COMP 3069 Computer Graphics

Tianqi Xia 20123768

How to run the program:

1. Use Xcode. The program is developed by Xcode on Mac. Download the whole project file and click the 'Coursework2.xcodeproj' file. Then you will go into the project file directly.

2. Change the absolute path for each texture image and object according to the specific path on your computer.

```
fanLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/fan/fan.obj");
couchLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/couch/Couch.obj");
tvLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/tv/tv.obj");
curtainLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/curtain/curtain.obj");
plantLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/plant/plant.obj");
telescopeLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/telescope/telescope.obj");
paperLoader.LoadFile("/Users/hailey/Desktop/Coursework2/Coursework2/data/paper/paper.obj");
```

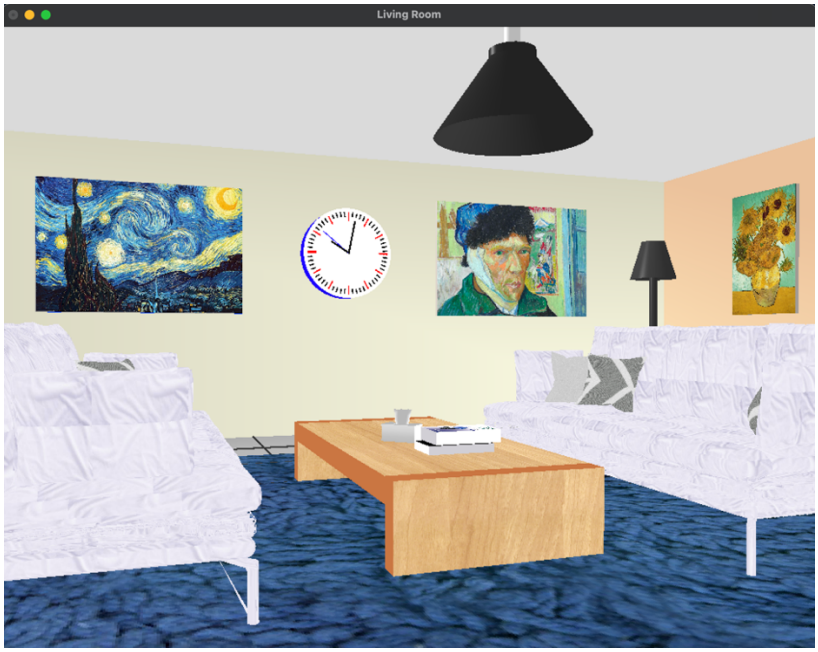
```
// Loading Textures
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/gogh.png", &vanGoghTexture);
setupTexture(&vanGoghTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/starry-night.png", &starTexture);
setupTexture(&starTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/vangogh.png", &GoghTexture);
setupTexture(&GoghTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/sunflower.png", &sunflowerTexture);
setupTexture(&sunflowerTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/flower.png", &flowerTexture);
setupTexture(&flowerTexture);

loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/sheet.png", &sheetTexture);
setupTexture(&sheetTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/wood.png", &woodTexture);
setupTexture(&woodTexture);
loadTexture("/Users/hailey/Desktop/Coursework2/Coursework2/texture/wood2.png", &wood1Texture);
setupTexture(&wood1Texture);
```

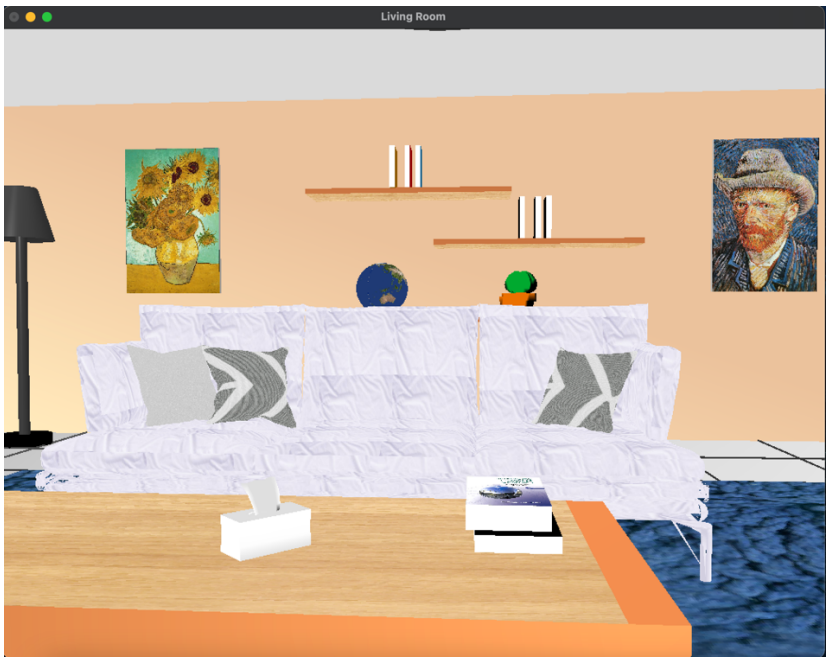
3. Run the program directly. Several keyboard commands for the program are shown below.
   a) Press O open/close the door.
   b) Press Y to change the environment between the city and rural area.
   c) Press P to open/close the window.
   d) Press V to turn on/off the fan. There are 3 speeds of the fan.
   e) Press T to turn on/off the tellurion. There are 3 speeds of the tellurion.
   f) Press B to bounce the basketball. Bouncing is based on the gravity and friction.
   g) Press W/A/S/D and use mouse to change the viewpoints.
   h) Press R to change to the telescope viewpoint.
   i) Press E to change to the couch viewpoint.
   j) Press L to turn on/off the lamp.
   k) Press K to turn on/off the celling lamp.
   l) Press J to turn on/off the sunlight.
   m) Press X/Z to tilt scene.
   n) Clock can show the local time.

Scene screenshots:

1. Inside view (from front, back, right, left)


Living Room
F·R·I·E·N·D·S
THE REUNION


Living Room


Living Room

2. Outside view

Project Requirements:

1. 3D Models (Blender & OpenGL) & Transformation

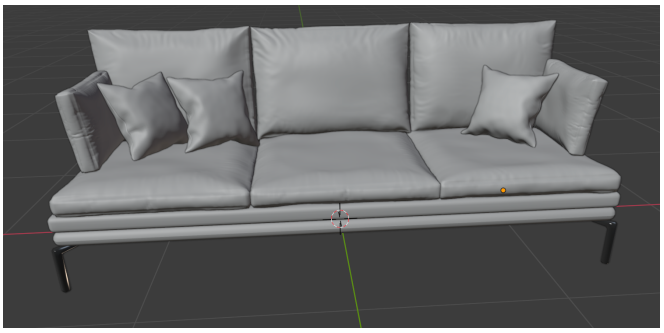
a) Blender: all the objects are created in Blender and loaded through 'LoadFile' Function from pmba[1].

```
bool LoadFile(std::string Path)
{
    // If the file is not an .obj file return false
    if (Path.substr(Path.size() - 4, 4) != ".obj")
        return false;


    std::ifstream file(Path);

    if (!file.is_open())
        return false;

    LoadedMeshes.clear();
    LoadedVertices.clear();
    LoadedIndices.clear();

    std::vector<Vector3> Positions;
    std::vector<Vector2> TCoords;
    std::vector<Vector3> Normals;

    std::vector<Vertex> Vertices;
    std::vector<unsigned int> Indices;
```

1. **Couch & Pillow**: Couch & pillows are built in blender but textured in OpenGL. Different pillows and couch

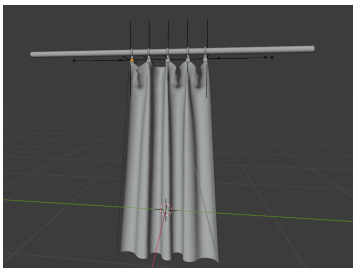
[1] https://github.com/pmba/opengl-room

have different textures. The couch and pillows consist of 14 items.

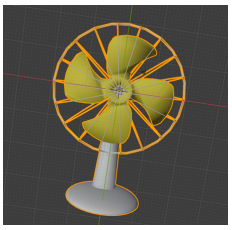


```
// left 2 pillow
glEnable(GL_TEXTURE_2D);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glBindTexture(GL_TEXTURE_2D, couchTexture1.id);
couchMesh0.material.active();
glVertexPointer(3, GL_FLOAT, 0, &couchMesh0.vertices_pointers[0]);
glTexCoordPointer(2, GL_FLOAT, 0, &couchMesh0.vertices_tex_coords[0]);
glNormalPointer(GL_FLOAT, 0, &couchMesh0.vertices_normals[0]);
glDrawElements(GL_TRIANGLES, couchMesh0.indices_pointers.size(), GL_UNSIGNED_INT, &couchMesh0.indices_pointers[0]);
glDisable(GL_TEXTURE_2D);
// glFlush();
```

2. **Curtain**: Curtains are built in blender but colored in OpenGL.

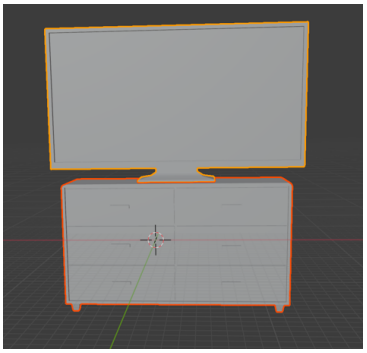


3. **Fan**: Fan is built in blender but colored in OpenGL.



```
// Fan
glPushMatrix();
    GLfloat fanTranslate[3] = {-8, 2, -7};
    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_NORMAL_ARRAY);

    glPushMatrix();
        glTranslatef(fanTranslate[0], fanTranslate[1], fanTranslate[2]);
        glRotatef(30, 0, 1, 0);
        glRotatef(-90, 1, 0, 0);
        glScalef(0.05, 0.05, 0.05);

        fanMesh.material.active();
        fanMesh.material.dye();

        glVertexPointer(3, GL_FLOAT, 0, &fanMesh.vertices_pointers[0]);
        glNormalPointer(GL_FLOAT, 0, &fanMesh.vertices_normals[0]);
        glDrawElements(GL_TRIANGLES, fanMesh.indices_pointers.size(), GL_UNSIGNED_INT,
            &fanMesh.indices_pointers[0]);
    glPopMatrix();
```

4. **TV & TV Dresser**: TV & TV Dresser are built in blender but textured in OpenGL. TV has a screen texture. TV Dresser has a wood texture.

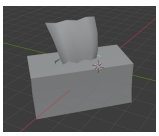
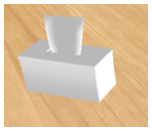



```
glPushMatrix();
glTranslatef(4, 2.7, 9.99);
glScalef(0.12, 0.12, 0.12);

glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);

glEnable(GL_TEXTURE_2D);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glBindTexture(GL_TEXTURE_2D, wood1Texture.id);
tvMesh0.material.active();
  tvMesh0.material.dye();
glVertexPointer(3, GL_FLOAT, 0, &tvMesh0.vertices_pointers[0]);
glTexCoordPointer(2, GL_FLOAT, 0, &tvMesh0.vertices_tex_coords[0]);
glNormalPointer(GL_FLOAT, 0, &tvMesh0.vertices_normals[0]);
glDrawElements(GL_TRIANGLES, tvMesh0.indices_pointers.size(), GL_UNSIGNED_INT, &tvMesh0.indices_pointers[0]);
glDisable(GL_TEXTURE_2D);
```

5. **Cactus**: Cactus is built in blender but colored in OpenGL.

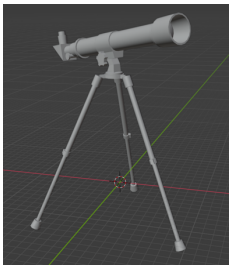
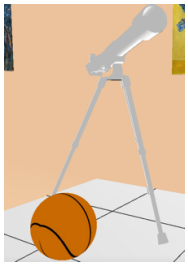

```
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
plantMesh0.material.active();
plantMesh0.material.dye();
glVertexPointer(3, GL_FLOAT, 0, &plantMesh0.vertices_pointers[0]);
glNormalPointer(GL_FLOAT, 0, &plantMesh0.vertices_normals[0]);
glDrawElements(GL_TRIANGLES, plantMesh0.indices_pointers.size(), GL_UNSIGNED_INT, &plantMesh0.indices_pointers[0]);
glPushMatrix();
plantMesh1.material.active();
plantMesh1.material.dye();
glVertexPointer(3, GL_FLOAT, 0, &plantMesh1.vertices_pointers[0]);
glNormalPointer(GL_FLOAT, 0, &plantMesh1.vertices_normals[0]);
glDrawElements(GL_TRIANGLES, plantMesh1.indices_pointers.size(), GL_UNSIGNED_INT, &plantMesh1.indices_pointers[0]);
```

6. **Tissue & Tissue Box**:

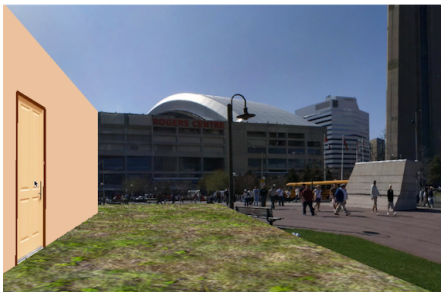
```
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
paperMesh.material.active();
paperMesh.material.dye();
glVertexPointer(3, GL_FLOAT, 0, &paperMesh.vertices_pointers[0]);
glNormalPointer(GL_FLOAT, 0, &paperMesh.vertices_normals[0]);
glDrawElements(GL_TRIANGLES, paperMesh.indices_pointers.size(), GL_UNSIGNED_INT, &paperMesh.indices_pointers[0]);
```

7. **Telescope**:

b) OpenGL

1. **Ground & Sky**: Sky is implemented by drawing six quadrangles and combine different texture images to each other. Ground is implemented by drawing a quadrangle and combine the texture to it.



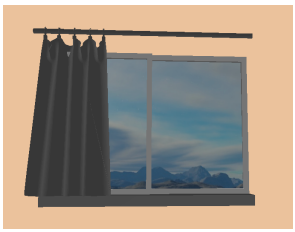

```
glTexCoord2f(0.0f, 1.0f);
glVertex3f(50.0f, 50.0f, 50.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(50.0f, -50.0f, 50.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-50.0f, -50.0f, 50.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(-50.0f, 50.0f, 50.0f);
```

2. **Door & Walls**: Use solid cube with texture and translation to represent the doors. Use quadrangle with colors to represent the walls and ceiling.

```
glColor3f(0.9294f, 0.9216f, 0.8353f);
glBegin(GL_QUADS);
glVertex3f(-10.0f, 0.0f, -10.0f);
glVertex3f(-10.0f, 7.0f, -10.0f);
glVertex3f(10.0f, 7.0f, -10.0f);
glVertex3f(10.0f, 0.0f, -10.0f);
glEnd();
```

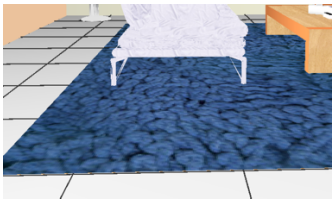
3. **Windows & Window Boards**: Window frames and window boards are built using solid cubes with scaling and translation. Window glasses are built using quadrangle and colored them with half transparency, so

that we can see the outside view through glasses.



```
glBlendFunc(GL_ZERO, GL_SRC_COLOR);
glEnable( GL_BLEND );
glPushMatrix();
    glColor3f(0.8, 0.8, 0.8);
    glTranslatef(0, 0, 0.1);
    glBegin(GL_QUADS);
        glVertex3f(0, 0, 0);
        glVertex3f(0.075, 0, innerWindowDimentions.x - 0.1);
        glVertex3f(0.075, innerWindowDimentions.y, innerWindowDimentions.x - 0.1);
        glVertex3f(0.075, innerWindowDimentions.y, 0);
    glEnd();
glPopMatrix();
glDisable( GL_BLEND );
```

4. **Floor, Floor Pattern, Carpet**: Build floor and floor pattern with quadrangle and line. Carpet is built with cube and texture.



```
glColor3f(0.149f, 0.149f, 0.149f);
glLineWidth(3.0f);
for(int i = 0; i < 20; i += 2)
{
    glBegin(GL_LINES);
    glVertex3f(-10.0f + i, 0.001f, -10.01f);
    glVertex3f(-10.0f + i, 0.001f, 10.01f);
    glEnd();
}
```

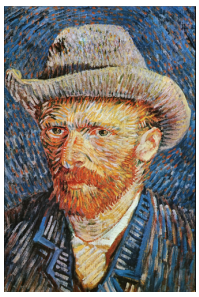
5. **Tea table & Books**: Use cube with texture, scaling, rotation to represent books and tea tables.

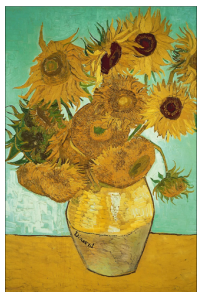
```
glPushMatrix();
    glTranslatef(-2, 0, -4.3);
    buildBoard(&woodTexture, rgb(212,134,84), {2.5, 1, 0.3});
glPopMatrix();
```

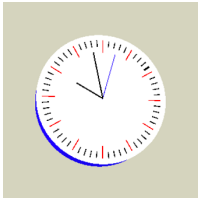
6. **Celling Lamp & Lamp**: Lamps were drawn using three cylinders with color, rotation and scaling.

```
glColor3f(0, 0, 0);
glTranslatef(0, 0.2f, 0);
glRotatef(90, 1, 0, 0);
gluCylinder(objCylinder, 0.5, 0.5, 0.2, 100, 100);
glPopMatrix();
```

7. **Murals & Clock**: There are five murals and a clock in the living room. Use cube and texture to represent the murals. Use cylinder, circle and line to draw the clock.



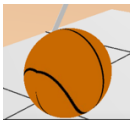




```
glBegin(GL_LINES);  'glBegin' is deprecated: first deprec
for(i=0; i<60; i++)
{
    if(i%5)
    {
        // normal minute
        if(i%5 == 1)
            //glColor3f(1.0f, 1.0f, 1.0f);
            glColor3f(0.0f, 0.0f, 0.0f);  'glCol
        newLine(minStart, minEnd, i*angle1min);
    }
    else
    {
        glColor3f(1.0f, 0.0f, 0.0f);  'glColor3f'
        newLine(stepStart, stepEnd, i*angle1min);
    }
}
```

8. **Table & Tellurion & Bookshelf with books**: Use sphere with texture to represent Tellurion. Use cubes with texture to represent the table. Use cubes with color and texture to represent bookshelf and books.

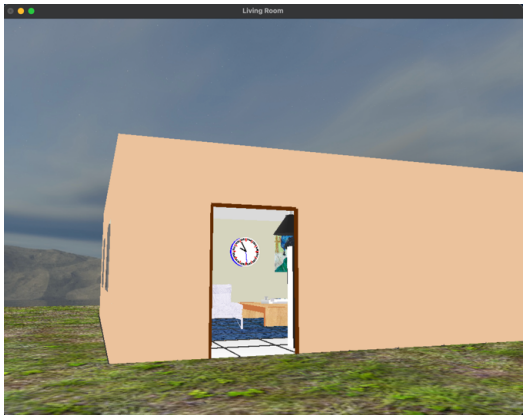
9. **Basketball**: Use sphere with texture to represent basketball.



10. **Loudspeaker Box**: Use solid cube with texture to represent the loudspeaker box.

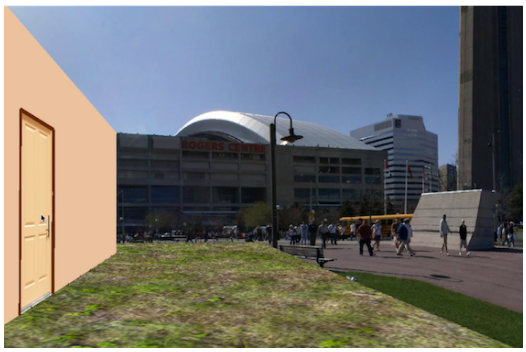
2. Animations

a) **Door**: Door can rotate. Press O open/close the door.

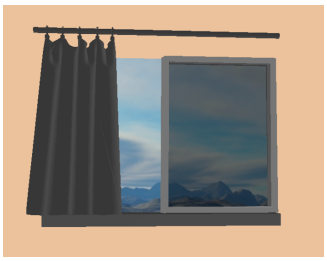



```
void checkDoorAngle()
{
    if (doorAnimation == FORWARDS || doorAnimation == BACKWARDS)
    {
        doorAngle += doorAnimation == FORWARDS ? 1 : -1;
        if (doorAngle == 90.0f || doorAngle == 0.0f) doorAnimation = IDLE;
    }
}
```

b) **Sky**: Press Y to change the environment between the city and rural area.



c) **Window**: Window can translate. Press P to open/close the window.



```
void checkWindowTranslation()
{
    if (windowAnimation == FORWARDS || windowAnimation == BACKWARDS)
    {
        windowTranslation += windowAnimation == FORWARDS ? 0.02 : -0.02;
        if (windowTranslation >= 1.0f || windowTranslation <= 0.0f) windowAnimation = IDLE;
    }
}
```

d) **Fan**: Fan can rotate. Press V to turn on/off the fan. There are 3 levels of speed of the fan.

```
void checkFanRotation()
{
    if (fanAnimation == FORWARDS || fanAnimation ==  BACKWARDS)
    {
        fanRotation += fanAnimation == FORWARDS ? fanRotationSpeed : -fanRotationSpeed;
        if (fanRotationSpeed > 360) fanRotationSpeed = 0;
        else if (fanRotationSpeed < 0) fanRotationSpeed = 360;
    }
}
```

e) **Tellurion**: Press T to rotate the tellurion. There are 3 levels of speed of the tellurion.

```
void checkTellurionRotation(){
    if (ballAnimation == FORWARDS || ballAnimation ==  BACKWARDS)
    {
        ballRotation += ballAnimation == FORWARDS ? ballRotationSpeed : -ballRotationSpeed;
        if (ballRotationSpeed > 360) ballRotationSpeed = 0;
        else if (ballRotationSpeed < 0) ballRotationSpeed = 360;
    }
}
```

f) **Basketball**: Press B to bounce the basketball. Bouncing is based on the gravity and friction, which makes the bouncing more smooth and real.

```
void checkBasketballRotation(){
    if (ballStart) {
        ballCurrentSpeed = ballStartSpeed;
        ballBouncing = true;
        ballStart = false;
    }
    if (ballBouncing) {
        ballCurrentSpeed -= gravity;
        if (relativeHeight + ballCurrentSpeed <= 0) {
            // collide with ground
            relativeHeight = 0;
            ballCurrentSpeed += resistence;
            if (ballCurrentSpeed >= 0) {
                // stop bouncing
                ballBouncing = false;
                ballCurrentSpeed = 0;
                relativeHeight = 0;
            } else {
                ballCurrentSpeed *= -1.0;
            }
        } else {
            relativeHeight += ballCurrentSpeed;
        }
```

g) **Clock**: Clock can show the local time. I use ‘TimerFunction’ function to get the real time.

```
void TimerFunction(int value)
{
    struct timeb tb;
    time_t tim=time(0);
    struct tm* t;
    t=localtime(&tim);
    ftime(&tb);

    angleSec = (float)(t->tm_sec+ (float)tb.millitm/1000.0f)/30.0f * M_PI;
    angleMin = (float)(t->tm_min)/30.0f * M_PI + angleSec/60.0f;
    angleHour = (float)(t->tm_hour > 12 ? t->tm_hour-12 : t->tm_hour)/6.0f * M_PI + angleMin/12.0f;

    glutPostRedisplay();
    glutTimerFunc(33,TimerFunction, 1);
}
```

3. Textures

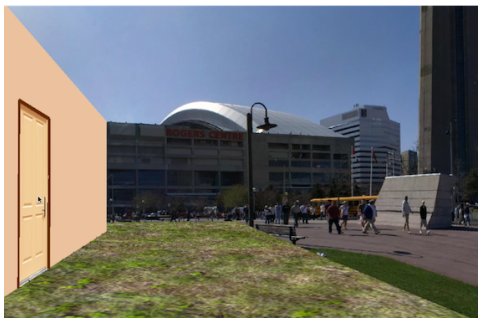
a) Texture images are read by ‘loadTexture’ and ‘setupTexture’ function from pmba[2]. All the texture images are kept in ‘texture’ folder.

[2] https://github.com/pmba/opengl-room

```cpp
void loadTexture(const char* fileName, Texture* texture)
{
    // Generate Texture
    glBindTexture(GL_TEXTURE_2D, texture->id);
    // set the texture wrapping/filtering options (on the currently bound texture object)
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    // load and generate the texture
    int nrChannels;

    texture->data = stbi_load(fileName, &texture->width, &texture->height, &nrChannels, 0);

    if (!texture->data) {
        cout << "Failed to load texture: " << fileName << endl;
        exit(1);
    }
```

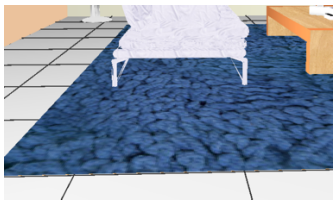
```cpp
void setupTexture(Texture* texture)
{
    glPixelStoref(GL_UNPACK_ALIGNMENT, 1);

    glGenTextures(1, &texture->id);
    glBindTexture(GL_TEXTURE_2D, texture->id);

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

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, texture->width,
                    texture->height, 0, GL_RGBA, GL_UNSIGNED_BYTE,
                    texture->data);
}
```

b) Ground & Sky




c) Door

d) Carpet



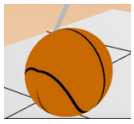
e) Tea table & Books

```cpp
glEnable(GL_TEXTURE_2D);

glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glBindTexture(GL_TEXTURE_2D, texture->id);

glColor3fv(glm::value_ptr(color));
```

f) Murals

g) Table & Tellurion & Bookshelf

h) Basketball



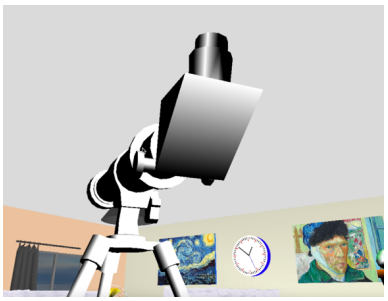
i) TV & TV Dresser & Loudspeaker box

j) **Couch & Pillow**

4. Different Viewpoints

a) In this program, there is a camera object, and you can change the camera view by pressing ‘W’, ‘A’, ‘S’, ‘D’, ‘up’, ‘left’, ‘right’, ‘down’ on the keyboard. Also, use mouse can change the viewpoint.

b) In addition, press ‘R’ to change to the telescope viewpoint directly. Press ‘E’ to change to the couch viewpoint directly.

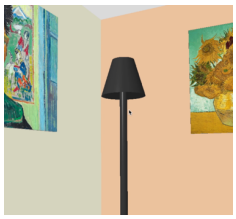


c) Press ‘X/Z’ on the keyboard to tilt scene.

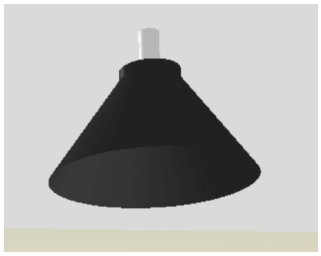
5. Lightning Effects

```
static bool light1_enabled          = true;
static GLfloat light1_offset[]       = { 8.5, 5, -8.5 };
static GLfloat light1_ambient[]       = { 0.2, 0.2, 0.2, 1.0 };
static GLfloat light1_diffuse[]       = { 1.0, 1.0, 1.0, 1.0 };
static GLfloat light1_specular[]    = { 1.0, 1.0, 1.0, 1.0 };
static GLfloat light1_position[]    = { 0.0, 3.0, 0.0, 1.0 };
static GLfloat light1_direction[]    = { 0.0, -1.0, 0.0 };
static GLfloat light1_angle           = 20.0;
static GLfloat light1_exponent        = 2.0;
```

a) **Lamp**: Press L to turn on/off the lamp.




b) **Ceiling Lamp**: Press K to turn on/off the celling lamp.




c) **Sunlight**: Press J to turn on/off the sunlight.

6. Creative Ideas

a) **Change the viewpoint**: Press R to change to the telescope viewpoint directly. Press E to change to the couch viewpoint directly.

b) **Lamp can turn on/off**: Press K to turn on/off the celling lamp. Press L to turn on/off the lamp. Set the switch on the light to make it more in line with the furniture requirements.

c) **Fan has several levels of speed**: Press 'V' can turn on/off the fan and change the speed of it. Set the speed level on the fan to make it more in line with the furniture requirements

d) **Basketball bouncing is based on the gravity and friction**: When pressing 'B' to bounce the basketball, it will firstly go up. Due to the gravity, the speed of the ball will decrease, and the ball drop down. When it reaches the ground, it will speed up again. But it will reduce the energy due to the friction. After several rounds, it will keep stay on the ground. I add the gravity and friction to make the ball bounce more smooth and real.

e) **Tellurion has several levels of speed**: Press 'T' can rotate tellurion and change the speed of it.

f) **Change the environment**: Press 'Y' to change the environment between the urban and rural area, which makes the living environment more vivid.

7. Code Structure

a) Use object-oriented programming.

b) All the textures are kept in the 'texture' folder. All the objects are kept in the 'data' folder. All the additional files, e.g., obj_loader.h are kept in the 'lib' folder.

c) Project structure