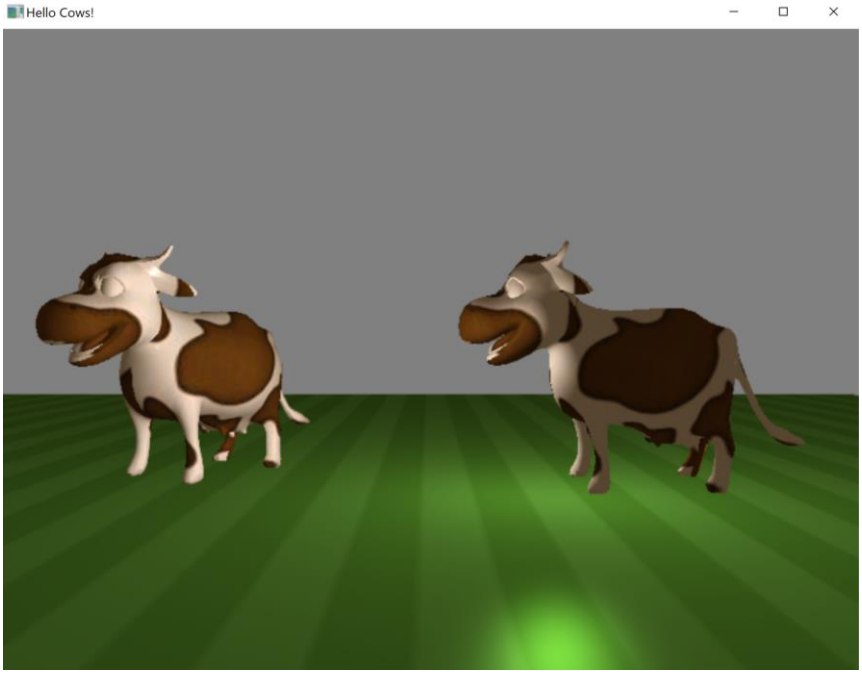
# Interactive Scene Setup for Final Project

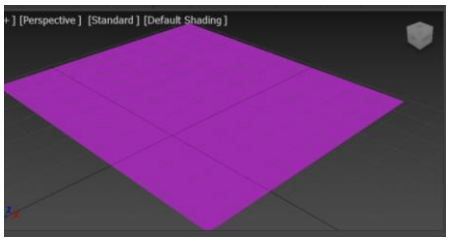
**Eva Nolan**

**14335443**

## More than one mesh is to be loaded from a file and displayed, and a consistent scene should be created.




I created a simple plane model on 3DS Max. The ground model had to be significantly scaled up to match the large cow models

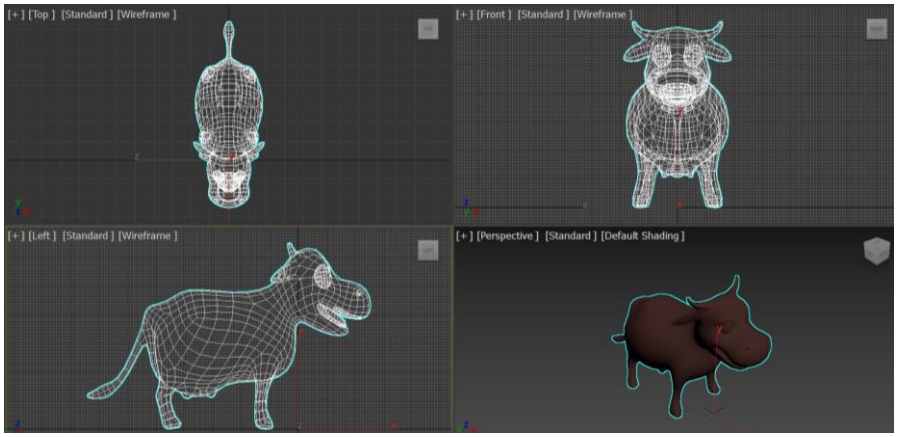



```
//ground
mat4 model3 = identity_mat4();
model3 = translate(model3, vec3(0.0, -200.0f, 1600.0f));
mat4 Smat3 = { 50.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 50.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 50.0f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
model3 = model3*Smat3;
mat4 global3 = model3;

// 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);
```

The cow model was downloaded from TurboSquid.com.

The cow model was instantiated twice

Both models had to be exported with the faces set to traingles so they would appear smooth in the scene




```
mat4 model1 = identity_mat4();
model1= rotate_y_deg(model1,90);
model1=translate(model1, vec3(0.0, -200.0f, 100.0f));
mat4 Smat = { 0.50f, 0.0f, 0.0f, 0.0f,
    0.0f, 0.50f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.50f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
};
model1 = model1*Smat;
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, COW_SKIN_ID);
glUniform1i(texture_location, 0);

glBindVertexArray(COW_ID);
glDrawArrays (GL_TRIANGLES, 0, cow_count);
```

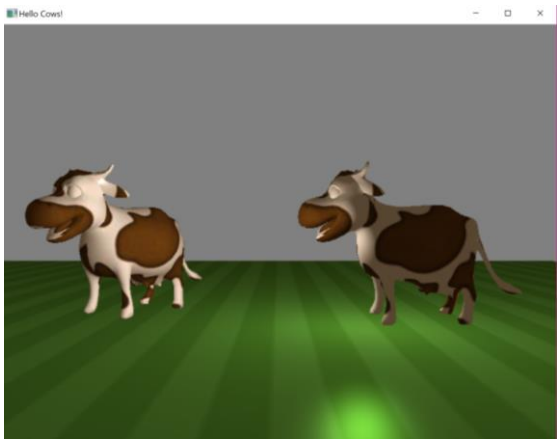
```
//cow 2
mat4 model2 = identity_mat4();

model2 = rotate_y_deg(model2, 145);
model2 = translate(model2, vec3(600.0, -200.0f, 100.0f));
model2 = model2*Smat;
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);
```

## Create an interactive walkthrough that allows the user to move around the scene using the keyboard and/or the mouse.

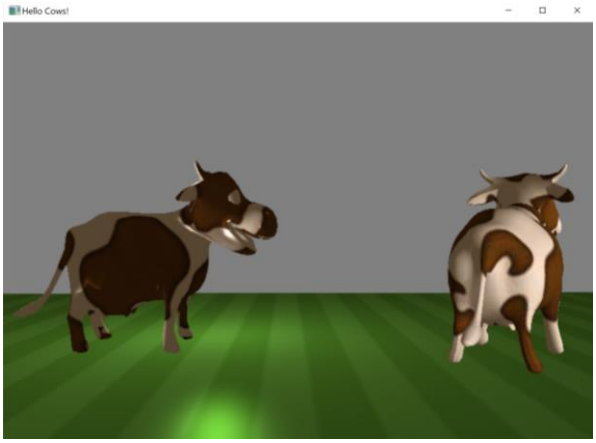
```
// Placeholder code for the keypress
void keypress(unsigned char key, int x, int y) {

    if (key == 'l') {
        camerarotationy += 0.01f;
    }
    if (key == 'r') {
        camerarotationy -= 0.01f;

    }
    glutPostRedisplay();
```

```
void processSpecialKeys(int key, int x, int y)
{
    switch (key)
    {
    case GLUT_KEY_UP:
        cameraPosition = cameraPosition + vec3(0.0f, 0.0f, 20.0f);
        break;
    case GLUT_KEY_DOWN:
        cameraPosition = cameraPosition - vec3(0.0f, 0.0f, 20.0f);
        break;
    case GLUT_KEY_LEFT:
        cameraPosition = cameraPosition + vec3(5.0f, 0.0f, -2.0f);
        break;
    case GLUT_KEY_RIGHT:
        cameraPosition = cameraPosition - vec3(5.0f, 0.0f, -2.0f);
    }
    glutPostRedisplay();
}
```

Special keys up, down, left and right were used to navigate forwards, backwards, left and right through the scene the “r” and “l” keys rotate the camera clockwise and anticlockwise about the y axis of the scene




*Cow scene from front*




*Cow scene from behind*

## Implementation of the Phong Illumination model

PhongFragmentShader - Notepad

File Edit Format View Help

```
#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 (0.0, 0.0, 2.0);
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;
```

The colours of light were defined in the Fragment shader outside of the main. They were chosen to give the scene as soft yellow light.

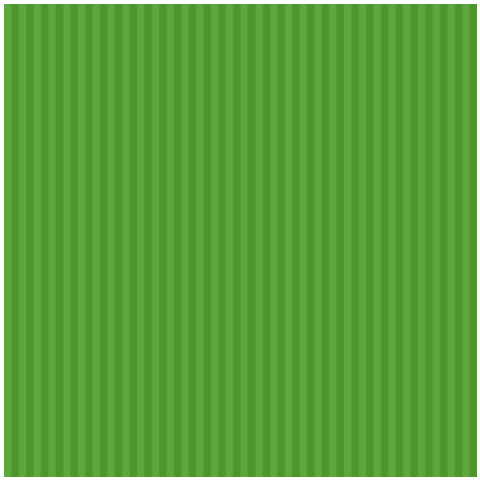
The specular power was set to 100 as this produced a nice specular reflection but didn't make the cows appear too shiny.

```
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;
}
```

The light position was raised to eye space and the normals were calculated in the main function of the fragment shader.

The final diffuse and final specular intensity were calculated

## Texturing your models using an image file



The files used to texture the image were.png files found online.

The striped grass function was chosen as the perspective projection used makes the stripes appear to converge in the distance.

The cow texture chosen is animated which suits the rounded animated shapes of the cow model.

The textures are mapped to the models in the fragment shader and in the Generatetexture function. In this function, the way in which the textures wrap around the objects is mapped.

```
void generateTexture(GLuint& texture_id, char* file_name)
{
    //printf("got here");
    int width, height, n;
    unsigned char* image = stbi_load(file_name, &width, &height, &n, STBI_rgb);

    glGenTextures(1, &texture_id);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, texture_id);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glBindTexture(GL_TEXTURE_2D, 0);
}
```