**Міністерство освіти і науки України**

**Запорізький національний технічний університет**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №4

з дисципліни «Методи та алгоритми комп'ютерної графіки» на тему:

"Моделювання фізичних явищ"

Варіант 22

Виконала:

студентка групи КНТ-415 В.С. Хохлова

Прийняв:

асистент Ж.К. Камінська

2018

1. Мета роботи

Навчитися моделювати прості фізичні явища в OpenGL.

1. Завдання

Моделювання фізичних явищ — коливання води, нічне небо, матричний дощ зі згасанням.

1. Хід роботи

// Волны

#include <GL/glut.h>

#include <cmath>

const float K = 100; // скорость угасания колебаний

const float DT = 0.03; // скорость распространения колебаний

float sqr(float x) {

return x \* x;

}

struct P {

P(): z(0), vz(0) {}

float x; float y; float z; float vz;

};

const int N = 48;

P p[N][N];

void display() {

// рисуем сетку

glClear(GL\_COLOR\_BUFFER\_BIT);

for (int x = 0; x < N; ++x) {

glBegin(GL\_LINE\_STRIP);

glColor3f(0.655, 0.655, 0.955);

for (int y = 0; y < N; ++y)

glVertex3f(p[x][y].x, p[x][y].y, p[x][y].z \* 10);

glEnd();

}

}

void timer(int = 0) {

// рандомно в одном из 4 направлений вызываем колебание (изменение z координаты точки сетки)

const int dx[] = {-1, 0, 1, 0};

const int dy[] = {0, 1, 0, -1};

if (rand() % 500 == 0)

p[rand() % (N - 2) + 1][rand() % (N - 2) + 1].vz += 150;

for (int y = 1; y < N - 1; ++y)

for (int x = 1; x < N - 1; ++x) {

P &p0 = p[x][y];

for (int i = 0; i < 4; ++i) {

P &p1 = p[x + dx[i]][y + dy[i]];

const float d = sqrt(sqr(p0.x - p1.x) + sqr(p0.y - p1.y) + sqr(p0.z - p1.z));

p0.vz += K \* (p1.z - p0.z) / d \* DT;

p0.vz \*= 0.998; // коэффициент трения

} }

for (int y = 1; y < N - 1; ++y)

for (int x = 1; x < N - 1; ++x) {

P &p0 = p[x][y];

p0.z += p0.vz \* DT;

}

display();

glutTimerFunc(1, timer, 0);

}

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

for (int x = 0; x < N; ++x)

for (int y = 0; y < N; ++y) {

p[x][y].x = x \* 480 / N;

p[x][y].y = y \* 480 / N;

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(852, 480);

glutInitWindowPosition(0, 86);

glutCreateWindow("'3d wave effect', Khokhlova Varvara, KNT-415, 2018");

glClearColor(0.008, 0.003, 0.231, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glFrustum(-100, 100, -100, 100, 100, 2000);

glMatrixMode(GL\_MODELVIEW);

glTranslatef(-240, -240, -200);

glRotatef(-30, 1, 0, 0);

glutDisplayFunc(display);

timer();

glutMainLoop();

}

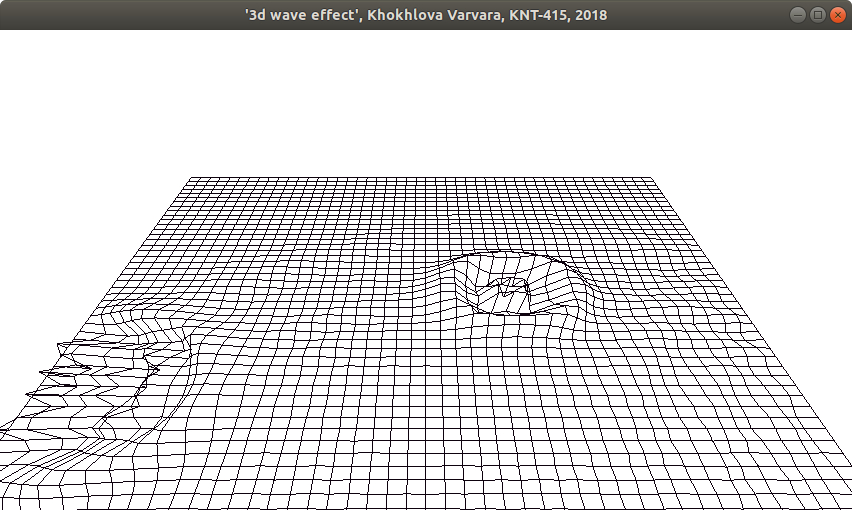


Рисунок 3.1 – Реалізація ефекту хвиль

// Matrix Code

#include <GL/glut.h>

#include <iostream>

using namespace std;

struct Ch {

// инициализируем начальными значениями

Ch(char ach = ' ', int alive = 0): ch(ach), live(alive) {}

char ch; // символ

int live; // время угасания

};

// задаем ширину и высоту экрана

const int W = 852 / 9; const int H = 480 / 15;

Ch screen[W][H];

void display() {

// выводим символ в зависимости от отставшегося времени его угасания

glClear(GL\_COLOR\_BUFFER\_BIT);

for (int y = 0; y < H; ++y)

for (int x = 0; x < W; ++x) {

glRasterPos2f(x \* 9, (y + 1) \* 15);

glColor3f(0, screen[x][y].live / 2000.0, 0);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, screen[x][y].ch);

}

glutSwapBuffers();

}

void timer(int = 0) {

// уменьшаем в таймере оставшее время жизни символа

for (int y = 0; y < H; ++y)

for (int x = 0; x < W; ++x)

if (screen[x][y].live > 0)

--screen[x][y].live;

// в рандомном месте экрана выводим новый символ

int x = rand() % W;

for (int y = 0; y < H; ++y)

if (screen[x][y].live <= 0) {

screen[x][y].ch = rand() % 256;

screen[x][y].live = 2000;

break;

}

glutPostRedisplay();

glutTimerFunc(0, timer, 0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(852, 480);

glutInitWindowPosition(0, 86);

glutCreateWindow("Matrix Code, Khokhlova Varvara, KNT-415, 2018");

glClearColor(0, 0, 0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, 852, 480, 0, -1, 1);

glutDisplayFunc(display);

timer();

glutMainLoop(); }

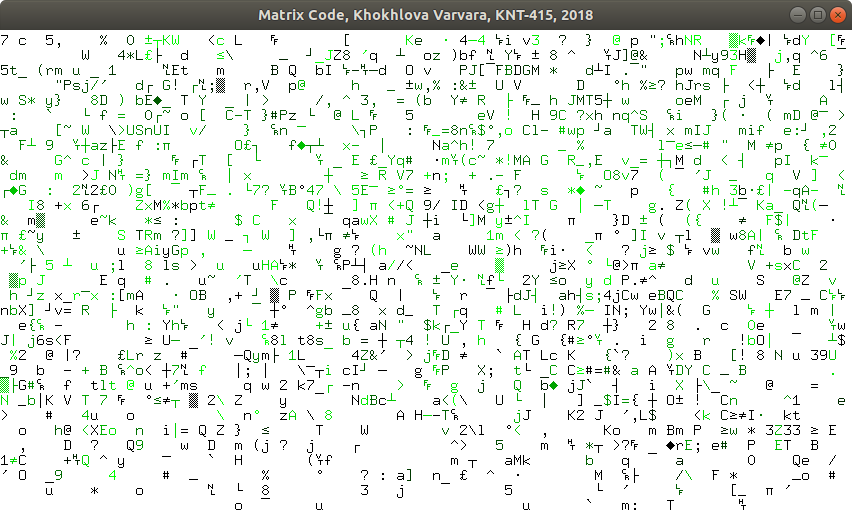


Рисунок 3.2 – Реалізація матричного коду з ефектом згасання

// Звездное небо

#include <GL/glut.h>

#include <vector>

#include <cmath>

using namespace std;

// структура с координатами звезд

struct Star {

float x, y, z;

};

vector<Star> starList;

enum { RANGE = 150 };

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

for (vector<Star>::iterator i = starList.begin(); i != starList.end(); ++i) {

// задаем яркость звезд в зависимости от дальности их нахождения

const float L = 110 / abs(i->z - (70 + 50));

// делаем звезды больше по мере их приближения

glPointSize(1.5f + 1.0f - (50 - i->z) / 550.0f);

glColor3f(L, L, L);

glBegin(GL\_POINTS);

glVertex3f(i->x, i->y, i->z);

glEnd();

}

glutSwapBuffers();

}

void timer(int = 0) {

for (vector<Star>::iterator i = starList.begin(); i != starList.end(); ++i) {

// регулируем скорость звезд

i->z += 0.25;

// отображаем звезды заново, если их уже не видно

if (i->z > 50) {

i->x = 1.0f \* (rand() % (2 \* RANGE) - RANGE);

i->y = 1.0f \* (rand() % (2 \* RANGE) - RANGE);

i->z = -500;

}

}

glutPostRedisplay();

glutTimerFunc(1, timer, 0);

}

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

// инициализируем массив 1000 звёзд

for (int i = 0; i < 1000; ++i) {

Star star = { 1.0f \* (rand() % (2 \* RANGE) - RANGE),

1.0f \* (rand() % (2 \* RANGE) - RANGE),

1.0f \* (rand() % 500 - 500) };

starList.push\_back(star);

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(852, 480);

glutInitWindowPosition(0, 86);

glutCreateWindow("Stars");

glClearColor(0, 0, 0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, 1, 0, 100);

glutDisplayFunc(display);

timer();

glutMainLoop();

}

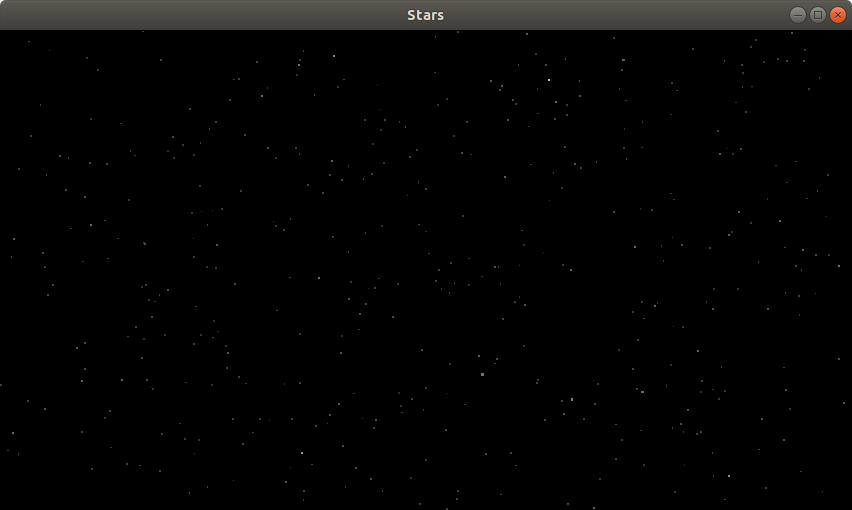


Рисунок 3.3 – Реалізація ефекту нічного неба

1. Висновки

Виконуючи лабораторну роботу, я навчилася моделювати прості фізичні явища в OpenGL такі як, туман, розщеплення, коливання води, нічне небо, матричний дощ зі згасанням.