**Tema proiect:**  Where’s Wally? Social Distancing Edition

**Autori:**Dumitrașcu Andrada-Gabriela , grupa 353

**Descriere:**

Proiectul pe care l-am realizat a fost inspirat de jocul “Where’s Wally?”, în care jucătorul trebuie să găsească un personaj numit Waldo într-o mulțime de oameni. Am updatat acest joc și am creat versiunea de distanțare socială, unde oamenii poartă măști. Spre deosebire de proiectul 2D, nu mai sunt incluse animațiile, iar proiectul nu mai este structurat sub formă de joc.

**Convertirea proiectului 2D in proiect 3D:** indicati sub forma de tabel cum au fost transformate primitivele 2D in primitive 3D. Exemplu:

|  |  |
| --- | --- |
| **Primitiva 2D** | **Primitiva 3D** |
| glRecti (dreptunghiuri) pentru cap, brațe, trunchi și picioare | glutSolidCube (cuburi) |
| glRecti pentru nor | glutSolidSphere |

**Alte aspecte punctuale:**indicati , conform grilei de evaluare ce ati mai utilizat la proiectul 3D (iluminare, texturi etc). Daca va este mai usor, informatiile pot fi organizate intr-un tabel. Exemplu:

1. *Textură de tip iarbă aplicata pe primitiva GL\_QUADS*
2. *Ceață aplicată în mediu (amestecare)*
3. *Iluminare cu păstrarea culorii originale*
4. *Deplasarea observatorului în scenă*

**Originalitate:**

Consider că proiectul este original deoarece a preluat ideea unui joc vechi, de revistă, și l-a transpus pe calculator. De asemenea, proiectul este plasat în actualitate datorită tematicii de distanțare socială. Mi-a fost relativ ușor să modific obiectele 2D în obiecte 3D.

**Contributii individuale:**

Am realizat proiectul de una singură.

**Resurse utilizate: indicati resursele utilizate (material curs, tutoriale, etc.)**

Am utilizat materialele de curs și codurile sursă pentru laborator / curs.

**Link GoogleDrive (sau OneDrive UB):**

https://drive.google.com/file/d/1r2s5hmIa5ymWjol2-jK2a8YGXPpz7s2h/view?usp=sharing

**Anexa cod**

//SURSA: lighthouse3D: http://www.lighthouse3d.com/tutorials/glut-tutorial/keyboard-example-moving-around-the-world/

#include<gl/freeglut.h>

#include<math.h>

#include "SOIL.h"

// angle of rotation for the camera direction

float angle = 0.0;

// actual vector representing the camera's direction

float lx = 0.0f, lz = -1.0f;

// XZ position of the camera

float x = 0.0f, z = 5.0f;

float fraction = 0.1f;

int keybBackground, menuBackground;

#define imageWidth 320

#define imageHeight 160

GLubyte image[3 \* imageWidth \* imageHeight];

GLuint texture1;

static GLint fogMode;

void changeSize(int w, int h)

{

// Prevent a divide by zero, when window is too short

// (you cant make a window of zero width).

if (h == 0)

h = 1;

float ratio = w \* 1.0 / h;

// Use the Projection Matrix

glMatrixMode(GL\_PROJECTION);

// Reset Matrix

glLoadIdentity();

// Set the viewport to be the entire window

glViewport(0, 0, w, h);

// Set the correct perspective.

gluPerspective(45.0f, ratio, 0.1f, 100.0f);

// Get Back to the Modelview

glMatrixMode(GL\_MODELVIEW);

}

void LoadTexture(void)

{

GLuint texture;

glGenTextures(1, &texture);

glBindTexture(GL\_TEXTURE\_2D, texture);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP); // Set texture wrapping to GL\_REPEAT

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

int width, height;

//unsigned char\* image = SOIL\_load\_image("text\_smiley\_face.png", &width, &height, 0, SOIL\_LOAD\_RGB);

unsigned char\* image = SOIL\_load\_image("grass.png", &width, &height, 0, SOIL\_LOAD\_RGB);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image);

// SOIL\_free\_image\_data(image);

// glBindTexture(GL\_TEXTURE\_2D, 0);

}

void drawMan() {

GLfloat no\_mat[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat alb[] = { 1.0, 1.0, 1.0, 0.0 };

GLfloat negru[] = { 0.0, 0.0, 0.0, 0.0 };

GLfloat portocaliu[] = { 1.0, 0.5, 0.5, 0.0 };

GLfloat maro[] = { 0.5f, 0.3, 0.16f, 0.0 };

GLfloat rosu[] = { 1.0, 0.0, 0.0, 0.0 };

GLfloat albastru[] = { 0.6f, 0.8f, 0.8f, 0.0 };

GLfloat crem[] = { 1.0f, 0.8f, 0.7f, 0.0 };

// Brown pants

glColor3f(0.5f, 0.3, 0.16f);

//left leg

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, maro);

glPushMatrix();

glTranslatef(0.0f, 0.25f, 0.0f);

glScalef(0.5, 1.0, 1.0);

//glRotatef(90.0f, 1.0f, 0.0f, 0.0f);

//glRecti(0.0f, 0.0f, 1.0f, 1.0f);

glutSolidCube(0.5f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//right leg

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, maro);

glPushMatrix();

glTranslatef(0.4f, 0.25f, 0.0f);

glScalef(0.5, 1.0, 1.0);

glutSolidCube(0.5f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//shirt

glColor3f(1.0f, 0.0f, 0.0f);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, rosu);

//body

glPushMatrix();

glTranslatef(0.2f, 0.85f, 0.0f);

//glScalef(0.5, 1.0, 1.0);

glutSolidCube(0.7f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//left arm

glColor3f(1.0f, 1.0f, 1.0f);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(-0.2f, 0.85f, 0.0f);

glScalef(0.3, 0.9, 0.4);

glutSolidCube(0.7f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//right arm

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(0.65f, 0.85f, 0.0f);

glScalef(0.3, 0.9, 0.4);

glutSolidCube(0.7f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//head

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, crem);

glColor3f(1.0f, 0.8f, 0.7f);

glPushMatrix();

glTranslatef(0.2f, 1.5f, 0.0f);

//glScalef(0.3, 0.3, 1.0);

glutSolidCube(0.4f);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

//mask

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, albastru);

glColor3f(0.6f, 0.8f, 0.8f);

glPushMatrix();

glTranslatef(0.05f, 1.35f, 0.25f);

glScalef(0.3, 0.15, 0.0);

glRecti(0.0, 0.0, 1.0, 1.0);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

}

void drawCloud()

{

GLfloat alb[] = { 1.0, 1.0, 1.0, 0.0 };

GLfloat no\_mat[] = { 0.0, 0.0, 0.0, 1.0 };

//cloud

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(-5.0f, 8.0f, -30.0f);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(2.0, 100, 10);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(-3.0f, 8.0f, -30.0f);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(2.5, 100, 10);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(0.0f, 8.0f, -30.0f);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(3.0, 100, 10);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glPushMatrix();

glTranslatef(3.0f, 8.0f, -30.0f);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(2.5, 100, 10);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

}

void renderScene(void) {

// Clear Color and Depth Buffers

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

GLfloat pozitial0[] = { 1.0, 20.0, 5.0, 1.0 };

GLfloat alb[] = { 1.0, 1.0, 1.0, 0.0 };

GLfloat albastru1[] = { 0.3f, 0.8f, 0.8f, 0.0 };

GLfloat albastru2[] = { 0.4f, 0.7f, 0.9f, 0.0 };

GLfloat albastru3[] = { 0.3f, 0.7f, 0.8f, 0.0 };

GLfloat no\_mat[] = { 0.0, 0.0, 0.0, 1.0 };

// sursa de lumina

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_POSITION, pozitial0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, alb);

glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.2);

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.1);

glLightf(GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, 0.2);

glEnable(GL\_FOG);

{

GLfloat fogColor[4] = { 0.5, 0.5, 0.5, 1.0 };

fogMode = GL\_EXP;

glFogi(GL\_FOG\_MODE, fogMode);

glFogfv(GL\_FOG\_COLOR, fogColor);

glFogf(GL\_FOG\_DENSITY, 0.01);

glHint(GL\_FOG\_HINT, GL\_DONT\_CARE);

glFogf(GL\_FOG\_START, 1.0);

glFogf(GL\_FOG\_END, 50.0);

}

if (keybBackground == 1)

glClearColor(0.3f, 0.8f, 0.8f, 0.0);

else if (keybBackground == 2)

glClearColor(0.4f, 0.7f, 0.9f, 0.0);

else if (keybBackground == 3)

glClearColor(0.3f, 0.7f, 0.8f, 0.0);

// Reset transformations

glLoadIdentity();

// Set the camera

gluLookAt(x, 1.0f, z, x + lx, 1.0f, z + lz, 0.0f, 1.0f, 0.0f);

// ground

glEnable(GL\_TEXTURE\_2D);

LoadTexture();

glBegin(GL\_QUADS);

glColor3f(1.0, 1.0, 1.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, alb);

glTexCoord2f(1.0, 1.0); glVertex3f(-100.0f, 0.0f, -100.0f);

glTexCoord2f(1.0, 0.0); glVertex3f(-100.0f, 0.0f, 100.0f);

glTexCoord2f(0.0, 0.0); glVertex3f(100.0f, 0.0f, 100.0f);

glTexCoord2f(0.0, 1.0); glVertex3f(100.0f, 0.0f, -100.0f);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, no\_mat);

glEnd();

glDisable(GL\_TEXTURE\_2D);

// Draw 36 Men

for (int i = -3; i < 3; i++)

for (int j = -3; j < 3; j++) {

glPushMatrix();

glTranslatef(i \* 10.0, 0, j \* 10.0);

drawMan();

glPopMatrix();

}

drawCloud();

glutSwapBuffers();

}

void processNormalKeys(unsigned char key, int x, int y)

{

switch (key) {

case 'a':

angle -= 0.01f;

lx = sin(angle);

lz = -cos(angle);

break;

case 'd':

angle += 0.01f;

lx = sin(angle);

lz = -cos(angle);

break;

case 's':

x -= lx \* fraction;

z -= lz \* fraction;

break;

case 'w':

x += lx \* fraction;

z += lz \* fraction;

break;

}

if (key == 27)

exit(0);

}

void Sky(int key)

{

switch (key)

{

case 1:

//glClearColor(0.0, 0.0, 0.0, 1.0);

keybBackground = 1;

break;

case 2:

//glClearColor(1, 0.0, 0.0, 0.0);

keybBackground = 2;

break;

default:

//glClearColor(0.0, 1.0, 0.0, 0.0);

keybBackground = 3;

break;

}

}

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

// init GLUT and create window

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(100, 100);

glutInitWindowSize(320, 320);

glutCreateWindow("Scena 3D");

Sky(keybBackground);

// register callbacks

glutDisplayFunc(renderScene);

glutReshapeFunc(changeSize);

glutIdleFunc(renderScene);

glutKeyboardFunc(processNormalKeys);

// OpenGL init

glEnable(GL\_DEPTH\_TEST);

menuBackground = glutCreateMenu(Sky);

glutAddMenuEntry("Sky shade 1", 1);

glutAddMenuEntry("Sky shade 2", 2);

glutAddMenuEntry("Sky shade 3", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

// enter GLUT event processing cycle

glutMainLoop();

return 1;

}

