Третья лаба по КГ

Параметрические кривые

Параметрические кривые задаются в виде:

$$R(t) = \begin{cases} x = f_x(t) \\ y = f_y(t) \\ z = f_z(t) \end{cases}.$$

Вид функций f_x, f_y, f_z определяют форму кривой.

Для рисования кривых будет достаточно проекта для первой лабы https://github.com/gavreg/grafika_lab1.

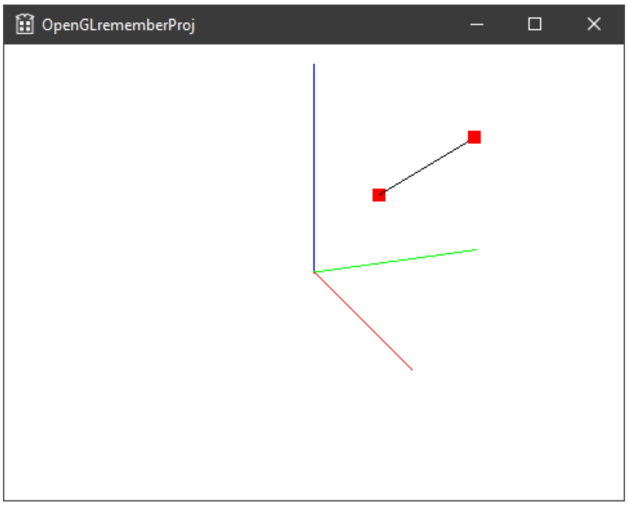
Например, давайте построим отрезок AB

Каждая точка p такого отрезка будет вычислится по формуле

$p(t) = A(t-1) + Bt, t \in [0,1]$, вычислять отдельно для x,y,z.

Давайте спрограммируем:

```
double f(double a, double b, double t)
{
	return a * (1 - t) + b * t;
}

void Render(double delta_time)
{
	double A[] = {1, 3, 4 };  //точки
	double B[] = { 4, 6, 7 };

	glBegin(GL_LINE_STRIP);   //режим рисования линии по точкам,
	//в этом режиме точки соединяются подряд 1-2-3-4-5-6...
	//а не  1-2   3-4   5-6 как с
	for(double t=0; t<=1.0001; t+=0.01)//1.0001 – нет времени объяснять, just do it
	{
		double x = f(A[0], B[0], t);
		double y = f(A[1], B[1], t);
		double z = f(A[2], B[2], t);

		glVertex3d(x, y, z);
	}
	glEnd();

	glPointSize(10); //размер точек
	glColor3d(1, 0, 0);
	glBegin(GL_POINTS); //нарисуем точки A и B
	glVertex3dv(A);
	glVertex3dv(B);
	glEnd();
}
```



А вот это окружность с радиусом 5

```
	glBegin(GL_LINE_STRIP);

	for(double t=0; t<=2*3.1415; t+=0.01)
	{
		double x = 5*cos(t);
		double y = 5*sin(t);
		double z = 0;

		glVertex3d(x, y, z);
	}
	glEnd();
```

В случае окружности $t \in [0, 2\pi]$.

Теперь построим кривую второго порядка.

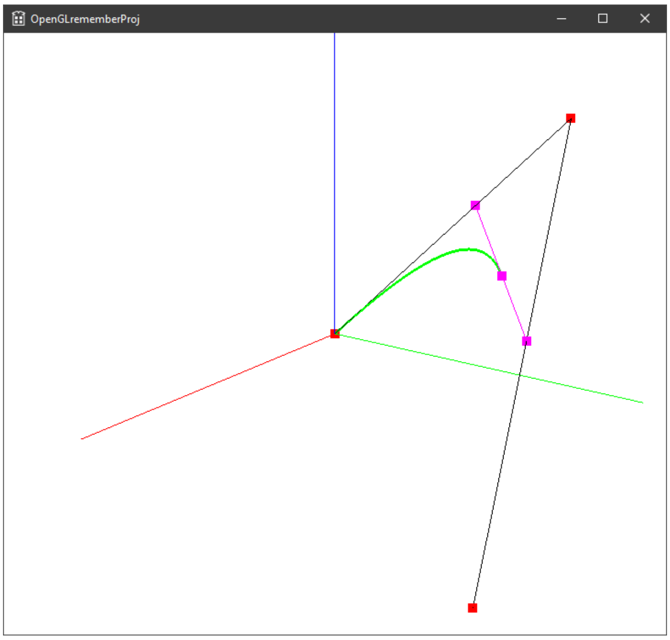
У нас есть две точки $P1, P2, P3$. Соберем из них два отрезка $P1P2$ и $P2P3$. Построим по ним параметрическую кривую первого порядка (прямую, т.е) как это мы делали ранее. Строим мы ее одновременно при $t \in [0,1]$. Представим, что мы находимся в середине этого процесса, при $t = k$. Т.е. мы находимся в некоторой точке недостроенного отрезка $P1P2$ и в какой-то точке недостроенного $P2P3$. Создадим из этих точек отрезок AB, и вычислим на нем точку при $t = k$. Вот из множества таких точек сформируется кривая второго порядка.

Код, который реализует построение кривой второго порядка с анимацией

```
double f(double a, double b, double t)
{
	return a * (1 - t) + b * t;
}

double t_max = 0;
void Render(double delta_time)
{

	t_max += delta_time / 5; //t_max становится = 1 за 5 секунд
	if (t_max > 1) t_max = 0; //после обнуляется
```

```
double P1[]=  { 0,0,0 }; //Наши точки, массивчик double
double P2[] = { -4,6,7 };
double P3[] = { 10,10,0 };

double A[3];   //промежуточные точки
double B[3];
double P[3];

glBegin(GL_LINES); //построим отрезки P1P2  и P2P3
glVertex3dv(P1);
glVertex3dv(P2);
glVertex3dv(P2);
glVertex3dv(P3);
glEnd();

glLineWidth(3);  //ширина линии

glColor3d(0, 1, 0);

glBegin(GL_LINE_STRIP);

for(double t=0; t<=t_max; t+=0.01)
//Мы теперь перебираем t не до 1, а до t_max
//так как t_max сама по себе изменяется от 0 до 1 постепенно от кадра к кадру
//и после сбрасывается (см. начало ф-ции Render)
//у нас получится анимация построения отрезка, потому что мы его строим не до
//конца, но все дальше и дальше.
{
       A[0] = f(P1[0], P2[0], t);  //Считаем точку A на отрезке P1P2
       A[1] = f(P1[1], P2[1], t);
       A[2] = f(P1[2], P2[2], t);

       B[0] = f(P2[0], P3[0], t); //Считаем точку B на отрезке P2P3
       B[1] = f(P2[1], P3[1], t);
       B[2] = f(P2[2], P3[2], t);

       P[0] = f(A[0], B[0], t);  //Считаем точку P на отрезке AB
       P[1] = f(A[1], B[1], t);
       P[2] = f(A[2], B[2], t);

       glVertex3dv(P);   //Рисуем точку P
}
glEnd();

glColor3d(1, 0, 1);
glLineWidth(1);  //возвращаем ширину линии = 1
glBegin(GL_LINES);  //построим отрезок AB
glVertex3dv(A);         //Из точек вычисленных при t=t_max (цикл то закончился)
glVertex3dv(B);
glEnd();

//нарисуем все точки
glPointSize(10);
glBegin(GL_POINTS);
glVertex3dv(P);
glVertex3dv(A);
glVertex3dv(B);
glEnd();

glColor3d(1, 0, 0);
glBegin(GL_POINTS);
glVertex3dv(P1);
glVertex3dv(P2);
glVertex3dv(P3);
```

```
        glEnd();
}
```



На самом деле, мы сейчас решили задачу в лоб. Строя «виртуальные» отрезки, брали на нем промежуточную точку и делали нашу кривую из этих промежуточных точек. Все это можно математически посчитать, упросить и не задумывается о промежуточных отрезках.

$A(t) = P1(1-t) + P2t$ - промежуточная точка на отрезке $P1P2$

$B(t) = P2(1-t) + P3t$ - промежуточная точка на отрезке $P2P3$

$P(t) = A(1-t) + Bt$ - промежуточная точка на отрезке AB, из них получается наша кривая.

Подставляем выражения для точек A и B в выражение для точки P:

$$P(t) = \left(P1(1-t) + P2t\right)\left(1-t\right) + \left(P2(1-t) + P3t\right)t =$$
$$= P1(1-t)^2 + P2t(1-t) + P2(1-t)t + P3t^2 =$$
$$= P1(1-t)^2 + 2 \cdot P2t(1-t) + P3t^2$$

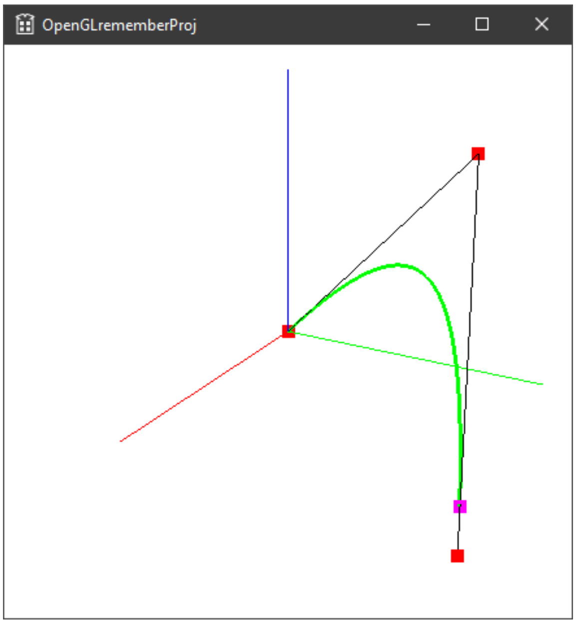
Таким образом, чтобы не мучится с промежуточными отрезками, нам надо всего лишь посчитать все точки кривой по формуле.

$$P(t) = P1(1-t)^2 + 2P2t(1-t) + P3t^2$$

Убедимся на практике:

```
double f(double p1, double p2, double p3, double t)
{
        return p1*(1-t)*(1-t)+2*p2*t*(1-t)+p3*t*t;  //посчитанная формула
}
```

```cpp
double t_max = 0;
void Render(double delta_time)
{
	t_max += delta_time / 5; //t_max становится = 1 за 5 секунд
	if (t_max > 1) t_max = 0; //после обнуляется

	double P1[] = { 0,0,0 }; //Наши точки
	double P2[] = { -4,6,7 };
	double P3[] = { 10,10,0 };

	double P[3];


	glBegin(GL_LINES_STRIP); //построим отрезки P1P2  и P2P3
	glVertex3dv(P1);
	glVertex3dv(P2);
	glVertex3dv(P3);
	glEnd();

	glLineWidth(3);  //ширина линии

	glColor3d(0, 1, 0);

	glBegin(GL_LINE_STRIP);
	for (double t = 0; t <= t_max; t += 0.01)
	{
		P[0] = f(P1[0], P2[0], P3[0], t);
		P[1] = f(P1[1], P2[1], P3[1], t);
		P[2] = f(P1[2], P2[2], P3[2], t);

		glVertex3dv(P);   //Рисуем точку P
	}
	glEnd();

	glColor3d(1, 0, 1);
	glLineWidth(1);  //возвращаем ширину линии = 1

	//нарисуем все точки
	glPointSize(10);
	glBegin(GL_POINTS);
	glVertex3dv(P);
	glEnd();

	glColor3d(1, 0, 0);
	glBegin(GL_POINTS);
	glVertex3dv(P1);
	glVertex3dv(P2);
	glVertex3dv(P3);
	glEnd();
}
```

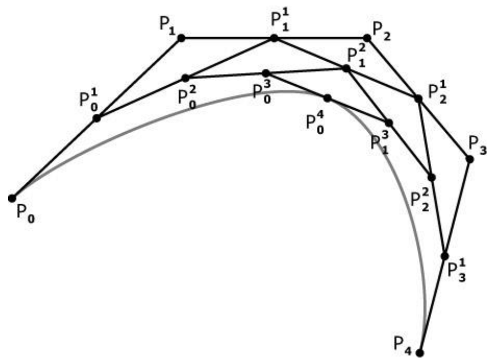


Таким образом, кривая рисуется не сложнее отрезка (изменилась только формула в ф-ции f). Код получился довольно объемным, так как помимо кривой рисовали еще всякие красивости. – если их убрать будет просто

```
//пометим ф-цию inline для более быстрого выполнения
inline double f(double p1, double p2, double p3, double t)
{
	return p1*(1-t)*(1-t)+2*p2*t*(1-t)+p3*t*t;  //посчитаная формула
}

void Render(double delta_time)
{

	double P1[] = { 0,0,0 }; //Наши точки
	double P2[] = { -4,6,7 };
	double P3[] = { 10,10,0 };

	glLineWidth(3);  //ширина линии

	glBegin(GL_LINE_STRIP);

	for (double t = 0; t <= 1.0001; t += 0.01)
	{
		double P[3];

		P[0] = f(P1[0], P2[0], P3[0], t);
		P[1] = f(P1[1], P2[1], P3[1], t);
		P[2] = f(P1[2], P2[2], P3[2], t);

		glVertex3dv(P);   //Рисуем точку P
	}
	glEnd();

	glLineWidth(1);  //возвращаем ширину линии = 1
}
```

Поздравляю! Вы нарисовали **кривую Безье** второго порядка!

Аналогично есть кривые третьего и более высоких порядков. (т.е мы считаем промежуточные точки на отрезках, построенных из точек на промежуточных отрезках итд.)



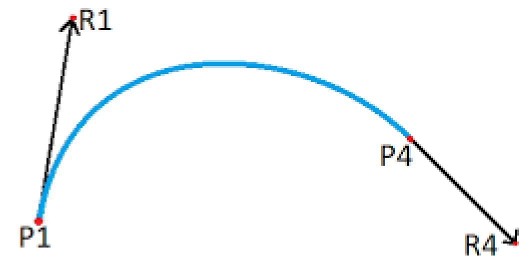
К счастью, все уже давно посчитано!

Задача лабораторной состоит в построении двух типов кривых третьего порядка:

кривой Эрмита и кривой Безье. Эти кривые часто используются в различных САПР как удобный математический аппарат, позволяющий управлять кривыми линиями.

Как их строить:

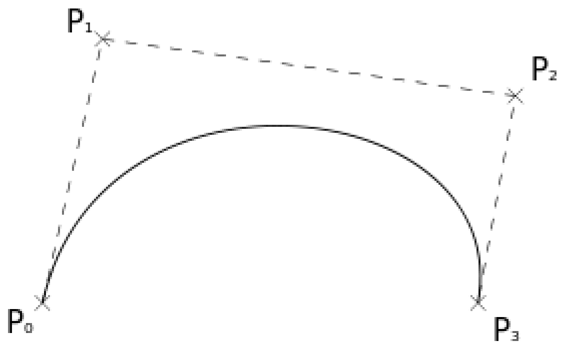
Кривая Эрмита получается следующим образом



$$\overrightarrow{X}(t) = \overrightarrow{P_1}(2t^3 - 3t^2 + 1) + \overrightarrow{P_4}(-2t^3 + 3t^2) + \overrightarrow{R_1}(t^3 - 2t^2 + t) + \overrightarrow{R_4}(t^3 - t^2)$$

Она начинается в точке P_1, по касательной к вектору $\overrightarrow{R_1}$ и заканчивается в точке P_4 по касательной к вектору $\overrightarrow{R_4}$. Обратите внимание что, P_1, P_4 это **точки**, а $\overrightarrow{R_1}, \overrightarrow{R_4}$ - **вектора** (что, в принципе, разность координат пары точек).

С кривой Безье второго порядка мы уже познакомились. Кривая третьего порядка выглядит так:



$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3, \quad t \in [0,1].$$

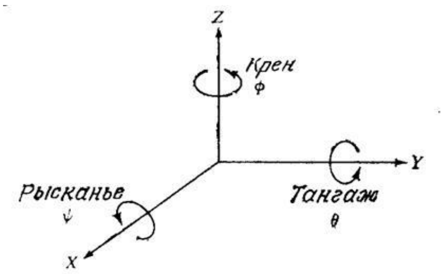
Начинаясь в нулевой точке, стримится приблизится ко первой, затем ко второй и заканчивается в третей.

Задание на Л/Р:

30 – Построить две кривые Эрмита и две кривые Безье (без анимации), для Эрмита нарисовать вектора $\overrightarrow{R_1}$, $\overrightarrow{R_4}$, для Безье – ломаную $P_0P_1P_2P_3$.

35– Заставить по любой кривой двигается какой ни будь объект, от начала до конца и от конца и к началу.

40 – Заставить по любой кривой двигаться какой ни будь объект (не симметричный, как шар), с расчетом углов крена, тангажа и рысканья, подобно тому как ракета разворачивается, преследуя цель.



45 – нарисовать поверхность Безье из точек. Формулы тут.

50 – нарисовать поверхность Безье из линий

54 – нарисовать поверхность Безье, с расчетом нормалей и наложить на нее текстуру.

60 – Поверхностью Безье (с нормалями и текстурой) можно управлять, перемещая точки мышкой.

См. комментарий к лабораторной https://youtu.be/KOEWjOwTQNU

Бонус: кривая Безье 8го порядка., код https://pastebin.com/cdfhHg0F