Graphics-Final-1

Graphics Final Project

Group Id (15)

Name	sec	Bn Number
Yara Metwally	2	43
Samar Ibrahim	1	40
Shymaa Gamal	2	1
Sondos Mohamed	1	42
Poline Atef	1	23

Initializing Variables:

Here we determine our variables and load our objects(flower,table,stool) and we give the objecs variavle (q,w,e) to make the robot move ad interact with these objects.comes later on the code,and give initiaizations values to the camera

```
#include <math.h>
#include <GL/glut.h>
#include "glm.h"
#include "imageloader.h"

int windowWidth = 1000;
int windowHeight = 1000;
float aspect = float(windowWidth) / float(windowHeight);

static int shoulder = 0, shoulder2 = 0, shoulder3 = 0, elbow = 0, fingerBase = 0,
fingerUp = 0, rhip = 0, rhip2 = 0, rknee = 0, lknee = 0, lhip = 0, lhip2 = 0;
static int flower=0;
static float x=0;
static int angle=0;
int press=0;
int pressm=0;
//flower
 float q=0.0;
 float w=0.0;
 float e=0.0;
 //  z directio
 float r=0.0;

float DRot = 90;
float Zmax, Zmin;
GLMmodel* pmodel;
```

```
float VRot =0.0;


GLMmodel* pmodel1 =glmReadOBJ("data/flowers.obj");
GLMmodel* pmodel2 =glmReadOBJ("data/table.obj");
GLMmodel* pmodel3 =glmReadOBJ("data/stool.obj");
double eye[] = { .03, .3,.1 };
double center[] = { 0, 0, -2};
double up[] = { 0, 1, 0 };
```

Colors Effect:

Here we assign the color of the source using ambient to choose the color , diffuse to make it more real and rigg it in 3-d plane finally specular to add some light on it ,when we change these parameters the colors ad lights effect will change

```
 // RGBA
GLfloat light_ambient2[] = { 1.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse2[] = { 1.0, 0.0, 0.0,1.0 };
GLfloat light_specular2[] = {1.0, 1.0, 1.0, 1.0 };
// x , y, z, w
GLfloat light_position2[] = {0.5,0.5, 0.5, 1.0 };


// RGBA
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 0.0 };
GLfloat light_diffuse[] = { 0.5, 0.5, 0.5,1.0 };
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0 };
// x , y, z, w
GLfloat light_position[] = {0.5,5.0, 0.0, 1.0 };
GLfloat lightPos1[] = {-0.5,-5.0,-2.0, 1.0 };

// Material Properties
GLfloat mat_amb_diff[] = {0.643, 0.753, 0.934, 1.0 };
GLfloat mat_specular[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat shininess[] = {100.0 };
```

the second part (Texture)

for the texture part ,we will apply the texture through four steps:

1. Enable Texture : the function is glEnable(GL_TEXTURE_2D) ,as TEXTURE_2D is the parameter .
2. Load Texture : the goal of this step is to load the texture image to the scene using the function of _textureId=loadTexture(image).
3. Bind Texture : the goal of this step is to select our loaded texture through the function of glBindTexture(GL_TEXTURE_2D, _textureId),where GL_TEXTURE_2D is the target and _textureId is the value returned by the load texture function in step 2.

4. Mapping Texture Coordinates : the goal of this step is to map each vertex in the texture to a specific vertex in the polygon through the function of glTexCoord2f(0.0f, 0.0f) that we call before each vertex .

   Repeating Textures :

   to repeat the texture on the same ploygon through the function :

   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST),glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR).

   ## conclusion :

   in this project we dealt with some valuable topics in computer graphics ,like transformation,texture mapping, Lightening an coloring to implement an application that may help in the biomedical field,specifically in the simulation of the body organs or creating animations in medical training .

```
 //Makes the image into a texture, and returns the id of the texture
GLuint loadTexture(Image* image) {
      GLuint textureId;
      glGenTextures(1, &textureId); //Make room for our texture
      glBindTexture(GL_TEXTURE_2D, textureId); //Tell OpenGL which texture to edit
      //Map the image to the texture
      glTexImage2D(GL_TEXTURE_2D,                //Always GL_TEXTURE_2D
                               0,                            //0 for now
                               GL_RGB,                       //Format OpenGL uses for
image
                               image->width, image->height,  //Width and height
                               0,                            //The border of the image
                               GL_RGB, //GL_RGB, because pixels are stored in RGB
format
                               GL_UNSIGNED_BYTE, //GL_UNSIGNED_BYTE, because pixels
are stored
                                                 //as unsigned numbers
                               image->pixels);               //The actual pixel data
      return textureId; //Returns the id of the texture
}

GLuint _textureId; //The id of the texture
GLuint _textureId1; //The id of the texture
```

Draw Function (Part1):

Here we import our three objects and draw them and assigining their scale and position using these functins: glmVertexNormals glmScale

```

void Draw_cube(GLdouble width, GLdouble height, GLdouble depth) // Draw function
{
   glPushMatrix();
   glScalef(width, height, depth);
   glutWireCube(1.0);
   glPopMatrix();
}
void init()
{
        glEnable(GL_LIGHTING);

        glEnable(GL_LIGHT2);

        glLightfv(GL_LIGHT2, GL_AMBIENT, light_ambient2);
        glLightfv(GL_LIGHT2, GL_DIFFUSE, light_diffuse2);
        glLightfv(GL_LIGHT2, GL_SPECULAR, light_specular2);
         GLfloat lightColor2[] = {1.0f, 1.0f,  1.0f, 1.0f };
        glLightfv(GL_LIGHT2, GL_DIFFUSE, lightColor2);
        glLightfv(GL_LIGHT2, GL_POSITION, lightPos1);

        // Enable Depth buffer
        glEnable(GL_DEPTH_TEST);
}

//flower
void drawmodel1(void)
{
        glmUnitize(pmodel1);
        glmFacetNormals(pmodel1);
        glmVertexNormals(pmodel1, 90.0);
        glmScale(pmodel1, .15);
        glmDraw(pmodel1, GLM_SMOOTH | GLM_MATERIAL);
}
//table
void drawmodel2(void)
{
        glmUnitize(pmodel2);
        glmFacetNormals(pmodel2);
        glmVertexNormals(pmodel2, 90.0);
        glmScale(pmodel2, .15);

    glmDraw(pmodel2, GLM_SMOOTH | GLM_MATERIAL);
}
void drawmodel3(void)
{
        glmUnitize(pmodel3);
        glmFacetNormals(pmodel3);
        glmVertexNormals(pmodel3, 90.0);
        glmScale(pmodel3, .15);

    glmDraw(pmodel3, GLM_SMOOTH | GLM_MATERIAL);
}
```

3-D Rendering

```
 //Initializes 3D rendering
void initRendering() {
          Image* image = loadBMP("image2.bmp");
          _textureId = loadTexture(image);
          delete image;
       // Turn on the power
        glEnable(GL_LIGHTING);
        // Flip light switch
        glEnable(GL_LIGHT0);
        glEnable(GL_LIGHT1);
        // assign light parameters
        glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
        glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
        glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse);
        glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular);
    // Material Properties
        glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE,mat_amb_diff);
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
    GLfloat lightColor1[] = {1.0f, 1.0f,  1.0f, 1.0f };
        glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
        glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor1);
        glEnable(GL_NORMALIZE);
        //Enable smooth shading
        glShadeModel(GL_SMOOTH);
        // Enable Depth buffer
        glEnable(GL_DEPTH_TEST);

}
```

Draw function (Part2)

we will draw and simulate the full body joints movement by applying transformations (translation,rotation,scaling),

```
  void draw_right_arm(void)
{
   glPushMatrix();

   glTranslatef(-2, 4.5, 0.0);
   glRotatef((-(GLfloat)shoulder) - 90, 0.0, 0.0, 1.0);
   glRotatef(180, 1.0, 0.0, 0.0);
   glTranslatef(1.0, 0.25, 0.0); //hena x,y
   glTranslatef(-1.0, 0.0, 0.5);
   glRotatef(-(GLfloat)shoulder2, 0.0, 1.0, 0.0);
   glTranslatef(1.0, 0.0, -0.5);
   glRotatef(-(GLfloat)shoulder3, 1.0, 0.0, 0.0);
   //glColor3f(0.0, 1.0, 0.0);
   Draw_cube(2.0f, 0.5f, 1.0f);
   //forearm
   glPushMatrix();
   glTranslatef(1.0, -0.25, 0.0);
   glRotatef((GLfloat)elbow, 0.0, 0.0, 1.0);
   glTranslatef(1.5, 0.25, 0.0);
   Draw_cube(3.0f, 0.5f, 1.0f);
   //Draw finger flang 1
   glPushMatrix();
   glTranslatef(1.5, 0.25, -0.4); // (b3mqal translation b3d el origin bta3 akher
shakl)
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 11
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger 2
   glPushMatrix();
   glTranslatef(1.5, 0.25, -0.2);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 22
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger flang 3
   glPushMatrix();
   glTranslatef(1.5, 0.25, 0.2);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 33
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger 4
   glPushMatrix();
   glTranslatef(1.5, 0.25, 0.4);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f); //law hagarb figer 4 bas ahot pop law harsem finger 44
ashel el pop
                                //Draw finger flang 44
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger flang 5
   glPushMatrix();
   glTranslatef(1.5, -0.25, 0.0);
   glRotatef(-(GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 55
   glPushMatrix();
   glTranslatef(0.25, -0.05, 0.0);
   glRotatef(-(GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
}

void draw_left_arm(void)
{

   glPushMatrix();
   glTranslatef(2, 4.5, 0.0);
   glRotatef(((GLfloat)shoulder) - 90, 0.0, 0.0, 1.0);
   glTranslatef(1.0, 0.25, 0.0);
   glTranslatef(-1.0, 0.0, -0.5);
   glRotatef((GLfloat)shoulder2, 0.0, 1.0, 0.0);
   glTranslatef(1.0, 0.0, 0.5);
   glRotatef((GLfloat)shoulder3, 1.0, 0.0, 0.0);
  // glColor3f(0.0, 1.0, 0.0);
   Draw_cube(2.0f, 0.5f, 1.0f);
   //forearm
   glPushMatrix();
   glTranslatef(1.0, -0.25, 0.0); //y,x
   glRotatef((GLfloat)elbow, 0.0, 0.0, 1.0);
   glTranslatef(1.5, 0.25, 0.0); //y,x
   Draw_cube(3.0f, 0.5f, 1.0f);
   //Draw finger flang 1
   glPushMatrix();
   glTranslatef(1.5, 0.25, -0.4); // (b3mqal translation b3d el origin bta3 akher
shakl)
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 11
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger 2
   glPushMatrix();
   glTranslatef(1.5, 0.25, -0.2);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 22
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger flang 3
   glPushMatrix();
   glTranslatef(1.5, 0.25, 0.2);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 33
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger 4
   glPushMatrix();
   glTranslatef(1.5, 0.25, 0.4);
   glRotatef((GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, -0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f); //law hagarb figer 4 bas ahot pop law harsem finger 44
ashel el pop
                                //Draw finger flang 44
   glPushMatrix();
   glTranslatef(.25, -0.05, 0.0);
   glRotatef((GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   //Draw finger flang 5
   glPushMatrix();
   glTranslatef(1.5, -0.25, 0.0);
   glRotatef(-(GLfloat)fingerBase, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   //Draw finger flang 55
   glPushMatrix();
   glTranslatef(0.25, -0.05, 0.0);
   glRotatef(-(GLfloat)fingerUp, 0.0, 0.0, 1.0);
   glTranslatef(0.25, 0.05, 0.0);
   Draw_cube(0.5f, 0.1f, 0.1f);
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
}

void draw_right_leg(void)
{
   glPushMatrix();
   glTranslatef(-0.5, -3.25, 0.0);
   glTranslatef(-0.75, 1.75, 0.0);
   glRotatef((GLfloat)rhip, 0.0, 0.0, 1.0);
   glTranslatef(0.5, -1.75, 0.0);
   glTranslatef(0.0, 1.75, -0.5);
   glRotatef((GLfloat)rhip2, 1.0, 0.0, 0.0);
   glTranslatef(0.0, -1.75, .5);
   Draw_cube(1.0f, 3.5f, 1.0f);
   glPushMatrix();
   glTranslatef(0.0, -3.25, 0.0);
   glTranslatef(0.0, 1.75, -0.5);
   glRotatef((GLfloat)rknee, 1.0, 0.0, 0.0);
   glTranslatef(0.0, -1.75, 0.5);
   Draw_cube(1.0f, 3.0f, 1.0f);
   glPushMatrix();
   glTranslatef(-0.05, -2, 0.0);
   glPushMatrix();
   glScalef(1.0, 1.0, 3.0);
   glutSolidCube(1);
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
}

void draw_left_leg(void)
```

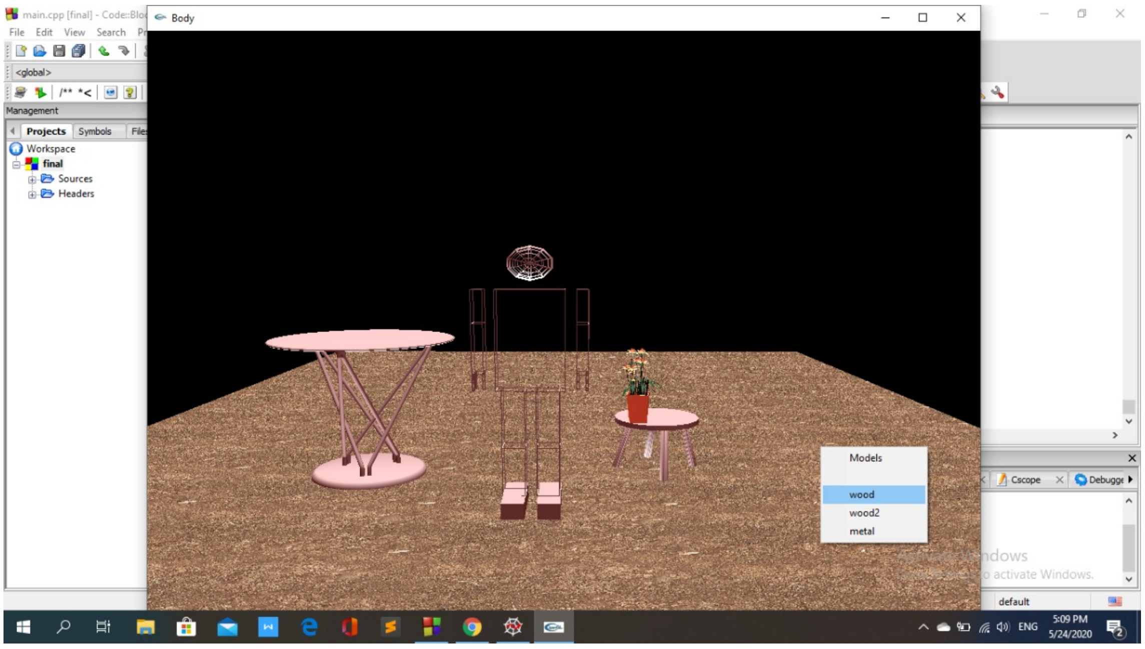
```
{
   //hena x bel negative nahet el ymen
   glPushMatrix();

   glTranslatef(.75, -3.25, 0.0);
   glTranslatef(.5, 1.75, 0.0);
   glRotatef((GLfloat)lhip, 0.0, 0.0, 1.0);
   glTranslatef(-0.5, -1.75, 0.0);
   glTranslatef(0.0, 1.75, -0.5);
   glRotatef((GLfloat)lhip2, 1.0, 0.0, 0.0);
   glTranslatef(0.0, -1.75, .5);
   Draw_cube(1.0f, 3.5f, 1.0f);
   glPushMatrix();
   glTranslatef(0.0, -3.25, 0.0);
   glTranslatef(0.0, 1.75, -0.5);
   glRotatef((GLfloat)lknee, 1.0, 0.0, 0.0);
   glTranslatef(0.0, -1.75, 0.5);
   Draw_cube(1.0f, 3.0f, 1.0f);
   glPushMatrix();
   glTranslatef(0.0, -2, 0.0);
   glPushMatrix();
   glScalef(1.0, 1.0, 3.0);
   glutSolidCube(1);
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
   glPopMatrix();
}
```

Choosing Different Grounds :

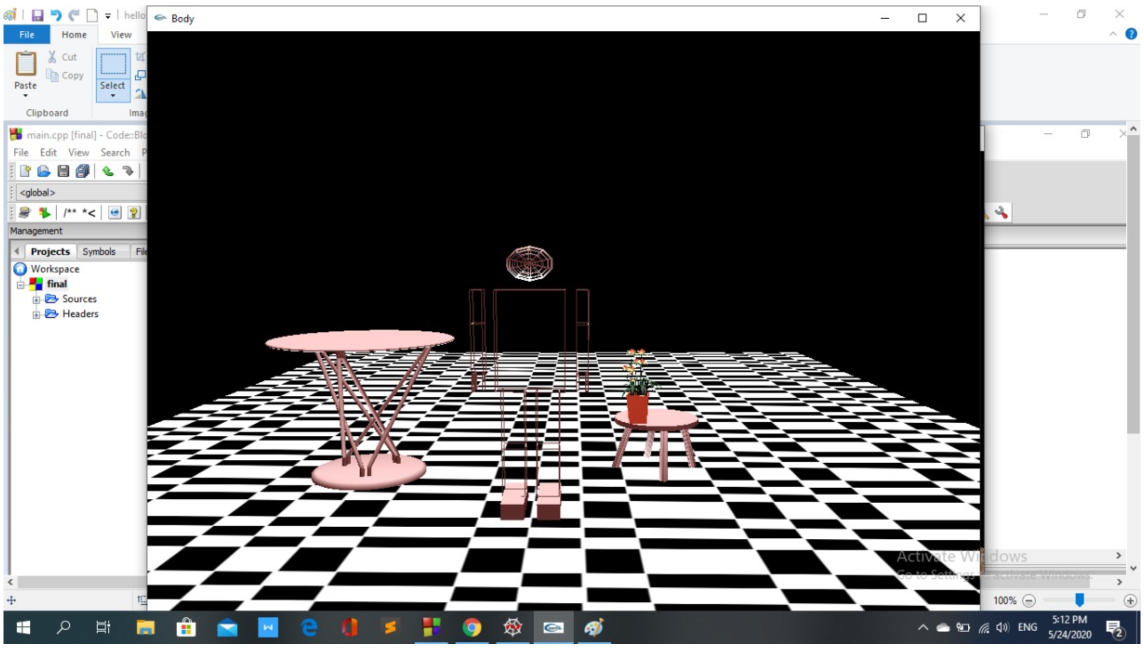
```
 void screen_menu(int value)
{
    char* name = 0;

    switch (value) {
    case 'a':
        name = "image.bmp";
        break;
    case 's':
        name = "image2.bmp";
        break;
    case 'd':
        name = "image3.bmp";
        break;

    }
    if (name) {
        Image* image = loadBMP(name);
        _textureId=loadTexture(image);
      if (!image) exit(0);
    }
     glutPostRedisplay();

}
```

Different ground :

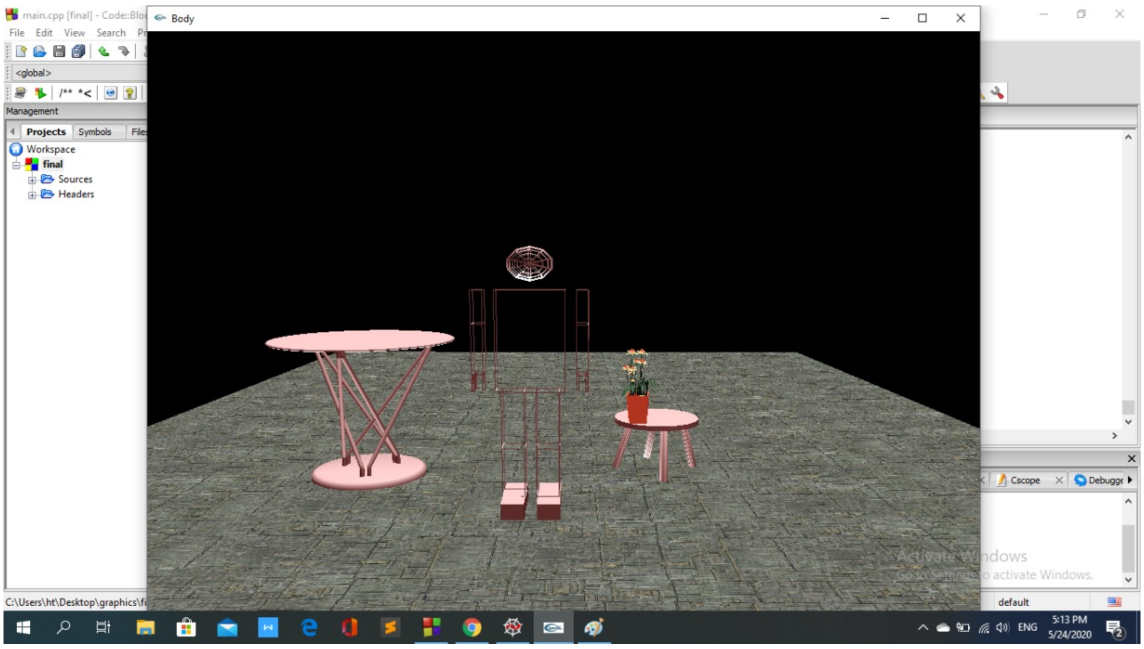
When we first run our code the default background we choose is (brown wood plane) background by clicking on the ground we can choose different ground , if we want to change it we click on the mouse we got (wood2) (chess background)



Brown Background

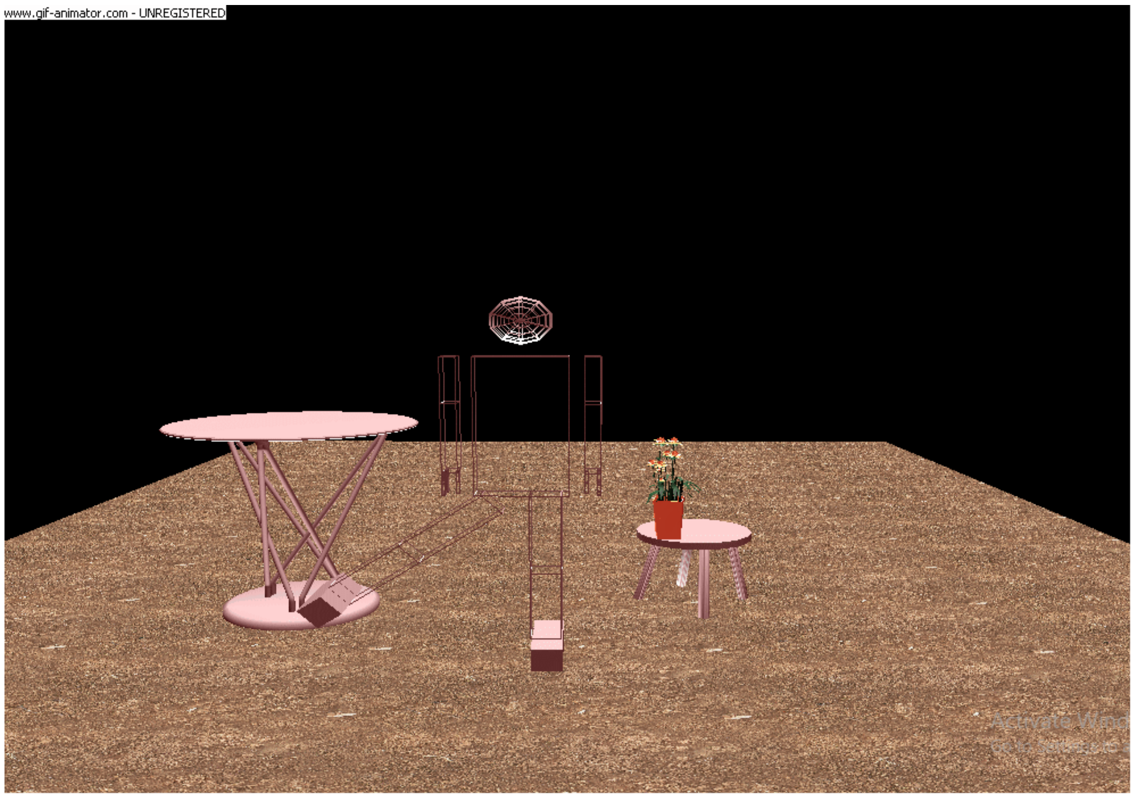


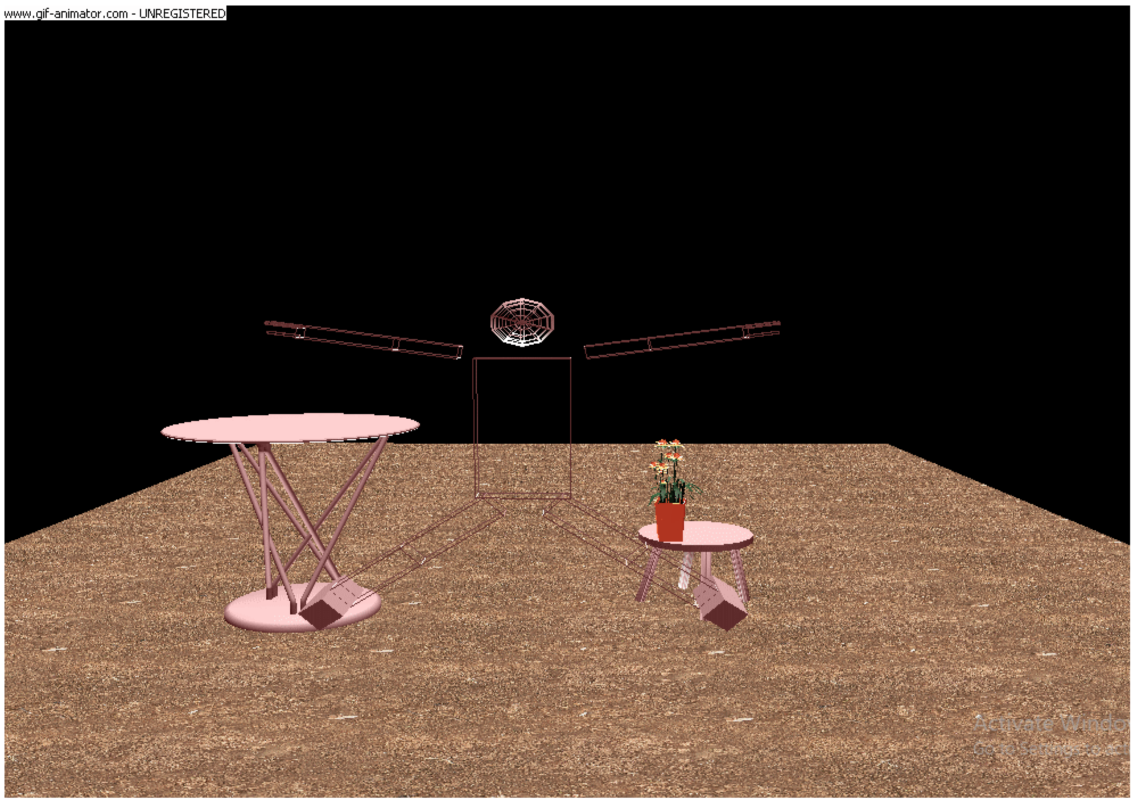
Chess Background

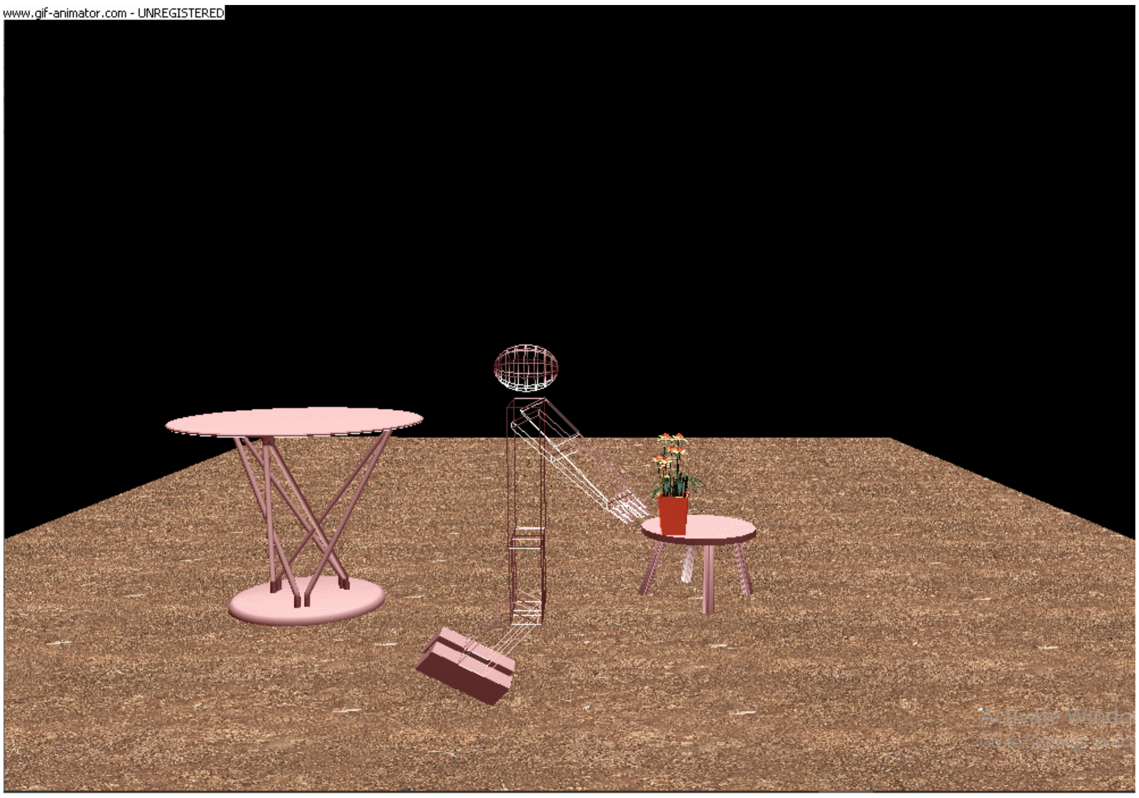


Grey Background

Animation :


www.gif-animator.com - UNREGISTERED


www.gif-animator.com - UNREGISTERED



Animation code :

```
 void Timer(int x){
    // Refresh and redraw
    glutPostRedisplay();
    glutTimerFunc(50, Timer, 0);
}




//////////////////////////////?????
void correct()
{
    double speed = 0.001;
    if (eye[0]>0)
    {
        eye[0] -= speed;
        center[0] -= speed;
    }
    else
    {
        eye[0] += speed;
        center[0] += speed;
    }

    if (DRot == 0)
    {
        if ((eye[2] >= -2 && eye[2] <= 2) || eye[2]>0)
        {
            eye[2] -= speed;
            center[2] -= speed;
        }
        else
        {
            eye[2] += speed;
            center[2] += speed;
        }
    }
    else
    {
        if (eye[2]>0)
        {
            eye[2] -= speed;
            center[2] -= speed;
        }
        else
        {
            eye[2] += speed;
            center[2] += speed;
        }
    }

}
void SetBound()
{
    if (DRot == 0 || eye[0]> 0.15 || eye[0]< -0.15)
    {
        if (eye[2] >= -1)
        {
            Zmax = 0.7;
            Zmin = -0.8;
        }
        else
        {
            Zmax = -1.2;
            Zmin = -2.4;
        }
    }
    else
    {
        Zmax = 0.7;
        Zmin = -2.4;
    }
}
```

```
void DTimer1(int x){

    DRot -= 1;
    if (DRot == 0)
        return;
    glutPostRedisplay();
    glutTimerFunc(30, DTimer1, 0);


}

void DTimer2(int x){
    DRot += 1;
    if (DRot == 90)
        return;
    glutPostRedisplay();
    glutTimerFunc(30, DTimer2, 0);
}

 // penguin Dance
   int check =1;

  int check2=1;

void timer(int value)
{
   int m=1;
   int l=30;
   int z=20;
  int k=10;
    if (value == 0)
    {

          if (lhip < 50)
      {
         lhip = (lhip + 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer, 0);
      }
       else
       glutTimerFunc(l, timer, 1);
    }


    else if (value == 1)
    {


         if (lhip >= 5)
         {
            lhip = (lhip - 5) % 360;
            glutPostRedisplay();
            glutTimerFunc(l, timer, 1);

         }

        else
         {
            check=check+1;
            if(check<=2)
            glutTimerFunc(l, timer, 0);
            else
            glutTimerFunc(l, timer, 2);

         }

    }


    else if (value == 2)
    {

       if (rhip > -50)
      {
         rhip = (rhip - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer, 2);
      }
       else
       glutTimerFunc(l, timer, 3);
    }

    else if (value == 3)
    {
       if (rhip <= -5)
      {
         rhip = (rhip + 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer, 3);
      }
        else
         {
            check2=check2+1;
            if(check2<=2)
            glutTimerFunc(l, timer, 2);
            else
            glutTimerFunc(k, timer, 4);
         }
    }

      if (value == 4)
    {
       if (lhip2 < 90 && lknee > -90)
      {
         lhip2 = (lhip2 + 5) % 360;
         rhip2 = (rhip2 + 5) % 360;
         lknee = (lknee - 5) % 360;
         rknee = (rknee - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(k, timer, 4);
      }
     else

        glutTimerFunc(k, timer, 5);
     }

    else if (value == 5)
     {
      if (lhip2 >=0  && lknee < 0 )
      {
            x=x-0.1;
            lhip2 = (lhip2 - 5) % 360;
            rhip2 = (rhip2 - 5) % 360;
            lknee = (lknee + 5) % 360;
            rknee = (rknee + 5) % 360;
           glutPostRedisplay();
           glutTimerFunc(k, timer, 5);
        }
         else
         glutTimerFunc(z, timer,6 );
      }


 if (value == 6)
    {
       if (lhip2 < 90 && lknee > -90)
      {
         lhip2 = (lhip2 + 5) % 360;
         rhip2 = (rhip2 + 5) % 360;
         lknee = (lknee - 5) % 360;
         rknee = (rknee - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(z, timer, 6);
      }
     else

        glutTimerFunc(z, timer, 7);
     }


    else if (value == 7)
     {
      if (lhip2 >= 0  && lknee < 0 )
      {
            x=x+0.1;
            lhip2 = (lhip2 - 5) % 360;
            rhip2 = (rhip2 - 5) % 360;
            lknee = (lknee + 5) % 360;
            rknee = (rknee + 5) % 360;
           glutPostRedisplay();
           glutTimerFunc(z, timer, 7);

      }
       else

        glutTimerFunc(m, timer, 8);
     }

      if (value == 8)
    {
       if (lhip2 < 90 && lknee > -90)
      {
         lhip2 = (lhip2 + 5) % 360;
         rhip2 = (rhip2 + 5) % 360;
         lknee = (lknee - 5) % 360;
         rknee = (rknee - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(m, timer, 8);
      }
     else

        glutTimerFunc(m, timer, 9);
     }


    else if (value == 9)
     {
      if (lhip2 > 0  && lknee < 0 )
      {

        if(x<=5.2)
        {
            x=x+0.1;
            lhip2 = (lhip2 - 5) % 360;
            rhip2 = (rhip2 - 5) % 360;
            lknee = (lknee + 5) % 360;
            rknee = (rknee + 5) % 360;
           glutPostRedisplay();
           glutTimerFunc(m, timer, 9);
        }
      }
      else
         glutTimerFunc(m, timer, 8);

     }
```

```
}


void timer4(int value)
{
   int m=1;

         if (value == 0)
    {
       if (lhip2 < 90 && lknee > -90)
      {
         lhip2 = (lhip2 + 5) % 360;
         rhip2 = (rhip2 + 5) % 360;
         lknee = (lknee - 5) % 360;
         rknee = (rknee - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(m, timer4, 0);
      }
     else

        glutTimerFunc(m, timer4, 1);
     }
     else if (value == 1)
     {
      if (lhip2 > 3  && lknee < 3 )
      {

        if(x > .3)
        {
            x=x-0.1;
            lhip2 = (lhip2 - 5) % 360;
            rhip2 = (rhip2 - 5) % 360;
            lknee = (lknee + 5) % 360;
            rknee = (rknee + 5) % 360;
           glutPostRedisplay();
           glutTimerFunc(m, timer4, 1);
        }
      }
          else

        glutTimerFunc(m, timer4, 0);


     }
}

 // Doing activity

void timer2(int value)
{
   int l=50;

    if (value == 1)
    {

      if (  lhip < 60 && shoulder <= 150)
      {
         shoulder = (shoulder + 10) % 360;
         lhip = (lhip + 5) % 360;
         rhip = (rhip - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 1);
      }
       else
       glutTimerFunc(l, timer2, 2);
    }


    else if (value == 2)
    {

      if ( lhip >= 0 && shoulder >= 5 )
      {
         shoulder = (shoulder - 10) % 360;
         lhip = (lhip - 5) % 360;
         rhip = (rhip +5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 2);

      }
           else
       glutTimerFunc(l, timer2, 3);
    }


   else if (value == 3)
    {

      if (lhip2 < 90 && shoulder2 <= 160 )
      {
         shoulder2 = (shoulder2 + 5) % 360;
         lhip2 = (lhip2 + 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 3);

      }
           else
       glutTimerFunc(l, timer2, 4);
    }

   else if (value == 4)
    {
       if (shoulder2 >= 5 && lhip2 >= 0)
      {
         lhip2 = (lhip2 - 5) % 360;
         shoulder2 = (shoulder2 - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 4);

      }
           else
       glutTimerFunc(l, timer2, 5);
    }

       else if (value == 5)
    {

      if (rhip2 < 90 && shoulder2 <= 160 )
      {
         shoulder2 = (shoulder2 + 5) % 360;
         rhip2 = (rhip2 + 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 5);

      }
           else
       glutTimerFunc(l, timer2, 6);
    }

   else if (value == 6)
    {

       if (shoulder2 >= 5 && rhip2 >= 0)
      {
         rhip2 = (rhip2 - 5) % 360;
         shoulder2 = (shoulder2 - 5) % 360;
         glutPostRedisplay();
         glutTimerFunc(l, timer2, 6);

      }

    }

    }



 // animation with interaction

   void timer3(int value)
{
   int l=70;
```

```
     if (value == 1)
     {
        angle=90;
        if ( lknee > -90 && shoulder2 < 90 && shoulder > -30 && fingerBase > -20 )
       {


          if(r == 0 || r >= -6.0)
          {
          lknee = (lknee - 5) % 360;
          rknee = (rknee - 5) % 360;
          shoulder2 = (shoulder2 + 5) % 360;
          shoulder = (shoulder -3) % 360;
          fingerBase = (fingerBase - 1) % 360;
          glutPostRedisplay();
          glutTimerFunc(l, timer3, 1);
            r=r-.01;
          }
       }


      else

         glutTimerFunc(l, timer3, 2);
      }

     else if (value == 2)
      {
       if ( lknee < 0  && fingerBase > -20)
       {
          fingerBase = (fingerBase - 1) % 360;
          lknee = (lknee + 5) % 360;
          rknee = (rknee + 5) % 360;
          glutPostRedisplay();
          glutTimerFunc(l, timer3, 2);
          // w=.3;

          w=w+.01;
          r=r+0.01;
       }
       else

         glutTimerFunc(l, timer3, 3);

      }

    else if (value == 3)
      {
       if ( angle > -60 )
       {
          q=q+.0019;
          e=e-.003;
          angle = (angle - 5) % 360;
          flower = (flower - 5) % 360;
          glutPostRedisplay();
          glutTimerFunc(l, timer3, 3);
       }
       else

         glutTimerFunc(l, timer3, 4);

      }
  else if (value == 4)
      {
       if ( shoulder2  < 90 && fingerBase > -30)
       {
          w=w+.01;
          q=q-.003;
          fingerBase = (fingerBase - 1) % 360;
          shoulder2= (shoulder2 + 5) % 360;
          glutPostRedisplay();
          glutTimerFunc(l, timer3, 4);
       }
       else

         glutTimerFunc(l, timer3, 5);

      }
        else if (value == 5)
      {
       if ( angle <= 0 && fingerBase < 0 && shoulder2  >0 && shoulder <0)
       {

          fingerBase = (fingerBase + 3) % 360;
           shoulder2= (shoulder2 - 9) % 360;
            shoulder = (shoulder +3) % 360;
          angle = (angle + 6) % 360;
          glutPostRedisplay();
          glutTimerFunc(l, timer3, 5);
       }

      }


 }
```

Key board function

```
void Keyboard(unsigned char Key, int x, int y){
    switch (Key)
    {
    case 27:
        exit(0);

        break;

    case ' ':
        if (DRot == 0 || DRot == 90)
        {
            if (DRot)
                DTimer1(0);
            else
                DTimer2(0);
        }
        break;

    case 'p':
   press=press+1;
   if(press == 1 )
   {

         glutTimerFunc(0, timer, 0); //penguin
   }
   else if (press == 2 && pressm==0)
   {
      glutTimerFunc(0, timer4, 0);
      glutTimerFunc(4500, timer, 0);
   }
   else {
      break;
   }
      break;
   case 'l':
       glutTimerFunc(0, timer2, 1);  // Activity

      break;

   case 'm':
      if(press==0 && pressm == 0)
      {
          pressm=1;
       glutTimerFunc(0, timer3, 1);  // Activity
      }
      else if(press != 0 && pressm == 0)
      {
         pressm=1;
         glutTimerFunc(0, timer4, 0);
          glutTimerFunc(2000, timer3, 1);

      }
      else if(pressm != 1)
      {
           break;
```

```
      }
      break;

    default:
        break;
    }
}
```

Main Function

```
int main (int argc, char** argv)
{

    glutInit(&argc, argv);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Body");
    initRendering();

    glMatrixMode(GL_PROJECTION);
    gluPerspective(60, aspect, 0.1, 10);



    glutDisplayFunc(display);
   init();
    glutKeyboardFunc(Keyboard);
    Timer(0);
    glutCreateMenu(screen_menu);

    glutAddMenuEntry("Models", 0);
    glutAddMenuEntry("", 0);
    glutAddMenuEntry("wood2", 's');
    glutAddMenuEntry("wood", 'a');
    glutAddMenuEntry("metal", 'd');

    glutAttachMenu(GLUT_RIGHT_BUTTON);
    //glutTimerFunc(0,Timer1,0);
    glutMainLoop();
    return 0;

}
```

Issues

- we had an issue with the glm header . we had an errot(cannot open include file:glm.h:No such file or directory) and we solved this issue byextracting the glm code directory to our project directory , then we added the full path of the glm directory to :=> right click on project in the solution viewer => from the drop down menu choose properties => C\C++ => General => Additional Include Directories. and finally we added the full path in the edit box of Additional Include Directories.

- we also had an issue in uploading the objects from the data folder, and had an error of cannot open imageloader.h file and we solved this problem by solving CMake installation problem and adding it to the system path variables.