**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;
}
```