Stol de pasari

Scopul proiectului este realizarea unei animatii in OpenGL ce reprezinta un stol de pasari in zbor, utilizand translatari, rotatii si redimensionari.

1) Desenul initial

i) Fundalul

Fundalul este un gradient realizat desenand cate un punct in fiecare colt al planului. Punctele din partea superioara sunt desenate cu o nuanta mai inchisa (R:0.6; G:0.9; B:1.0) decat cele din partea inferioara a fundalului (R:0.0; G:0.6; B:1.0).

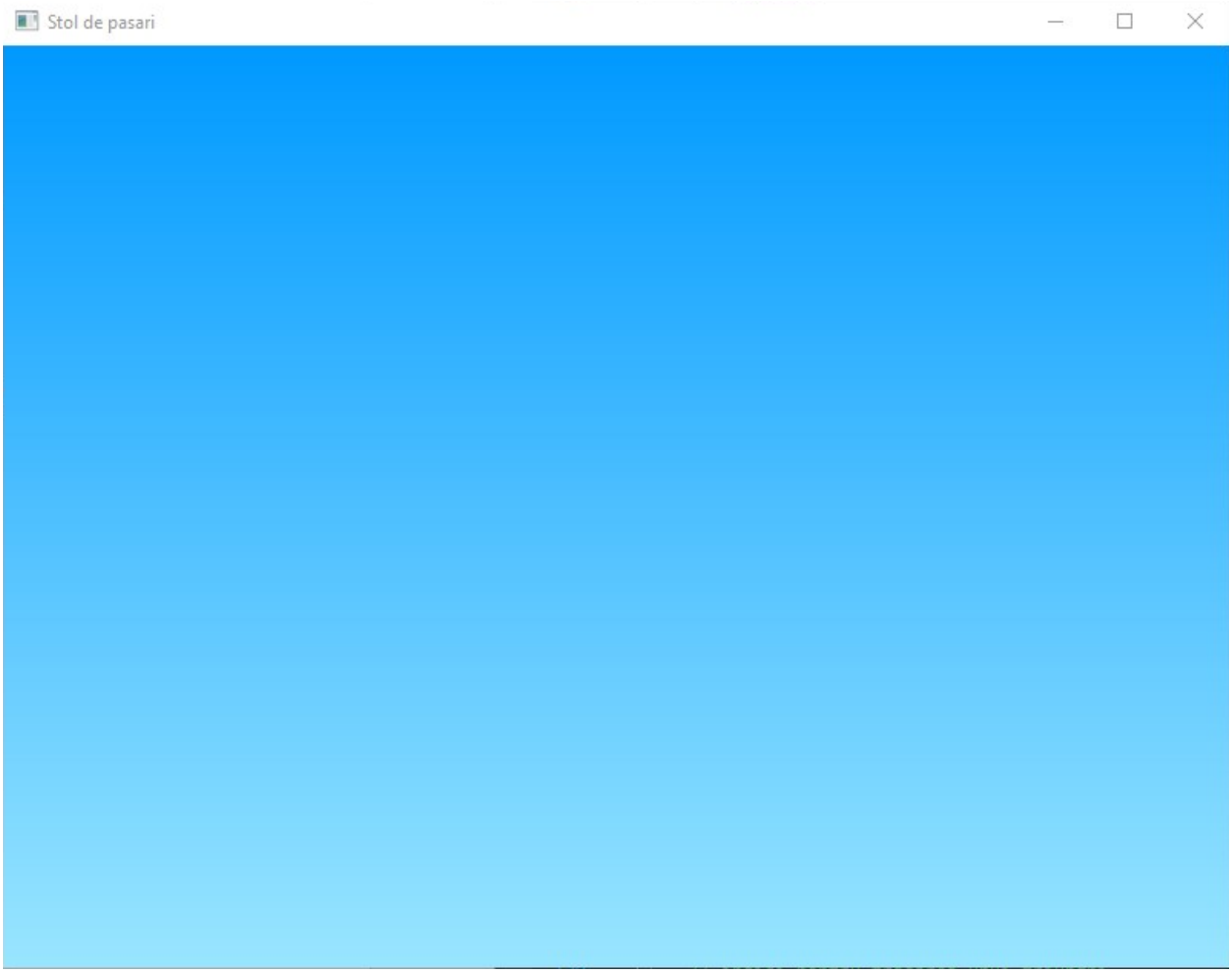

Fig 1: gradient fundal

ii) Nori

Pentru fiecare nor am folosit cate un poligon format din 7 puncte de culoare alba.

```
//nor 1 dreapta
73.0f,  66.0f, 0.0f, 1.0f, // 4 = E
59.0f,  80.0f, 0.0f, 1.0f, // 5 = F
20.0f,  59.0f, 0.0f, 1.0f, // 6 = g
40.0f,  40.0f, 0.0f, 1.0f, // 7 = h
52.0f,  57.0f, 0.0f, 1.0f, // 8 = i
35.0f,  81.0f, 0.0f, 1.0f, // 9 = j
33.0f,  68.0f, 0.0f, 1.0f, // 10 = k

//nor 2 stanga
-77.0f,  32.0f, 0.0f, 1.0f, // 11 = l
-60.0f,  60.0f, 0.0f, 1.0f, // 12 = m
-38.0f,  52.0f, 0.0f, 1.0f, // 13 = n
-44.0f,  21.0f, 0.0f, 1.0f, // 14 = o
-28.0f,  32.0f, 0.0f, 1.0f, // 15 = p
-52.0f,  27.0f, 0.0f, 1.0f, // 16 = q
-65.0f,  41.0f, 0.0f, 1.0f, // 17 = r
```

Fig 2: punctele folosite pentru realizarea norilor

```
glDrawElements(GL_TRIANGLE_FAN, 7, GL_UNSIGNED_INT, (void*)(20));
glDrawElements(GL_TRIANGLE_FAN, 7, GL_UNSIGNED_INT, (void*)(48));
```

Fig 3: desenarea norilor

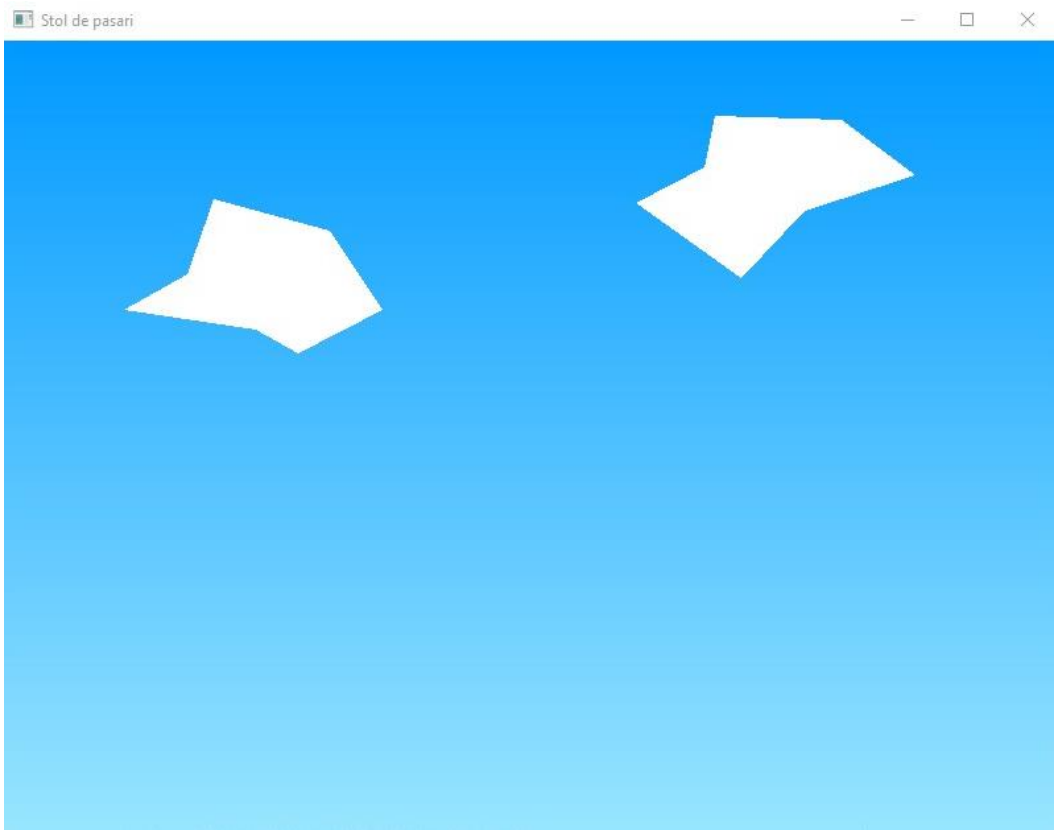

Fig 4: fundal cu nori

iii) Pasare

Pasarile sunt desenate ca un “V” intors, format din doua triunghiuri.

```
//pasare
-3.0f, -4.0f, 0.0f, 1.0f, // 18
 0.0f,  4.0f, 0.0f, 1.0f,  // 19 = varf
 3.0f, -4.0f, 0.0f, 1.0f, // 20
 0.0f,  0.5f, 0.0f, 1.0f, // 21
```

Fig 5: puncte pasare

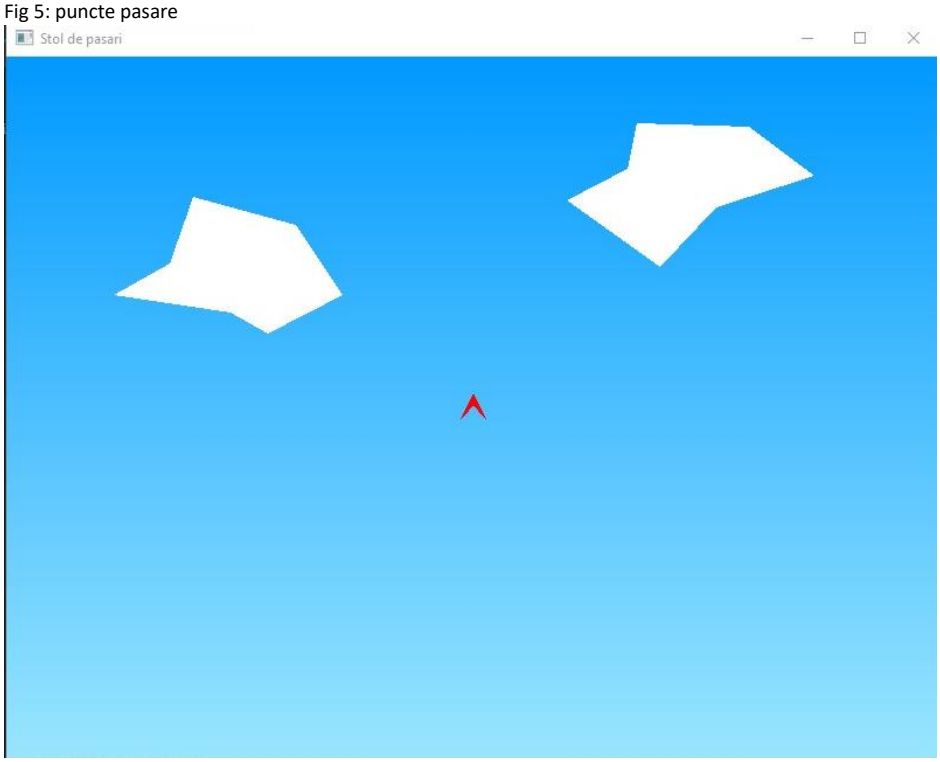

Fig 6: desenarea primei pasari

2) Stolul

i. Obiectul “Pasare”

Pentru memorarea informatiilor despre pasarile din stol am creat un obiect de tip “Pasare” cu atributele:

- Int culoare: valoarea care va fi transmisa in shader pentru culoare
- Float transX: translatia pe axa OX care va fi aplicata pasarii la randare
- Float transY: translatia pe axa OY care va fi aplicata pasarii la randare
- Float angle: masura unghiului cu care este rotita pasarea in sens trigonometric pornind de la axa OY
- Int mutareRandom: determina conform unor reguli (vezi Animatie -> Miscarile aleatoare ale pasarilor) ultima mutare random a pasarii

```
struct pasare {
    int culoare;
    float transX;
    float transY;
    float angle;
    int mutareRandom;
};
```
Fig 7: Strunctura obiectului “Pasare”

ii. Memorarea informatiilor despre stol

Pentru memorarea informatiilor am folosit un Array de obiecte de tip “Pasare” in care am incarcat valorile pentru translatii folosind o structura repetitiva. Toate pasarile au initial valoarea “2” pentru culoare (rosu) si unghiul de 0 grade (indreptate in sus, paralel cu axa OY).

```
for (int i = 0; i < 10; i++) {
    pasari[i].culoare = 2;
    pasari[i].transX = xs[i];
    pasari[i].transY = ys[i];
    pasari[i].angle = 0;
}
```
Fig 8: Incarcarea informatiilor in Array

iii. Aplicarea translatiilor si rotatiilor

Pentru fiecare pasare au fost generate o matrice de translatie si una de rotatie cu ajutorul informatiilor salvate mai devreme. Aceste matrici au fost inmultite astfel incat sa se aplice intai rotatia, apoi translatei si la final o redimensionare.

```
glm::mat4 matrRot = glm::rotate(glm::mat4(1.0f), glm::radians(pasari[i].angle), glm::vec3(0.0, 0.0, 1.0));
glm::mat4 matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(pasari[i].transX, pasari[i].transY, 0.0));
myMatrix = resizeMatrix * matrTransl * matrRot;
```
Fig 9: Matricile

iv. Desenarea

Stolul a fost desenat parcurgand Array-ul de obiecte si desenand fiecare pasare pe rand.

```
// Desenarea pasarilor
for (int i = 0; i < nrPasari; i++) {
    //culoare
    codCol = pasari[i].culoare;
    codColLocation = glGetUniformLocation(ProgramId, "codCol");
    glUniform1i(codColLocation, codCol);
    // matrici de rotatie si translatie
    glm::mat4 matrRot = glm::rotate(glm::mat4(1.0f), glm::radians(pasari[i].angle), glm::vec3(0.0, 0.0, 1.0));
    glm::mat4 matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(pasari[i].transX, pasari[i].transY, 0.0));
    myMatrix = resizeMatrix * matrTransl * matrRot;
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
    // desenarea pasarii i
    glDrawElements(GL_TRIANGLE_FAN, 4, GL_UNSIGNED_INT, (void*)(76));
}
```
Fig 10: Desenarea pasarilor

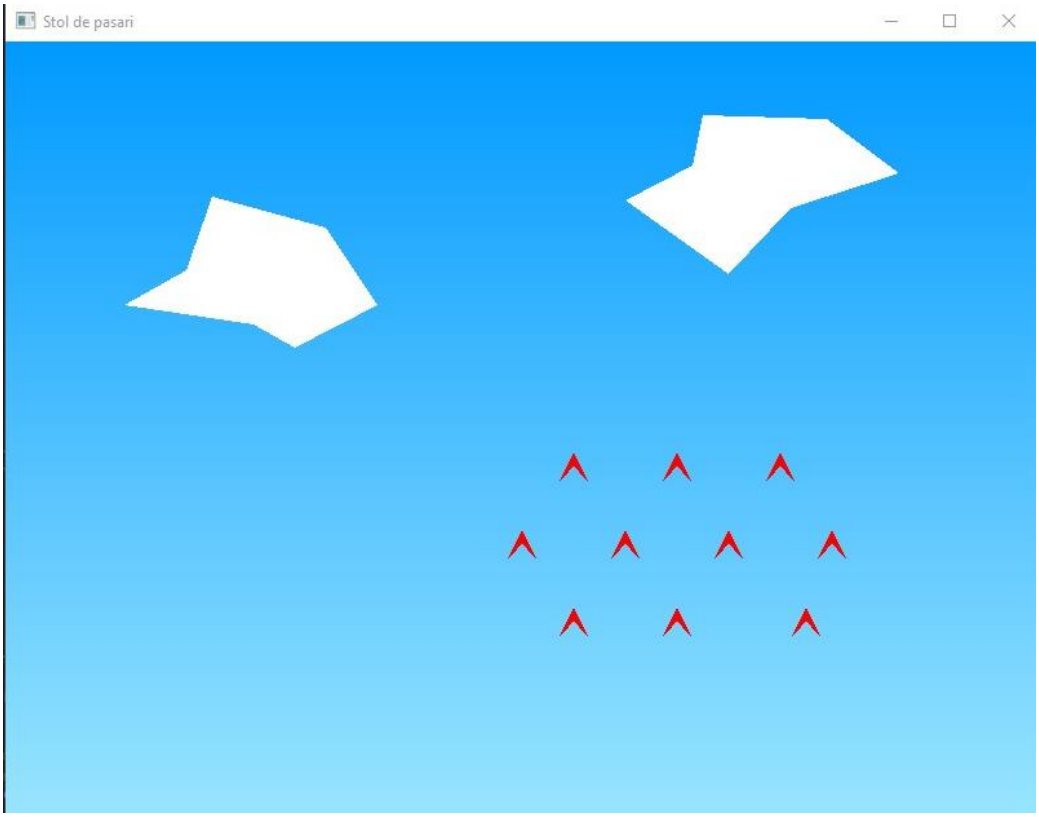

Fig 11: Imaginea obtinuta dupa desenarea pasarilor

3) Animatia

i) Miscarea stolului

Animatia de miscare a stolului este realizare in urma unei functii de mouse, conform informatiilor salvate in structuri de tip deque. Aceasta functie extrage prima valoare din fiecare deque, dupa care le adauga la finalul acestora si modifica valoarea fiecarei pasari din Array cu acea valoare. Dupa aceasta re-desenez intreaga imagine, cu noile valori pentru pasari.

```
int moveX = nextMoveX[0];
int moveY = nextMoveY[0];
int angle = nextAngle[0];
nextMoveX.pop_front();
nextMoveY.pop_front();
nextAngle.pop_front();
nextMoveX.push_back(moveX);
nextMoveY.push_back(moveY);
nextAngle.push_back(angle);
for (int j = 0; j < nrPasari; j++)
{
    pasari[j].transX += moveX;
    pasari[j].transY += moveY;
    pasari[j].angle += angle;
}
Sleep(300);
```

Fig 12: functia de mutare a stolului

ii) Schimbarea pozitiilor pasarilor

Pe parcursul functiei de mutarea a stolului generez un numar aleator intre 0 si 9. Daca acesta este mai mic decat 2 (o sansa de 20%) aleg doua pasari aleatoare si diferite, le colorez in albastru, respectiv verde si calculez diferenta dintre translatiile acestora (distanta) pe fiecare axa. Dupa aceea setez variabilei switchOn valoarea “1” care arata ca exista o schimbare in curs si variabilei switchStage valoare “0”, care arata ca inca nu a fost efectuat primul pas al algoritmului de schimbare. Aceste variabile, cat si cele in care salvez pozitiile pasarilor din Array si distanta dintre acestea pe fiecare axa, sunt declarate global pentru a pastra informatia in iteratiile urmatoare.

La urmatoarea iteratie a functiei de mouse, se intra pe ramura de schimbare care aduce fiecare pasare cu 1/5 din distanta dintre aceste mai aproape de pozitia initiala a celeilalte si se incrementeaza valoarea variabilei switchStage. Cand switchStage ajunge la valoarea “5”, pasarilor li se atribuie culoarea initiala si se modifica valoarea lui switchOn la “0”, pentru a nu se mai intra pe aceasta ramura a functiei de mouse.

La fiecare stagiu al schimbarii stolul inca se muta conform miscarilor definite mai sus.

```
int fstBird, sndBird, switchOn = 0, switchStage;
float difX, difY;
void cycle(void)
{
    int randomNr = rand() % 10; // numar random intre 0 si 9;
    if (switchOn == 1)
    {
        //le mut pe fiecare cu cate diferenta/5 in directia celeilalte
        pasari[fstBird].transX -= difX / 5;
        pasari[fstBird].transY -= difY / 5;
        pasari[sndBird].transX += difX / 5;
        pasari[sndBird].transY += difY / 5;
        switchStage++;
        if (switchStage == 5) // daca am mutat de 5 ori
        {
            switchOn = 0;
            pasari[fstBird].culoare = 2;// revin la culorile initiale
            pasari[sndBird].culoare = 2;
        }
    }else if (randomNr < 2) //20% sansa
    {
        switchOn = 1;// setez switchOn pentru a intra pe cealalta ramura
        fstBird = rand() % 10;//aleg 2 pasari random
        do { sndBird = rand() % 10; } while (sndBird == fstBird);
        pasari[fstBird].culoare = 1;//le colorez diferit
        pasari[sndBird].culoare = 3;
        float fx = pasari[fstBird].transX;//extrag coordonatele
        float fy = pasari[fstBird].transY;
        float sx = pasari[sndBird].transX;
        float sy = pasari[sndBird].transY;
        difX = fx - sx;//calculez diferenta pe ambele axe
        difY = fy - sy;
        switchStage = 0;//setez prima stare a schimbari
    }
```

Fig 13: Algoritmul de schimbare a pasarilor

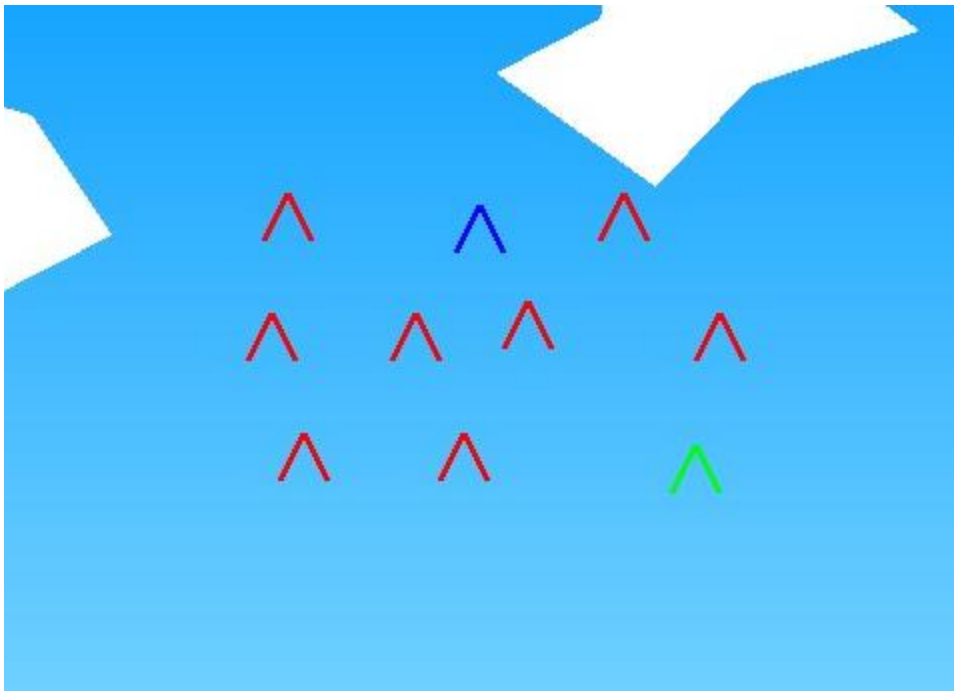

Fig 14: Miscarea, part 1
**In imagine a fost utilizata o metoda initiala (nefinala) pentru desenarea pasarilor

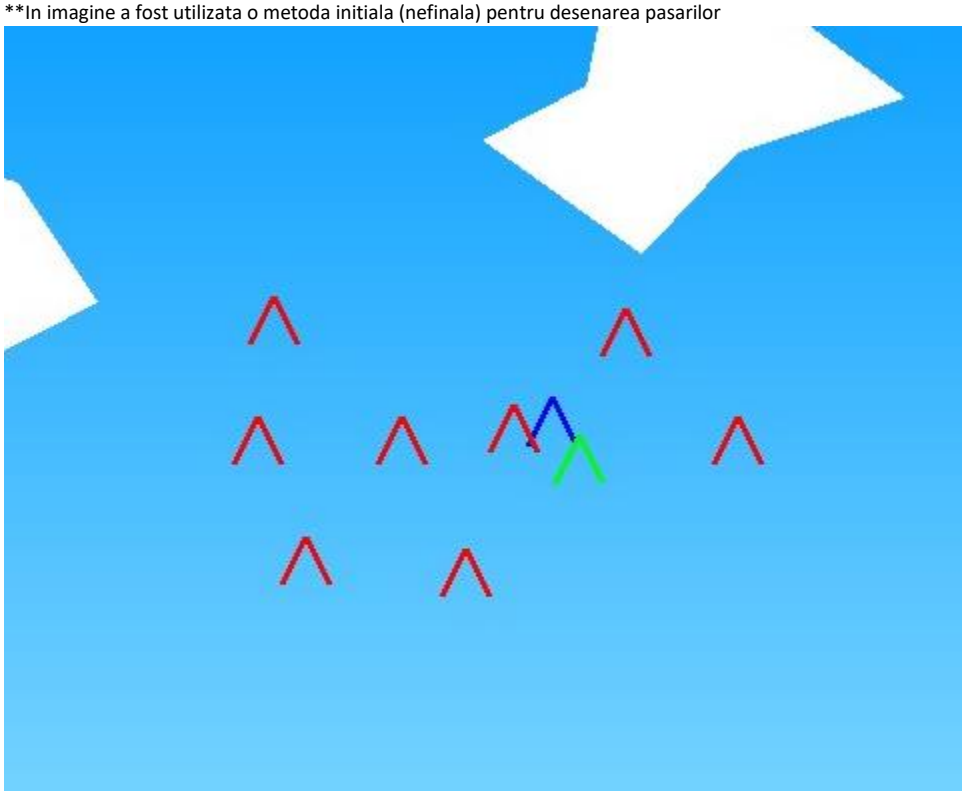

Fig 15: Miscarea, part 2
**In imagine a fost utilizata o metoda initiala (nefinala) pentru desenarea pasarilor

iii) Miscarile aleatoare ale pasarilor

La fiecare miscare a stolului, fiecare pasare are o sansa de 40% sa i se aplice inca o mutare aleatoare in una din cele 4 directii. Acestea fac animatie sa para mai putin paralela si da o nota de naturalete miscarii stolului.

O mutare este data de una din valorile atribuite aleator variabilei mutareRandom:

- -1 -> nicio mutare (valoare default)
- 0 -> mutare la dreapta (+2 pe axa OX)
- 1 -> mutare la stanga (-2 pe axa OX)
- 2 -> mutare in sus (+2 pe axa OY)
- 3 -> mutare in jos (-2 pe axa OY

De asemenea, la fiecare mutare a stolului se anuleaza ultima mutare aleatoare efectuata pasarii pentru a evita “despartirea” (indepartarea) stolului.

```
for (int j = 0; j < nrPasari; j++)
{
    int randomNr2 = rand() % 10; // numar random intre 0 si 9;
    int mutareRandom = -1; // valoare default, nicio mutare
    if (randomNr2 < 4) // 40 %
        mutareRandom = rand() % 4;
    pasari[j].transX += moveX + mutareX(mutareRandom) - mutareX(pasari[j].mutareRandom);
    pasari[j].transY += moveY + mutareY(mutareRandom) - mutareY(pasari[j].mutareRandom);
    pasari[j].angle += angle;
    pasari[j].mutareRandom = mutareRandom;
}
```

Fig 16: miscarile aleatoare, cod

```
int mutareX(int m)
{
    if (m == 0)
        return 2; // o mutare la dreapta
    else if (m == 1)
        return -2; // o mutare la stanga
    else
        return 0; // nicio mutare
}
int mutareY(int m)
{
    if (m == 2)
        return 2; // o mutare in sus
    else if (m == 3)
        return -2; // o mutare in jos
    else
        return 0; // nicio mutare
}
```

Fig 17: generarea valorilor pentru mutarile aleatoare

4) Originalitate

Originalitatea proiectului este data de metoda de salvare a informatiilor despre membri stolului de pasari in obiectele de tip “Pasare”, dar si de algoritmul de mutari aleatorii si desenul folosit pentru fundal.

5) Modificari aduse in urma prezentarii:

Am schimbat modul de desenare al pasarilor, de la doua segmente (fig. 14 & 15) la doua triunghiuri, cum se poate observa in sectiunea despre desenarea pasarilor si a stolului (fig. 6 & 11).

6) Cod Sursa:

```
#include <windows.h>  // biblioteci care urmeaza sa fie incluse
#include <stdlib.h> // necesare pentru citirea shader-elor
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <GL/glew.h> // glew apare inainte de freeglut
#include <GL/freeglut.h> // nu trebuie uitat freeglut.h

#include <windows.h>
#include <deque>

#include "loadShaders.h"

#include "glm/glm/glm.hpp"
#include "glm/glm/gtc/matrix_transform.hpp"
#include "glm/glm/gtx/transform.hpp"
#include "glm/glm/gtc/type_ptr.hpp"

using namespace std;

//////////////////////////////////////

GLuint
VaoId,
VboId,
EboId,
ColorBufferId,
ProgramId,
myMatrixLocation,
codColLocation;


glm::mat4 myMatrix, resizeMatrix, matrTransl;

std::deque<int> nextMoveX;
std::deque<int> nextMoveY;
std::deque<int> nextAngle;
int codCol, nrPasari = 10;
float PI = 3.141592, angle = 0;
float tx = 0; float ty = 0;
```

```
float width = 100.f, height = 100.f;

int nrMutari = 95;
int mutariX[1000] = {-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3, -3,  3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3,  3, -3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3, -3,  3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,  3,  0, 0, 0, 0, 0 };
int mutariY[1000] = { 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,  3,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,  0, -3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3, -3,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0,  3, 3, 3, 3, 3 };
int angles[1000] = { 25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,-25,-90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,-90,-45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,+45,+90, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,+45,+45, 0, 0, 0, 0 };

struct pasare {
	int culoare;
	float transX;
	float transY;
	float angle;
	int mutareRandom;
};

int xs[10] = { 10, 30, 50,   0, 20, 40, 60,  10,  30,  55};
int ys[10] = {-10,-10,-10, -30,-30,-30,-30, -50, -50, -50};

pasare pasari[10];

void CreateVBO(void)
{

	// coordonatele varfurilor
	static const GLfloat vf_pos[] =
	{
		//fundal
		-100.0f, -100.0f, 0.0f, 1.0f,
		 100.0f, -100.0f, 0.0f, 1.0f,
		 100.0f,  100.0f, 0.0f, 1.0f,
		-100.0f,  100.0f, 0.0f, 1.0f,

		//nor 1 dreapta
		73.0f,  66.0f, 0.0f, 1.0f, // 4 = E
		59.0f,  80.0f, 0.0f, 1.0f, // 5 = F
		20.0f,  59.0f, 0.0f, 1.0f, // 6 = g
		40.0f,  40.0f, 0.0f, 1.0f, // 7 = h
		52.0f,  57.0f, 0.0f, 1.0f, // 8 = i
```

```
        35.0f,  81.0f, 0.0f, 1.0f, // 9 = j
        33.0f,  68.0f, 0.0f, 1.0f, // 10 = k

        //nor 2 stanga
        -77.0f,  32.0f, 0.0f, 1.0f, // 11 = l
        -60.0f,  60.0f, 0.0f, 1.0f, // 12 = m
        -38.0f,  52.0f, 0.0f, 1.0f, // 13 = n
        -44.0f,  21.0f, 0.0f, 1.0f, // 14 = o
        -28.0f,  32.0f, 0.0f, 1.0f, // 15 = p
        -52.0f,  27.0f, 0.0f, 1.0f, // 16 = q
        -65.0f,  41.0f, 0.0f, 1.0f, // 17 = r

        //pasare
        -3.0f, -4.0f, 0.0f, 1.0f, // 18
         0.0f,  4.0f, 0.0f, 1.0f,  // 19 = varf
         3.0f, -4.0f, 0.0f, 1.0f, // 20
         0.0f,  0.5f, 0.0f, 1.0f, // 21


};
// culorile varfurilor
static const GLfloat vf_col[] =
{
        //fundal
        0.6f, 0.9f, 1.0f, 1.0f,
        0.6f, 0.9f, 1.0f, 1.0f,
        0.0f, 0.6f, 1.0f, 1.0f,
        0.0f, 0.6f, 1.0f, 1.0f,

        //nor 1
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,

        //nor 2
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,
```

```
        1.0f, 1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f, 1.0f,

        //pasare
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,


};
// indici pentru trasarea unui triunghi
static const GLuint vf_ind[] =
{
0, 1, 2, 3, 0, // 4
10, 9, 5, 4, 8, 7, 6, // 11
16, 14, 15, 13, 12, 17, 11, // 18
21, 18, 19, 20 // 21
};


// se creeaza un buffer nou pentru varfuri
glGenBuffers(1, &VboId);
// buffer pentru indici
glGenBuffers(1, &EboId);
// se creeaza / se leaga un VAO (Vertex Array Object)
glGenVertexArrays(1, &VaoId);


// legare VAO
glBindVertexArray(VaoId);

// buffer-ul este setat ca buffer curent
glBindBuffer(GL_ARRAY_BUFFER, VboId);

// buffer-ul va contine atat coordonatele varfurilor, cat si datele de culoare
glBufferData(GL_ARRAY_BUFFER, sizeof(vf_col) + sizeof(vf_pos), NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(vf_pos), vf_pos);
glBufferSubData(GL_ARRAY_BUFFER, sizeof(vf_pos), sizeof(vf_col), vf_col);

// buffer-ul pentru indici
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EboId);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(vf_ind), vf_ind, GL_STATIC_DRAW);
```

```
    // se activeaza lucrul cu atribute; atributul 0 = pozitie, atributul 1 = culoare, acestea sunt
indicate corect in VBO
    glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, NULL);
    glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 0, (const GLvoid*)sizeof(vf_pos));
    glEnableVertexAttribArray(0);
    glEnableVertexAttribArray(1);


}
void DestroyVBO(void)
{
    glDisableVertexAttribArray(1);
    glDisableVertexAttribArray(0);


    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glDeleteBuffers(1, &EboId);
    glDeleteBuffers(1, &ColorBufferId);
    glDeleteBuffers(1, &VboId);

    glBindVertexArray(0);
    glDeleteVertexArrays(1, &VaoId);


}

void CreateShaders(void)
{
    ProgramId = LoadShaders("05_03_Shader.vert", "05_03_Shader.frag");
    glUseProgram(ProgramId);
}

void DestroyShaders(void)
{
    glDeleteProgram(ProgramId);
}

void Initialize(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f); // culoarea de fond a ecranului

    for (int i = 0; i < 10; i++) {
            pasari[i].culoare = 2;
            pasari[i].transX = xs[i];
```

```
		pasari[i].transY = ys[i];
		pasari[i].angle = 0;
	}
	for (int j = 0; j < nrMutari; j++)
	{
		nextMoveX.push_back(mutariX[j]);
		nextMoveY.push_back(mutariY[j]);
		nextAngle.push_back(angles[j]);
	}
}
void RenderFunction(void)
{
	resizeMatrix = glm::ortho(-width, width, -height, height);

	glClear(GL_COLOR_BUFFER_BIT);

	// creare VBO
	CreateVBO();
	// Creare shader + transmitere variabile uniforme
	CreateShaders();
	myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

	myMatrix = resizeMatrix; // desenarea fundalului
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glLineWidth(3.0f);
	glPointSize(10.0f);
	glDrawElements(GL_TRIANGLE_STRIP, 5, GL_UNSIGNED_INT, (void*)(0));

	glDrawElements(GL_TRIANGLE_FAN, 7, GL_UNSIGNED_INT, (void*)(20));
	glDrawElements(GL_TRIANGLE_FAN, 7, GL_UNSIGNED_INT, (void*)(48));

	// Desenarea pasarilor
	for (int i = 0; i < nrPasari; i++) {
		//culoare
		codCol = pasari[i].culoare;
		codColLocation = glGetUniformLocation(ProgramId, "codCol");
		glUniform1i(codColLocation, codCol);
		// matrici de rotatie si translatie
		glm::mat4 matrRot = glm::rotate(glm::mat4(1.0f), glm::radians(pasari[i].angle),
glm::vec3(0.0, 0.0, 1.0));
		glm::mat4 matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(pasari[i].transX,
pasari[i].transY, 0.0));
		myMatrix = resizeMatrix * matrTransl * matrRot;
		glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
```

```
			// desenarea pasarii i
			glDrawElements(GL_TRIANGLE_FAN, 4, GL_UNSIGNED_INT, (void*)(76));
	}

	glFlush();
}
void Cleanup(void)
{
	DestroyShaders();
	DestroyVBO();
}


int mutareX(int m)
{
	if (m == 0)
			return 2; // o mutare la dreapta
	else if (m == 1)
			return -2; // o mutare la stanga
	else
			return 0; // nicio mutare
}
int mutareY(int m)
{
	if (m == 2)
			return 2; // o mutare in sus
	else if (m == 3)
			return -2; // o mutare in jos
	else
			return 0; // nicio mutare
}

int fstBird, sndBird, switchOn = 0, switchStage;
float difX, difY;
void cycle(void)
{
	int randomNr = rand() % 10; // numar random intre 0 si 9;
	if (switchOn == 1)
	{
			//le mut pe fiecare cu cate diferenta/5 in directia celeilalte
			pasari[fstBird].transX -= difX / 5;
			pasari[fstBird].transY -= difY / 5;
			pasari[sndBird].transX += difX / 5;
			pasari[sndBird].transY += difY / 5;
```

```
            switchStage++;
            if (switchStage == 5) // daca am mutat de 5 ori
            {
                    switchOn = 0;
                    pasari[fstBird].culoare = 2;// revin la culorile initiale
                    pasari[sndBird].culoare = 2;
            }
    }else if (randomNr < 2) //20% sansa
    {
            switchOn = 1;// setez switchOn pentru a intra pe cealalta ramura
            fstBird = rand() % 10;//aleg 2 pasari random
            do { sndBird = rand() % 10; } while (sndBird == fstBird);
            pasari[fstBird].culoare = 1;//le colorez diferit
            pasari[sndBird].culoare = 3;
            float fx = pasari[fstBird].transX;//extrag coordonatele
            float fy = pasari[fstBird].transY;
            float sx = pasari[sndBird].transX;
            float sy = pasari[sndBird].transY;
            difX = fx - sx;//calculez diferenta pe ambele axe
            difY = fy - sy;
            switchStage = 0;//setez prima stare a schimbari
    }

            int moveX = nextMoveX[0];
            int moveY = nextMoveY[0];
            int angle = nextAngle[0];
            nextMoveX.pop_front();
            nextMoveY.pop_front();
            nextAngle.pop_front();
            nextMoveX.push_back(moveX);
            nextMoveY.push_back(moveY);
            nextAngle.push_back(angle);
            for (int j = 0; j < nrPasari; j++)
            {
                    int randomNr2 = rand() % 10; // numar random intre 0 si 9;
                    int mutareRandom = -1; // valoare default, nicio mutare
                    if (randomNr2 < 4) // 40 %
                            mutareRandom = rand() % 4;
                    pasari[j].transX += moveX + mutareX(mutareRandom) -
mutareX(pasari[j].mutareRandom);
                    pasari[j].transY += moveY + mutareY(mutareRandom) -
mutareY(pasari[j].mutareRandom);
                    pasari[j].angle += angle;
                    pasari[j].mutareRandom = mutareRandom;
```

```
            }
            Sleep(300);



        glutPostRedisplay();
    }
    void mouse(int button, int state, int x, int y)
    {
        switch (button) {
        case GLUT_LEFT_BUTTON:
            glutIdleFunc(cycle);

            break;
        case GLUT_RIGHT_BUTTON:

            break;

        }
    }

    int main(int argc, char* argv[])
    {

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition(100, 100);
        glutInitWindowSize(800, 600);
        glutCreateWindow("Stol de pasari");
        glewInit();
        Initialize();
        glutDisplayFunc(RenderFunction);
        glutMouseFunc(mouse);
        glutCloseFunc(Cleanup);
        glutMainLoop();


    }
```

7) Shaders:

```
// Shader-ul de fragment / Fragment shader
 #version 400
in vec4 ex_Color;
uniform int codCol;
```

```
out vec4 out_Color;

void main(void)
  {
      if ( codCol==0 )
            out_Color = ex_Color;
      if ( codCol==1 )
            out_Color=vec4 (0.0, 0.0, 1.0, 0.0);
      if ( codCol==2 )
            out_Color=vec4 (1.0, 0.0, 0.0, 0.0);
      if ( codCol==3 )
            out_Color=vec4 (0.0, 1.0, 0.0, 0.0);

  }



// Shader-ul de varfuri
 #version 400

layout(location=0) in vec4 in_Position;
layout(location=1) in vec4 in_Color;

out vec4 gl_Position;
out vec4 ex_Color;
uniform mat4 myMatrix;

void main(void)
  {
    gl_Position = myMatrix*in_Position;
    ex_Color = in_Color;
   }
```