МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

РЕАЛИЗАЦИЯ ВЫВОДА КРИВОЙ БЕЗЬЕ

Отчет по лабораторной работе № 4, 5 дисциплины
«Компьютерная графика»

Выполнил студент группы ИВТ-21 _____/Рзаев А.Э./
Проверил старший преподаватель _____/Вожегов Д.В./

2016 г.

1 Постановка задачи

Написать программу, реализующую геометрический алгоритм построения кривой Безъе. Кривая должна строиться пошагово (с задержкой), отображая вспомогательные многоугольники, используемые для получения каждой точки. Реализовать демонстрацию анимации движения объектов по сложным траекториям, составленным из кривых Безье.

2 Общие сведения о кривой Безье

Кривые (сплайны) и поверхности Безье были использованы в 60-х годах компанией "Рено" для компьютерного проектирования формы кузовов автомобилей. В настоящее время они широко используются в компьютерной графике. Кривые Безье описываются в параметрической форме:

$x=P_x(t)$,

$y=P_y(t)$.

Значение t выступает как параметр, которому отвечают координаты отдельной точки линии. Параметрическая форма описания может быть более удобной для некоторых кривых, чем задание в виде функции y=f(x). Это потому, что функция f(x) может быть намного сложнее, чем $P_x(t)$ и $P_y(t)$, кроме того, f(x) может быть неоднозначной.

Многочлены Безье для Px(t) и Py(t) имеют такой вид:

$$Px(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} x_i$$

$$Py(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i$$

где C_m^i - сочетание m по i (известное также по биному Ньютона),

$$C_m^i = \frac{m!}{i! * (m-i)!}$$

а x_i и y_i координаты точек-ориентиров P_i.

Значение m можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров.

3 Разработка алгоритма

1. Каждая сторона контура многоугольника, проходящего по точкам-ориентирам, делится пропорционально значению t.
2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предыдущего контура.

3. Стороны нового контура снова делятся пропорционально значению t.

Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье.

Схема приведённого выше алгоритма приведена в приложении А.

Листинг программы, осуществляющей данный алгоритм, приведён в приложении Б.

Экранные формы программы приведены в приложении В.

4 Вывод

В ходе данной лабораторной работы были получены знания о построении кривой Безье, изучены различные алгоритмы и их интерпретации. Это позволило написать программу, реализующую этот алгоритм, а также создать анимацию движения объекта по сложным траекториям, а именно двоичный трёхразрядный счетчик на кривых Безье.

Приложение А
(обязательное)
Блок-схемы алгоритмов

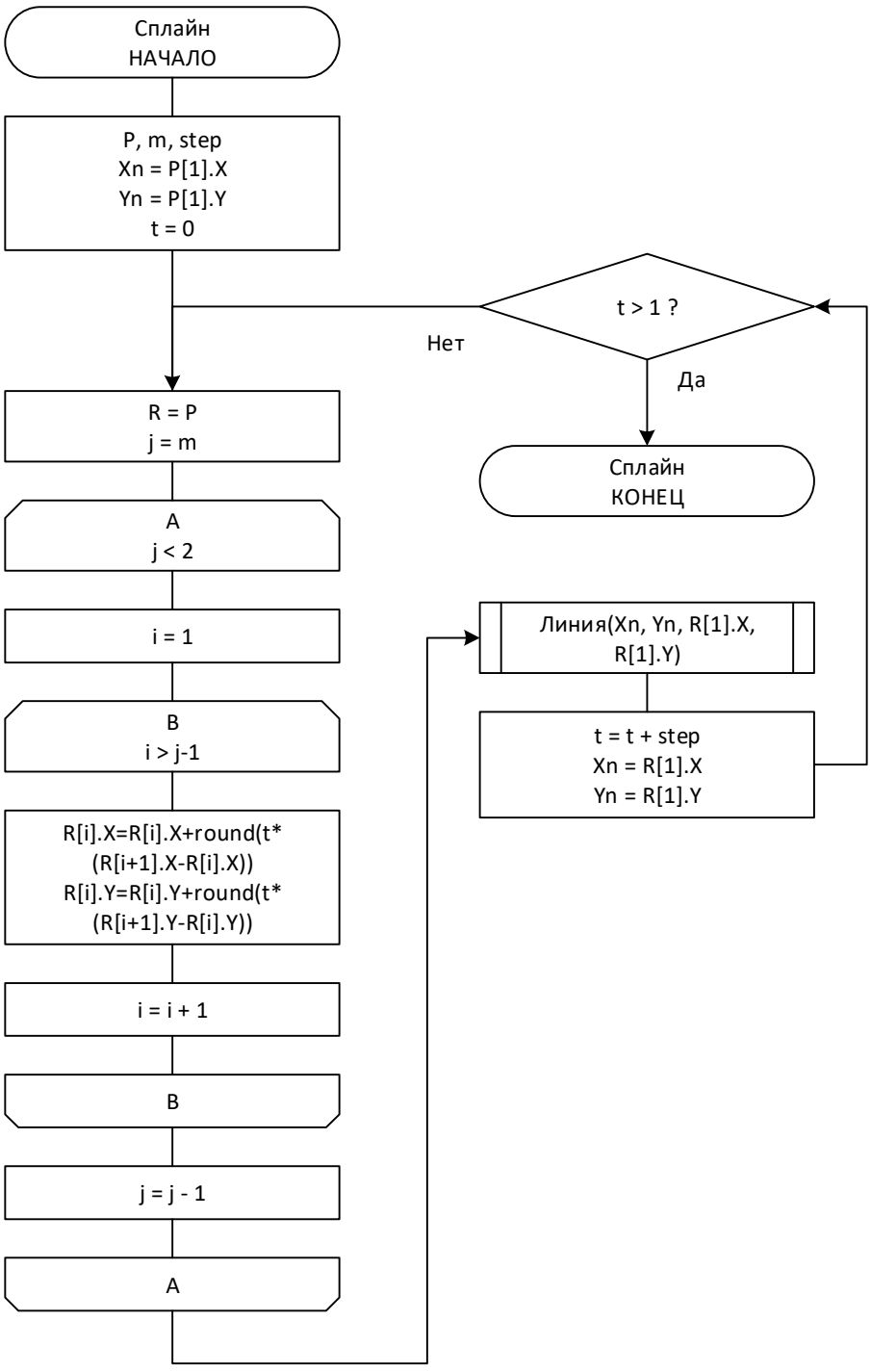

Рисунок А.1 – Схема алгоритма отрисовки сплайна

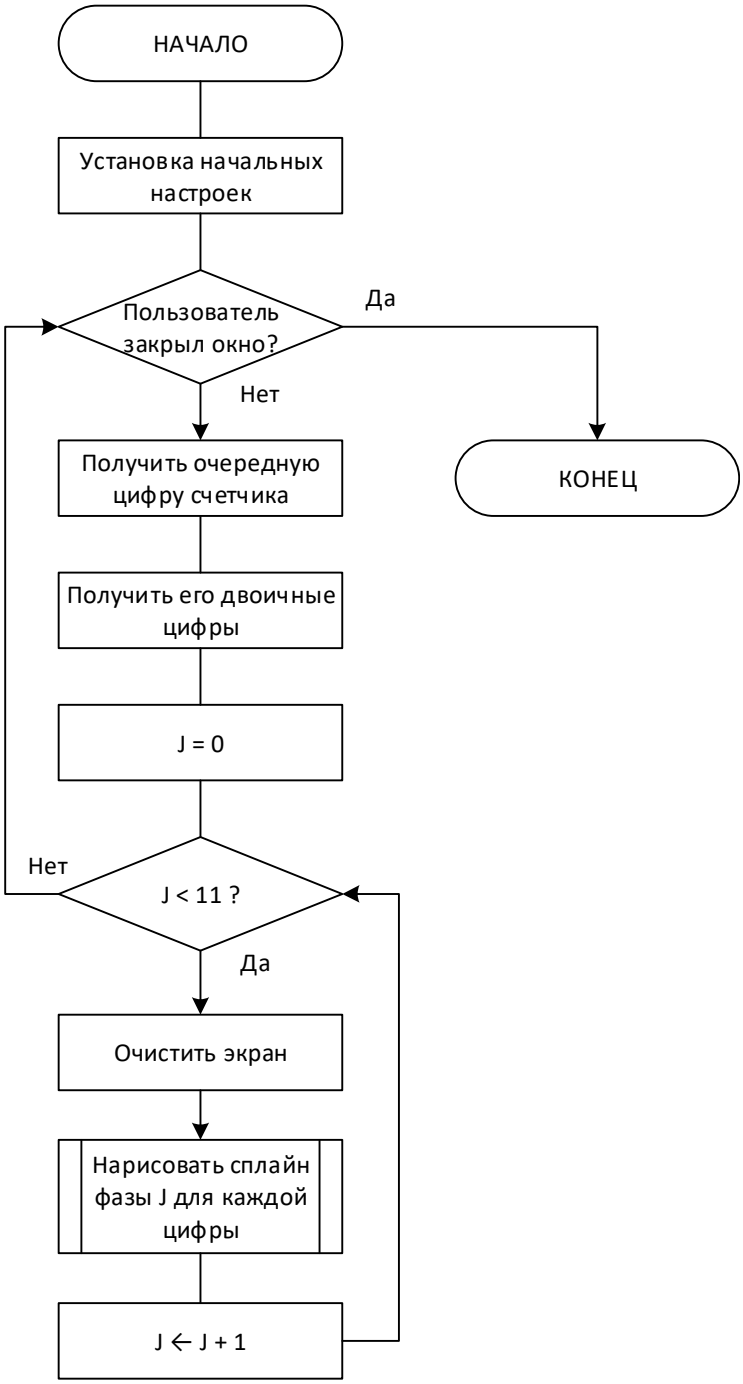

Рисунок А.2 – Схема алгоритма перерисовки изображения

Приложение Б
(обязательное)
Листинг программы

```
#include <iostream>
#include <fstream>
#include <functional>
#include <memory>
#include <vector>
#include <cmath>
#include <SDL.h>

class window_deleter {
public:
	void operator ()(SDL_Window *win) {
		SDL_DestroyWindow(win);
	}
};

class renderer_deleter {
public:
	void operator ()(SDL_Renderer *ren) {
		SDL_DestroyRenderer(ren);
	}
};

typedef std::unique_ptr < SDL_Window, window_deleter > win_ptr;
typedef std::unique_ptr < SDL_Renderer, renderer_deleter > ren_ptr;
typedef std::pair < int, int > pii;
typedef std::vector<std::vector<pii> > digit_points;

int myround(double val) {
  return (int)std::ceil(val - 0.5);
}

SDL_Point MakePoint(int x, int y) {
	SDL_Point p; p.x = x; p.y = y;
	return p;
}

void DrawCircle(const ren_ptr& ren, int x, int y, int r) {
	// отрисовка окружностей
	int x0 = x, y0 = y;
	x = 0; y = r;
	int e = 3 - 2 * r;
	std::vector<SDL_Point> points;
	while (x <= y) {
		points.emplace_back(MakePoint(x0 + x, y0 + y));
		points.emplace_back(MakePoint(x0 + y, y0 + x));
		points.emplace_back(MakePoint(x0 + y, y0 - x));
		points.emplace_back(MakePoint(x0 + x, y0 - y));
		points.emplace_back(MakePoint(x0 - x, y0 - y));
		points.emplace_back(MakePoint(x0 - y, y0 - x));
		points.emplace_back(MakePoint(x0 - y, y0 + x));
		points.emplace_back(MakePoint(x0 - x, y0 + y));
		if (e >= 0) {
			e += 4 * (x - y) + 10;
			y -= 1;
		}
		else {
			e += 4 * x + 6;
		}
		x += 1;
		SDL_RenderDrawPoints(ren.get(), points.data(), points.size());
	}
```

```
}

std::vector<pii> GetSplinePoints(const std::vector<pii>& points, double step) {
	// получение точек сплайна
	std::vector<pii> R = points, P = points, spline_points;
	int m = points.size();
	double t = 0;
	pii current_point( P[0] );
	spline_points.push_back(current_point);
	while (t < 1) {
		R = P;
		for (int j = m; j > 1; --j) {
		for (int i = 0; i < j-1; ++i) {
		R[i].first = R[i].first + myround(t * (R[i + 1].first - R[i].first));
		R[i].second = R[i].second + myround(t * (R[i + 1].second - R[i].second));
		}
		}

		t += step;
		current_point = R[0];
		spline_points.push_back(current_point);
	}
	return spline_points;
}

void DrawPolyline(const ren_ptr& ren, const std::vector<pii>& points, int bx, int by)
{
	// отрисовка сплайна
	for (std::vector<pii>::size_type i = 0; i < points.size(); i += 5) {
		DrawCircle(ren, points[i].first + bx, points[i].second + by, 2);
	}
}

std::vector<digit_points> LoadDigits() {
	std::ifstream f("points.txt");
	std::vector<digit_points> digits(2);

	for (int k = 0; k < 2; ++k) {
		int n; f >> n;
		digits[k].resize(n);
		for (int i = 0; i < n; ++i) {
			for (int j = 0; j < 4; ++j) {
				int x, y; f >> x >> y;
				digits[k][i].emplace_back(x, y);
			}
		}
	}

	return digits;
}

void DrawNumber(const ren_ptr& ren, const digit_points& a1, const digit_points& a2,
const digit_points& a3) {
	SDL_SetRenderDrawColor(ren.get(), 0xff, 0xff, 0xff, 0);
	SDL_RenderClear(ren.get());
	SDL_SetRenderDrawColor(ren.get(), 0x0, 0x0, 0x0, 0);

	DrawPolyline(ren, GetSplinePoints(a1[0], 0.01), 30, 30);
	DrawPolyline(ren, GetSplinePoints(a1[1], 0.01), 30, 30);

	DrawPolyline(ren, GetSplinePoints(a2[0], 0.01), 150, 30);
	DrawPolyline(ren, GetSplinePoints(a2[1], 0.01), 150, 30);

	DrawPolyline(ren, GetSplinePoints(a3[0], 0.01), 270, 30);
	DrawPolyline(ren, GetSplinePoints(a3[1], 0.01), 270, 30);
	SDL_RenderPresent(ren.get());
```

```
}

std::vector<digit_points> MakeTransform(const digit_points& first, const
digit_points& second) {
	std::vector<digit_points> tr(11);
	for (int i = 0; i < 11; ++i) {
		tr[i].resize(2);
		for (int j = 0; j < 2; ++j) {
		tr[i][j].resize(4);
		for (int p = 0; p < 4; ++p) {
		tr[i][j][p] = pii(
		first[j][p].first + (second[j][p].first - first[j][p].first) * i / 10,
		first[j][p].second + (second[j][p].second - first[j][p].second) * i / 10
		);
		}
		}
	}
	return tr;
}

void MainLoop() {
	win_ptr window( SDL_CreateWindow("Test", 100, 100, 640, 480, SDL_WINDOW_SHOWN)
);
	ren_ptr renderer(SDL_CreateRenderer(window.get(), -1,
SDL_RENDERER_ACCELERATED));

	std::vector<digit_points> digits = LoadDigits();
	std::vector<digit_points> transforms[2] = { MakeTransform(digits[0],
digits[1]), MakeTransform(digits[1], digits[0]) };

	SDL_Event e;
	SDL_WaitEvent(&e);
	while (e.type != SDL_KEYDOWN) {
		SDL_WaitEvent(&e);
	}

	for (int i = 1, prev = 0; ; ++i, ++prev) {
		// основной цикл отрисовки
		int a12 = i % 2, a22 = i / 2 % 2, a32 = i / 4 % 2,
			a11 = prev % 2, a21 = prev / 2 % 2, a31 = prev / 4 % 2;

		for (int j = 0; j < 11; ++j) {
			DrawNumber(renderer,
				(a31 != a32 ? transforms[a31][j] : digits[a32]),
				(a21 != a22 ? transforms[a21][j] : digits[a22]),
				(a11 != a12 ? transforms[a11][j] : digits[a11]));

			SDL_Event e;
			if (SDL_WaitEventTimeout(&e, 80)) {
				if (e.type == SDL_KEYDOWN) {
					return;
				}
			}
		}
		SDL_Delay(300);
	}
}

int main(int, char**){
	if (SDL_Init(SDL_INIT_VIDEO) != 0){
			return 1;
	}
	MainLoop();
	SDL_Quit();
	return 0;
}
```

Приложение В
(обязательное)
Экранные формы программы

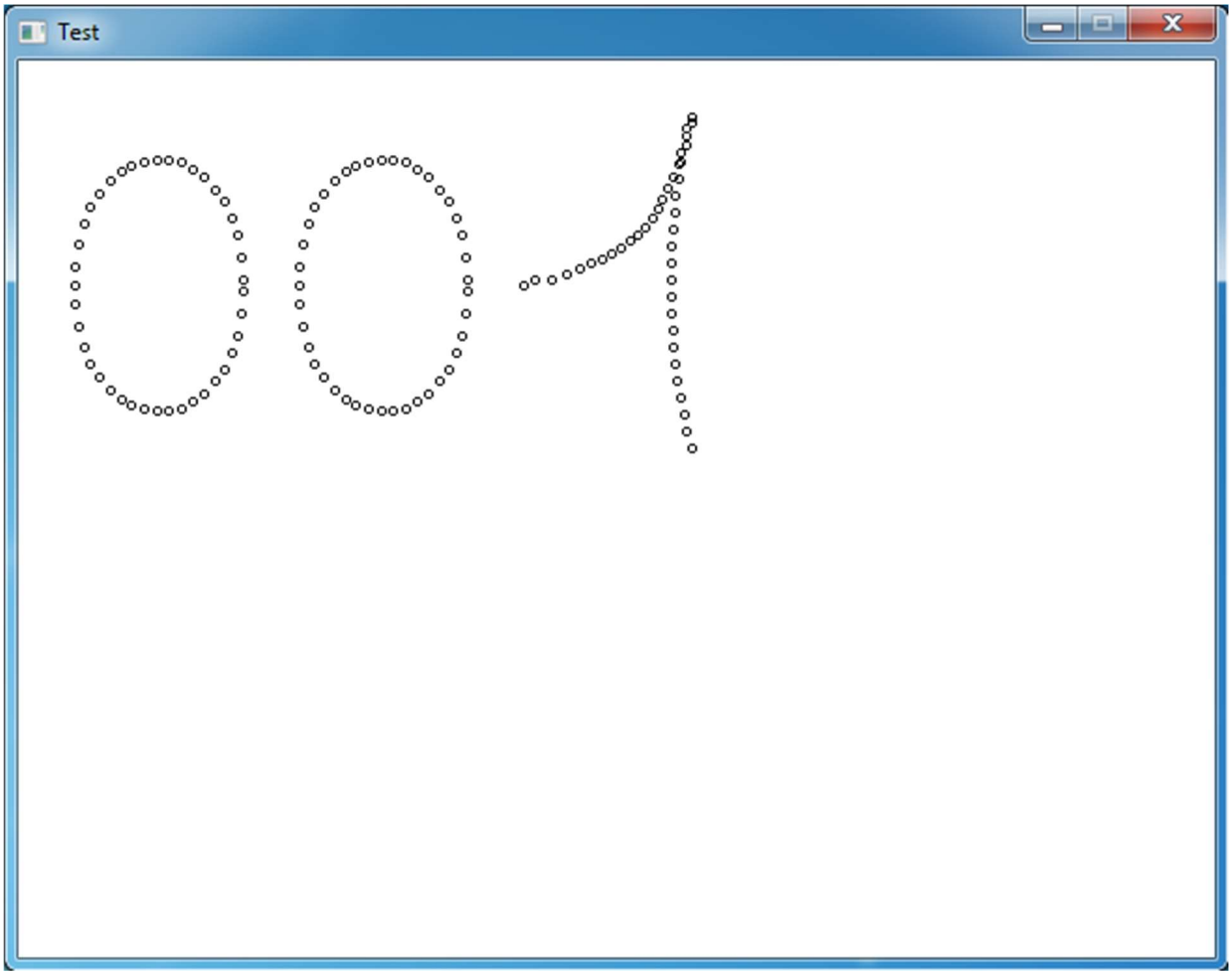

Рисунок В.1 – Основное окно программы