Лабораторная работа № 3

по дисциплине комп. геометрия и графика

студента группы ИТ-32

Манукова Давида Альбертовича

Выполнение: \_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_

**ОСНОВЫ КОМПЬЮТЕРНОЙ АНИМАЦИИ**

Цель работы: Изучение основных методов и получение практических навыков создания компьютерной анимации.

Содержание работы

1. Изучить возможности создания компьютерной анимации с помощью GDI+.
2. Разработать в Visual C++ оконное приложение, которое создает анимированное изображение согласно варианту задания.

*При создании изображения обязательно использовать все рассмотренные виды компьютерной анимации. Кроме того, следует применить не менее одного сложного аффинного преобразования*

1. Протестировать работу приложения, разработанного в п. 2. Результаты тестирования отразить в отчете.
2. Включить в отчет исходный программный код и выводы о проделанной работе.



Ход работы

1. Разработал в Visual C++ оконное приложение, которое создает анимированное изображение согласно варианту задания.

#include "framework.h"

#include "CGG-Lab\_3.h"

#include <windows.h>

#include <objidl.h>

#include <gdiplus.h>

#include "Resource.h"

using namespace Gdiplus;

#pragma comment (lib,"Gdiplus.lib")

#define MAX\_LOADSTRING 100

// Глобальные переменные:

GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

int frame\_step = 0;

float dist = 0.f;

float angle = 0.f;

float t = 0.f;

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

Image\* image;

UINT frameIndex;

UINT frameCount;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void Display(HDC);

void Tween(const PointF\*, const PointF\*, PointF\*, int, float);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_CGGLAB3, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_CGGLAB3));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

GdiplusShutdown(gdiplusToken);

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_CGGLAB3));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_CGGLAB3);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

{

image = Image::FromFile(L"bckgrnd.gif");

if (image == NULL) return -1;

frameIndex = 0;

frameCount = image->GetFrameCount(&FrameDimensionTime);

SetTimer(hWnd, 1, 40, NULL);

return 0;

}

case WM\_TIMER:

{

frameIndex = (frameIndex + 1) % frameCount;

image->SelectActiveFrame(&FrameDimensionTime, frameIndex);

frame\_step++;

t += 0.01f;

InvalidateRect(hWnd, NULL, FALSE);

break;

}

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_ERASEBKGND: return 1;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

Display(hdc);

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

if (image != NULL)

{

delete image;

}

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

void DrawBike(Graphics& g)

{

Rect rect;

g.GetVisibleClipBounds(&rect);

Bitmap backbuffer(rect.Width, rect.Height, &g);

Graphics temp(&backbuffer);

temp.SetSmoothingMode(SmoothingModeHighQuality);

Pen compPen(Color::Black, 10.f);

float comp[6] = {

0.0f, 0.2f,

0.3f, 0.7f,

0.8f, 1.0f };

compPen.SetCompoundArray(comp, 6);

g.DrawEllipse(&compPen, 400, 400, 120, 120); //заднее колесо

g.DrawEllipse(&compPen, 620, 400, 120, 120); //переднее колесо

HatchBrush hatchBrush(HatchStyleOutlinedDiamond, Color::Gray, Color::White);

g.FillEllipse(&hatchBrush, 400, 400, 120, 120); //заднее колесо

g.FillEllipse(&hatchBrush, 620, 400, 120, 120); //переднее колесо

//g.FillEllipse(&hatchBrush, 447, 447, 26, 26); //задняя ось

//g.FillEllipse(&hatchBrush, 671, 451, 18, 18); //передняя ось

Pen redPen(Color::Red, 8.f);

redPen.SetStartCap(LineCapRound);

redPen.SetLineJoin(LineJoinRound);

redPen.SetEndCap(LineCapRound);

// Рама

PointF framePoints[4] = {

PointF(460.f, 460.f),

PointF(550.f, 460.f),

PointF(650.f, 360.f),

PointF(490.f, 360.f)

};

g.DrawPolygon(&redPen, framePoints, 4);

// Руль

Point wheelPoints[3] = {

Point(680, 460),

Point(637, 320),

Point(610, 320)

};

g.DrawLines(&redPen, wheelPoints, 3);

// Сиденье

Point Wpt1(550, 460);

Point Wpt2(475, 340);

g.DrawLine(&redPen, Wpt1, Wpt2);

Pen blackPen(Color::Black, 8.f);

blackPen.SetStartCap(LineCapRound);

blackPen.SetEndCap(LineCapRound);

Point Qpt1(460, 340);

Point Qpt2(490, 340);

g.DrawLine(&blackPen, Qpt1, Qpt2);

}

void DrawStar(Graphics& g)

{

Rect rect;

g.GetVisibleClipBounds(&rect);

Bitmap backbuffer(rect.Width, rect.Height, &g);

Graphics temp(&backbuffer);

temp.SetSmoothingMode(SmoothingModeHighQuality);

Pen starPen(Color::Snow, 2.f);

SolidBrush starBrush(Color::Snow);

float \_X = -800.f; //смещение звезды по X

float \_Y = 200.f; //смещение звезды по Y

//PointF start\_star\_Points[] = {

// PointF(900.f, 20.f),

// PointF(905.f, 25.f),

// PointF(920.f, 25.f),

// PointF(910.f, 30.f),

// PointF(915.f, 35.f),

// PointF(900.f, 33.f),

// PointF(885.f, 35.f),

// PointF(890.f, 30.f),

// PointF(880.f, 25.f),

// PointF(895.f, 25.f),

//};

PointF start\_star\_Points[] = {

PointF(900.f, 20.f),

PointF(890.f, 35.f),

PointF(910.f, 25.f),

PointF(890.f, 25.f),

PointF(910.f, 35.f),

};

PointF end\_star\_Points[] = {

PointF(900.f + \_X, 20.f + \_Y),

PointF(880.f + \_X, 45.f + \_Y),

PointF(920.f + \_X, 30.f + \_Y),

PointF(880.f + \_X, 30.f + \_Y),

PointF(920.f + \_X, 45.f + \_Y),

};

//PointF end\_star\_Points[] = {

// PointF(900.f + \_X, 20.f + \_Y),

// PointF(915.f + \_X, 35.f + \_Y),

// PointF(930.f + \_X, 35.f + \_Y),

// PointF(920.f + \_X, 40.f + \_Y),

// PointF(930.f + \_X, 50.f + \_Y),

// PointF(900.f + \_X, 43.f + \_Y),

// PointF(870.f + \_X, 50.f + \_Y),

// PointF(880.f + \_X, 40.f + \_Y),

// PointF(870.f + \_X, 35.f + \_Y),

// PointF(885.f + \_X, 35.f + \_Y),

//};

PointF int\_star\_Points[5];

Tween(start\_star\_Points, end\_star\_Points, int\_star\_Points, 5, t);

g.FillPolygon(&starBrush, int\_star\_Points, 5, FillModeWinding);

}

void Display(HDC hdc)

{

Graphics g(hdc);

Rect rect;

g.GetVisibleClipBounds(&rect);

Bitmap backBuffer(rect.Width, rect.Height, &g);

Graphics temp(&backBuffer);

temp.DrawImage(image, 0, 0, rect.Width, rect.Height);

angle = -90.f \* frame\_step / frameCount;

DrawStar(temp);

if (t < 1)

{

DrawStar(temp);

}

if (t >= 1)

{

t = 0;

}

if (frame\_step < 50)

{

dist = 5.f \* (frame\_step - 50);

temp.TranslateTransform(0.f + dist, 0.f);

DrawBike(temp);

}

else if (frame\_step < 150)

{

temp.RotateTransform(angle + 10);

DrawBike(temp);

}

else if (frame\_step == 150)

{

frame\_step = -1;

temp.ResetTransform();

}

g.DrawImage(&backBuffer, rect);

}

PointF Tween(const PointF& A, const PointF& B, float t)

{

return PointF(A.X \* (1.f - t) + B.X \* t, A.Y \* (1.f - t) + B.Y \* t);

}

void Tween(const PointF\* A, const PointF\* B, PointF\* P, int count, float t)

{

for (int i = 0; i < count; i++)

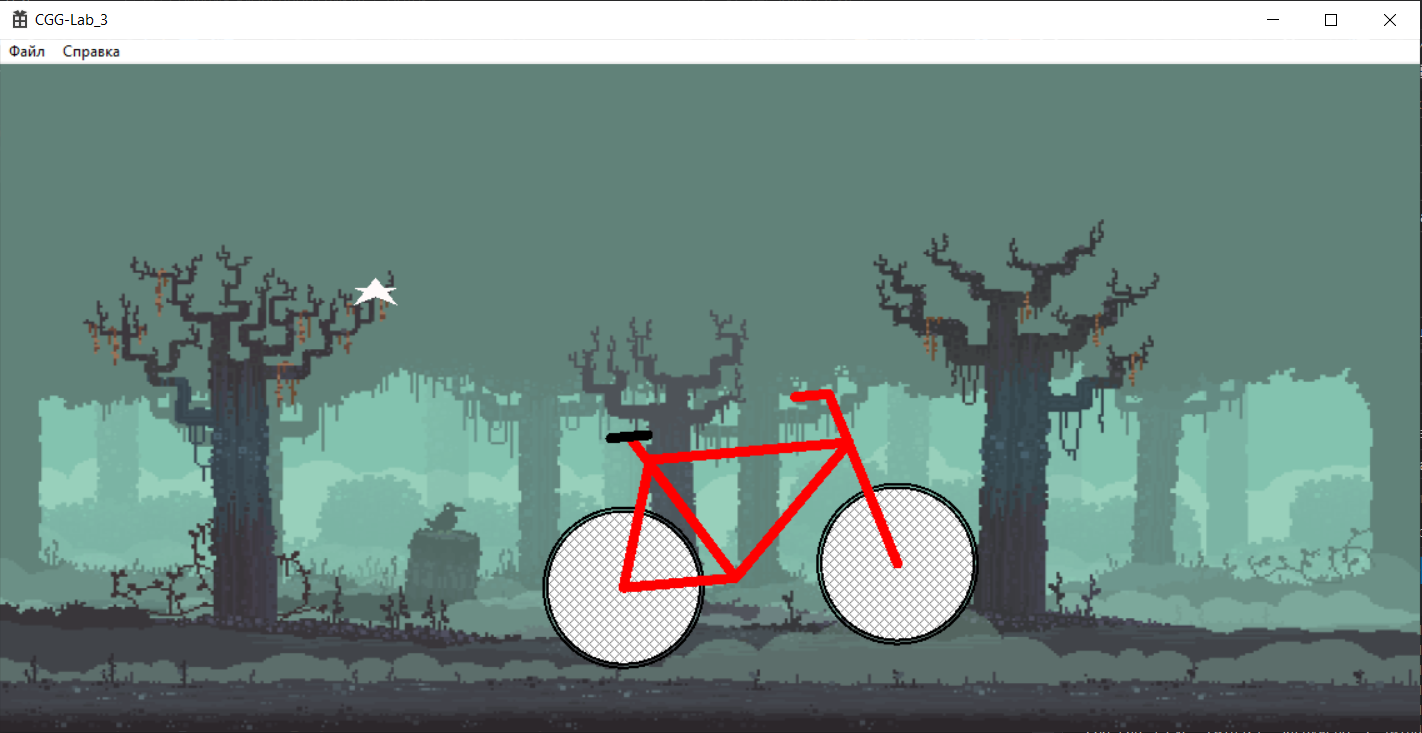
{

P[i] = Tween(A[i], B[i], t);

}

}

1. Результат работы



Вывод: изучил основные методы и получил практические навыки создания компьютерной анимации.