#include <GL/glut.h>

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

float x = 0, y = 0; // координаты смещения дерева (для его анимации ветра)

float DT1 = 0.0, DT2 = 0.0; // зависимые коэффициенты скорости полёта листвы

static int l = 0; // состояние ветра

struct toch

{

int x, y;

toch(int ax, int ay) : x(ax), y(ay) {}

void sel()

{

glPointSize(6);

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_POINTS);

glVertex2f(x, y);

glEnd();

}

}; vector<toch> pix; // для отладки по точкам с помощью мыши. (В финальной версии программы не используется)

struct wood {

void createEarth() // отрисовка горы

{

glColor3ub(128, 128, 128);

glBegin(GL\_LINE\_STRIP);

glVertex2f(0, 484); // 1 Начало

// неровности

glVertex2f(33, 479);

glVertex2f(45, 479);

glVertex2f(67, 483);

glVertex2f(98, 479);

glVertex2f(102, 484);

glVertex2f(133, 479);

glVertex2f(159, 482);

glVertex2f(170, 477);

glVertex2f(193, 480);

glVertex2f(212, 485);

glVertex2f(236, 478);

glVertex2f(262, 478);

glVertex2f(277, 483);

glVertex2f(314, 485);

glVertex2f(326, 487);

glVertex2f(328, 489);

glVertex2f(390, 486);

glVertex2f(439, 494); // 2 край горы

// неровности

glVertex2f(444, 506);

glVertex2f(448, 516);

glVertex2f(451, 533);

glVertex2f(459, 540);

glVertex2f(469, 574);

glVertex2f(475, 577);

glVertex2f(484, 590);

glVertex2f(497, 600); // 3 нижная видимая часть горы

glEnd();

}

void createWood(int x, int y) // отрисовка дерева и его анимация

{

glColor3ub(150, 150, 150);

glBegin(GL\_LINE\_STRIP); // левая сторона ствола

glVertex2f(186 + x, 302 - y);

glVertex2f(182 + x, 317 - y);

glVertex2f(186 + x, 327 - y);

glVertex2f(180 + x, 339 - y);

glVertex2f(179 + (x-0.5), 351 - (y+0.5)); // -+0.5 для более плавной анимации качания дерева

glVertex2f(181 + (x - 1.0), 360 - (y+1.0)); // -+0.5 для более плавной анимации качания дерева

glVertex2f(182+(x-1.9), 370-(y+1.9)); // -+1.9 для более плавной анимации качания дерева

glVertex2f(180, 387);

glVertex2f(178, 395);

glVertex2f(175, 412);

glVertex2f(178, 429);

glVertex2f(178, 436);

glVertex2f(176, 445);

glVertex2f(176, 458);

glVertex2f(175, 473);

glVertex2f(145, 480);

glEnd();

glBegin(GL\_LINE\_STRIP); // правая сторона ствола

glVertex2f(246 + x, 299 - y);

glVertex2f(248 + x, 312 - y);

glVertex2f(250 + x, 316 - y);

glVertex2f(252 + x, 326 - y);

glVertex2f(254 + x, 333 - y);

glVertex2f(254 + x, 338 - y);

glVertex2f(253 + x, 344 - y);

glVertex2f(253 + x, 351 - y);

glVertex2f(252 + x, 356 - y);

glVertex2f(252 + (x-0.5), 365- (y+0.5)); // -+0.5 для более плавной анимации качания дерева

glVertex2f(253+ (x-1), 376-(y+1)); // -+1 для более плавной анимации качания дерева

glVertex2f(255, 392);

glVertex2f(255, 396);

glVertex2f(254, 403);

glVertex2f(252, 426);

glVertex2f(252, 433);

glVertex2f(254, 441);

glVertex2f(256, 449);

glVertex2f(256, 460);

glVertex2f(256, 464);

glVertex2f(256, 471);

glVertex2f(287, 483);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(194, 427);

glVertex2f(194, 435);

glVertex2f(196, 451);

glVertex2f(198, 454);

glVertex2f(202, 459);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(220, 437);

glVertex2f(223, 457);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(234, 432);

glVertex2f(239, 443);

glVertex2f(237, 450);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(201, 362);

glVertex2f(198, 385);

glVertex2f(199, 400);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(216, 394);

glVertex2f(214, 408);

glVertex2f(220, 418);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(228 + x, 322 - y);

glVertex2f(224 + x, 342 - y);

glVertex2f(223 + x, 343 - y);

glVertex2f(221 + x, 354 - y);

glVertex2f(225 + x, 363 - y);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(203 + x, 323 - y);

glVertex2f(203 + x, 327 - y);

glVertex2f(201 + x, 345 - y);

glEnd();

glBegin(GL\_LINE\_STRIP); // узор ствола

glVertex2f(234, 374);

glVertex2f(238, 382);

glVertex2f(235, 397);

glEnd();

}

void createCrown(int x, int y) // отрисовка кроны и взаимодействие с её анимацией

{

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_LOOP); // форма кроны

glVertex2f(186 + x \* 1.5, 300 - y);

glVertex2f(169 + x \* 1.5, 297 - y);

glVertex2f(156 + x \* 1.5, 296 - y);

glVertex2f(153 + x \* 1.5, 302 - y);

glVertex2f(128 + x \* 1.5, 307 - y);

glVertex2f(118 + x \* 1.5, 305 - y);

glVertex2f(108 + x \* 1.5, 296 - y);

glVertex2f(101 + x \* 1.5, 279 - y);

glVertex2f(107 + x \* 1.5, 260 - y);

glVertex2f(110 + x \* 1.5, 240 - y);

glVertex2f(104 + x \* 1.5, 217 - y);

glVertex2f(100 + x \* 1.5, 206 - y);

glVertex2f(97 + x \* 1.5, 195 - y);

glVertex2f(104 + x \* 1.5, 173 - y);

glVertex2f(115 + x \* 1.5, 152 - y);

glVertex2f(120 + x \* 1.5, 127 - y);

glVertex2f(123 + x \* 1.5, 122 - y);

glVertex2f(127 + x \* 1.5, 116 - y);

glVertex2f(149 + x \* 1.5, 104 - y);

glVertex2f(162 + x \* 1.5, 104 - y);

glVertex2f(182 + x \* 1.5, 103 - y);

glVertex2f(196 + x \* 1.5, 97 - y);

glVertex2f(209 + x \* 1.5, 91 - y);

glVertex2f(224 + x \* 1.5, 91 - y);

glVertex2f(240 + x \* 1.5, 97 - y);

glVertex2f(250 + x \* 1.5, 98 - y);

glVertex2f(270 + x \* 1.5, 94 - y);

glVertex2f(281 + x \* 1.5, 92 - y);

glVertex2f(297 + x \* 1.5, 93 - y);

glVertex2f(316 + x \* 1.5, 100 - y);

glVertex2f(322 + x \* 1.5, 105 - y);

glVertex2f(337 + x \* 1.5, 125 - y);

glVertex2f(347 + x \* 1.5, 146 - y);

glVertex2f(347 + x \* 1.5, 198 - y);

glVertex2f(348 + x \* 1.5, 233 - y);

glVertex2f(342 + x \* 1.5, 257 - y);

glVertex2f(336 + x \* 1.5, 275 - y);

glVertex2f(330 + x \* 1.5, 295 - y);

glVertex2f(306 + x \* 1.5, 301 - y);

glVertex2f(302 + x \* 1.5, 310 - y);

glVertex2f(290 + x \* 1.5, 313 - y);

glVertex2f(277 + x \* 1.5, 314 - y);

glVertex2f(269 + x \* 1.5, 306 - y);

glVertex2f(254 + x \* 1.5, 297 - y);

glVertex2f(246 + x \* 1.5, 297 - y);

glEnd();

glBegin(GL\_LINE\_LOOP); // Узоры листвы

glVertex2f(151, 122 - y / 1.1);

glVertex2f(151, 137 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(196, 171 - y / 1.1);

glVertex2f(200, 183 - y / 1.1);

glVertex2f(203, 196 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(132, 166 - y / 1.1);

glVertex2f(132, 183 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(129, 222 - y / 1.1);

glVertex2f(127, 237 - y / 1.1);

glVertex2f(126, 264 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(162, 164 - y / 1.1);

glVertex2f(159, 182 - y / 1.1);

glVertex2f(159, 195 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(169, 229 - y / 1.1);

glVertex2f(172, 244 - y / 1.1);

glVertex2f(173, 243 - y / 1.1);

glVertex2f(176, 261 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(192, 115 - y / 1.1);

glVertex2f(195, 129 - y / 1.1);

glVertex2f(197, 148 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(239, 110 - y / 1.1);

glVertex2f(244, 129 - y / 1.1);

glVertex2f(246, 143 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(299, 109 - y / 1.1);

glVertex2f(294, 123 - y / 1.1);

glVertex2f(291, 139 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(205, 228 - y / 1.1);

glVertex2f(211, 246 - y / 1.1);

glVertex2f(209, 265 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(266, 228 - y / 1.1);

glVertex2f(261, 247 - y / 1.1);

glVertex2f(262, 274 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(306, 233 - y / 1.1);

glVertex2f(315, 251 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(314, 165 - y / 1.1);

glVertex2f(308, 181 - y / 1.1);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2f(266, 160 - y / 1.1);

glVertex2f(257, 177 - y / 1.1);

glVertex2f(249, 191 - y / 1.1);

glEnd();

}

}; wood w; // дерево

struct Lay {

bool c = 0; // состояние цвета. (невидима до определённого момента)

float \_x, x1, x2, \_y, y1, y2; // координаты составных частей листа

float vx, vy; // скорость по координатам, для эмуляции полёта листа

float m; // масса для эмуляции полёта листвы

Lay(int x, int y, int v1, int v2, int m) { // констуктор создания листа

\_x = x; x1 = \_x; x2 = \_x + 11;

\_y = y; y1 = \_y + 11; y2 = \_y - 8;

vx = v1; vy = v2;

this->m = m;

}

void NewStart(int x, int y) { // новая генерация листвы

\_x = x; x1 = \_x; x2 = \_x + 11;

\_y = y; y1 = \_y + 11; y2 = \_y - 8;

}

}; vector<Lay> Lays; // листва

void Draw()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

for (auto i : pix) { // прорисовки точек с помощью мышки (В финальной версии программы не используется)

i.sel();

}

w.createEarth();

w.createWood(x, y);

w.createCrown(x, y);

for (vector<Lay>::iterator i = Lays.begin(); i != Lays.end(); ++i)

{

if (i->c == 0)

glColor3f(0, 0, 0);

else if (i->c == 1)

glColor3f(1, 1, 1);

glBegin(GL\_LINE\_LOOP);

glVertex2f(i->\_x, i->\_y);

glVertex2f(i->x1, i->y1);

glVertex2f(i->x2, i->y2);

glEnd();

}

glutSwapBuffers();

}

void timer(int = 0) // анимация полёта листвы или же "ветра"

{

float d = 0, f = 0;

for (vector<Lay>::iterator i = Lays.begin(); i != Lays.end(); ++i) // высчитывание скорости для каждого листа...

for (vector<Lay>::iterator j = Lays.begin(); j != Lays.end(); ++j)

if (i != j)

{

d = sqrt((i->\_x - j->\_x) \* (i->\_x - j->\_x) + (i->\_y - j->\_y) \* (i->\_y - j->\_y));

if (d < i->m + j->m)

{

f = 50 \* (i->m + j->m - d);

i->vx += f \* (i->\_x - j->\_x) / d / i->m \* DT1;

i->vy += f \* (i->\_y - j->\_y) / d / i->m \* DT2;

j->vx -= f \* (i->\_x - j->\_x) / d / j->m \* DT1;

j->vy -= f \* (i->\_y - j->\_y) / d / j->m \* DT2;

}

}

for (vector<Lay>::iterator i = Lays.begin(); i != Lays.end(); ++i) // смещаем всю конструкцию листа

{

i->\_x -= i->vx \* DT1;

i->\_y -= i->vy \* DT2;

i->x1 -= i->vx \* DT1;

i->y1 -= i->vy \* DT2;

i->x2 -= i->vx \* DT1;

i->y2 -= i->vy \* DT2;

if (i->\_x > 1000) { // если лист вышел за пределы экрана по х (за тысячных пиксель), то заново регенерируем его в кроне

i->c = 1; // делаем листву видимой (один раз, потом это не имеет смысла)

i->NewStart(rand() % 210 + 120, rand() % 171 + 124);

}

}

Draw();

glutTimerFunc(10, timer, 0);

}

void timer2(int v) // анимация дерева

{

switch (l) // режимы ветра

{

case 1:

if (x < 2) {

x += 1; y -= 1;

}

else {

x -= 1; y += 1;

}

Draw();

glutTimerFunc(300, timer2, 0);

break;

case 2:

if (x < 3.2) {

x += 1.6; y -= 1.6;

}

else {

x -= 1.6; y += 1.6;

}

Draw();

glutTimerFunc(300, timer2, 0);

break;

case 3:

if (x < 4.7) {

x += 2.35; y -= 2.35;

}

else {

x -= 2.35; y += 2.35;

}

Draw();

glutTimerFunc(300, timer2, 0);

break;

}

}

void ChangeSpeed(int v) // управление скоростью ветра

{

switch (v) {

case 0:

{

if (l == 1) { // понижаем ветер до состояния спокойствия

Lays.clear();

DT1 -= 0.002;

DT2 -= 0.002;

--l;

}

else if (l == 2) { // понижаем ветер до лёгкого

Lays.erase(end(Lays) - 21, end(Lays) - 1);

DT1 -= 0.002;

DT2 -= 0.002;

--l;

}

else if (l == 3) { // понижаем ветер до среднего

Lays.erase(end(Lays) - 41, end(Lays) - 1);

DT1 -= 0.002;

DT2 -= 0.002;

--l;

}

break;

}

case 1:

{

if (l == 0) {

for (int i = 0; i < 10; ++i) // добавляем 10 листов для лёгкого ветра

{

Lay l(rand() % 480 + 374, rand() % 343 + 120, rand() % 100000 / 500.0 - 100, rand() % 100000 / 500.0 - 100, 4);

Lays.push\_back(l);

}

DT1 = 0.026;

DT2 = 0.014;

++l;

glutTimerFunc(200, timer2, 0);

}

else if (l == 1) {

for (int i = 0; i < 20; ++i) // добавляем 20 листов для среднего ветра

{

Lay l(rand() % 480 + 374, rand() % 343 + 120, rand() % 100000 / 500.0 - 100, rand() % 100000 / 500.0 - 100, 4);

Lays.push\_back(l);

}

DT1 += 0.002;

DT2 += 0.002;

++l;

}

else if (l == 2) {

for (int i = 0; i < 40; ++i) // добавляем 40 листов для сильного ветра

{

Lay l(rand() % 480 + 374, rand() % 343 + 120, rand() % 100000 / 500.0 - 100, rand() % 100000 / 500.0 - 100, 4);

Lays.push\_back(l);

}

DT1 += 0.002;

DT2 += 0.002;

++l;

}

break;

}

case 2:

{

system("cls");

}

}

cout << "Текущий уровень ветра: " << l << " из 3" << endl <<"=================================" << endl;

}

void MenuChangeSpeed() // меню с кнопками управления скорости ветра

{

int M = glutCreateMenu(ChangeSpeed);

glutSetMenu(M);

glutAddMenuEntry("Increase wind speed(+)", 1);

glutAddMenuEntry("Reduce wind speed(-)", 0);

glutAddMenuEntry("Clear console", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void Initialize() // инициализация

{

glClearColor(0.0, 0.0, 0.0, 1.0); // фон первая цифра кол-во красного, затем зелёный, потом. Ласт оставляем на 1

glMatrixMode(GL\_PROJECTION); // матрица проекции. (как-то связана с Орто)

glLoadIdentity(); //сбрасываем все параметры

glOrtho(0.0, 1000.0, 600, 0.0, -200.0, 200.0); // создаём сетку, на которой можно рисовать точки.

}

void Reshape(int w, int h) // убираем растяжение экрана

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, w, h, 0, -200, 200); // эта штука должна быть такой же как в инициализации

}

void PressedMouse(int button, int state, int ax, int ay) // реагирует на нажатие мыши (для упрощения работы при рисовании)

{

// ax - ay - координаты, где находится мышь

// state имеет два состояниея GLUT\_UP - кнопка отпущена, и GLUT\_DOWN - кнопка нажата

switch (button)

{

case GLUT\_LEFT\_BUTTON:

{

if (state == GLUT\_UP) {

toch z(ax, ay);

pix.push\_back(z);

cout << "glVertex2f(" << ax << ", " << ay << ");" << endl;

break;

}

}

}

}

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

{

srand(time(NULL));

setlocale(LC\_ALL, "Ru");

// Инициализация

glutInit(&argc, argv); // начало работы

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); // Режим рисования и вычисления

glutInitWindowSize(1000, 600); // размер окна

glutInitWindowPosition(100, 200); // позиция окна

glutCreateWindow("Заголовок окна"); // заголовок окна

glutDisplayFunc(Draw); // передаём команды для рисования

Initialize(); // создаём область для рисования

glutReshapeFunc(Reshape); // убираем растяжение экрана

glutTimerFunc(500, timer, 0); // отвечает за анимацию листвы

MenuChangeSpeed(); // меню с кнопками управления скорости ветра

// glutMouseFunc(PressedMouse); // реагирует на нажатие мыши (для упрощения работы при рисовании.В финальной версии программы не используется)

glutMainLoop();

}