计算机图形学

# 第六讲：曲线造型技术

## –Bézier曲线

主讲老师：杨垠晖 博士

手机：13732297585

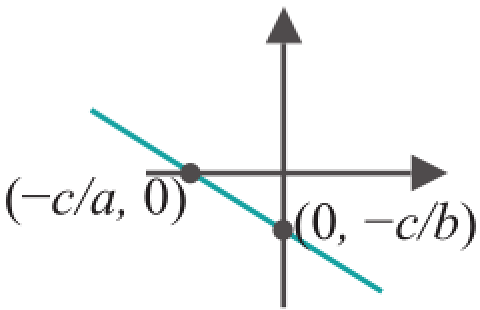
Email: yhyang@zafu.edu.cn

# 本讲内容

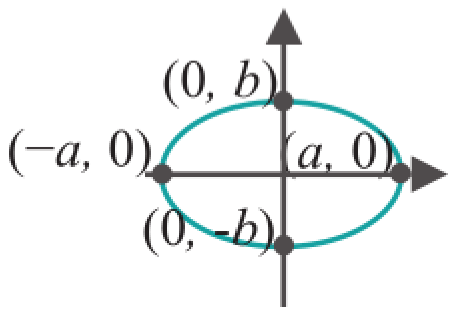
- 曲线造型技术概览
- 一次Bézier曲线
- 二次Bézier曲线
- 三次Bézier曲线
- 广义Bézier曲线
- OpenGL实践

# 曲线造型技术

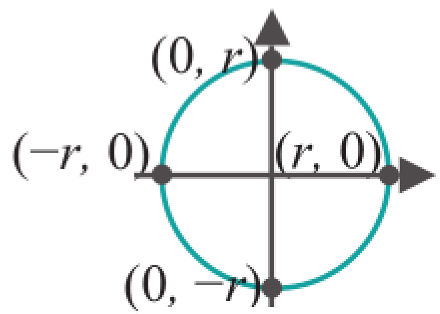
- 常见数学曲线



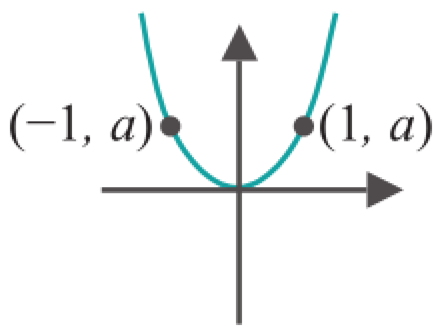

直线



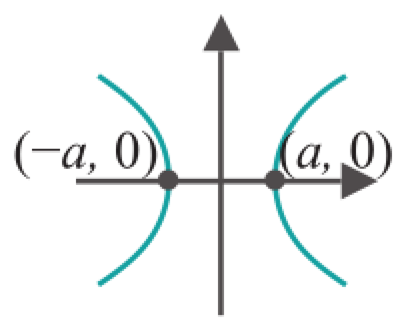

椭圆




圆



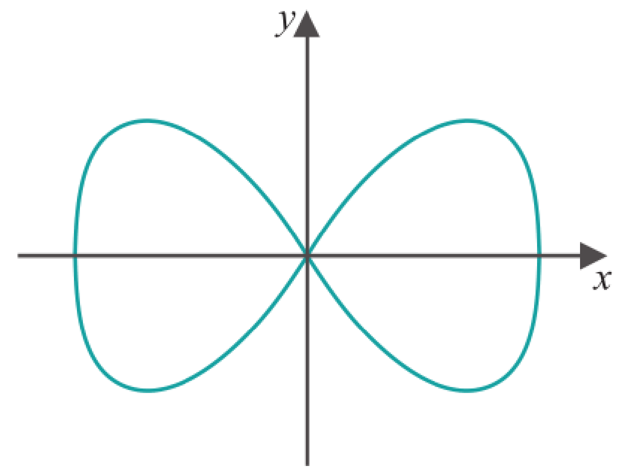

抛物线




双曲线

# 曲线造型技术

- 曲线的数学表达
- 隐式表达： $F(x,y)=0$
  - 直线：$ax+by+c=0$
  - 圆： $x^2+y^2=r^2$
- 参数表达：$x=f(t), y=g(t), t\in T$
  - 直线： $x=t,\ y=-\frac{a}{b}t-\frac{c}{b}t,\ t\in(-\infty,\infty)$
  - 圆： $x=r\cos(t),\ y=r\sin(t),\ t\in[-\pi,\pi]$




$x^2(x^2-1)+y^2=0$

$x=\cos(t),\ y=\cos(t)\sin(t),\ t\in[-\pi,\pi]$

# 曲线造型技术

- 多项式参数化

$$x = f(t), y = g(t), t \in T$$

- 如果f,和g是有关t的多项式函数，则称曲线c:

$$c = \{(x = f(t), y = g(t)) | t \in T\}$$

是具有多项式参数化的多项式曲线。

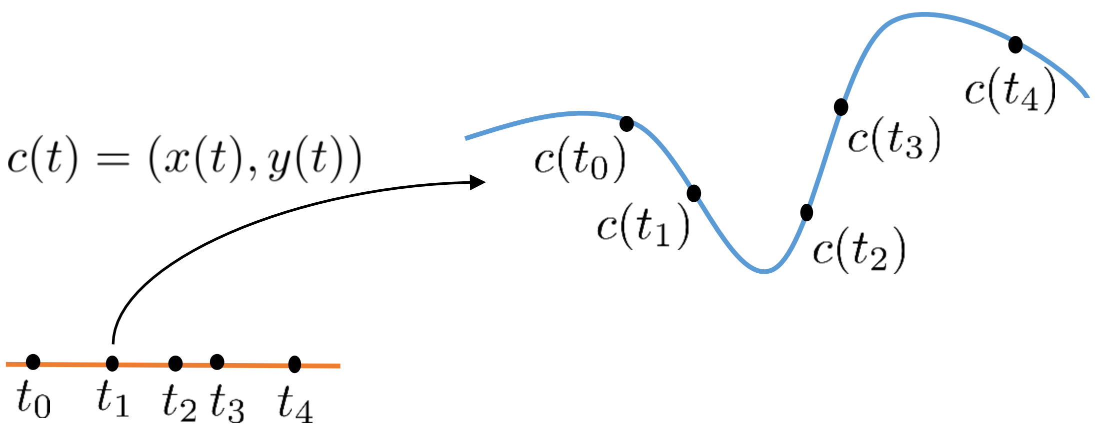
- 例如：$f(t) = a_0 + a_1 t + a_2 t^2$, $g(t) = b_0 + b_1 t + b_2 t^2$

# 曲线造型技术

- 多项式曲线

$$x(t) = a_0 + a_1 t + a_2 t^2,\ y(t) = b_0 + b_1 t + b_2 t^2,\ \ t \in [t_1, t_2]$$

- 可以通过调节多项式系数来生成不同的曲线。

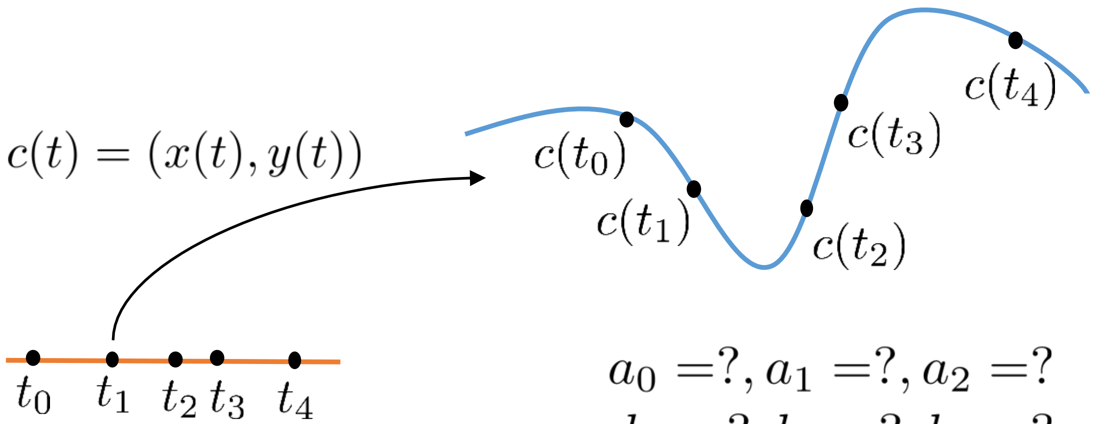


# 曲线造型技术

- 多项式曲线

$$x(t) = a_0 + a_1 t + a_2 t^2,\ y(t) = b_0 + b_1 t + b_2 t^2,\ \ t \in [t_1, t_2]$$

- 可以通过调节多项式系数来生成不同的曲线。



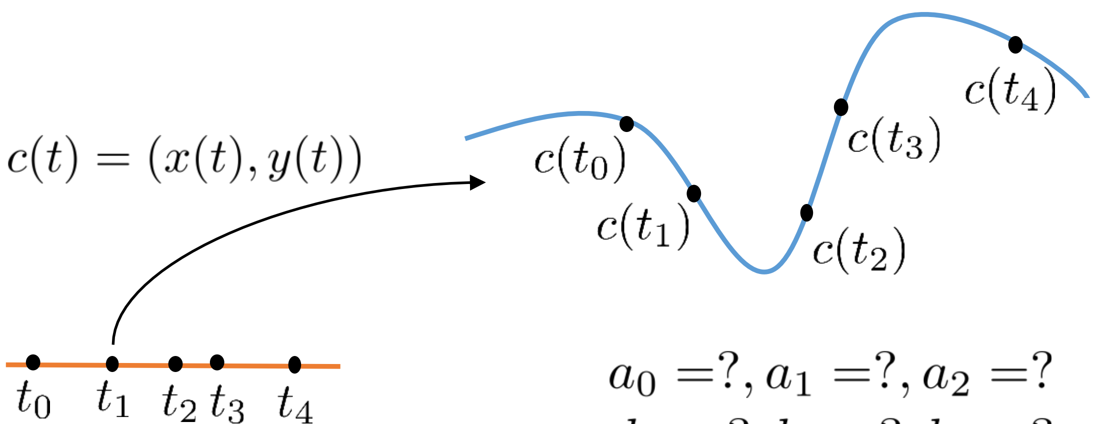

$$a_0 = ?, a_1 = ?, a_2 = ?$$
$$b_0 = ?, b_1 = ?, b_2 = ?$$

# 曲线造型技术

- 多项式曲线

$$x(t) = a_0 + a_1 t + a_2 t^2,\ y(t) = b_0 + b_1 t + b_2 t^2,\ \ t \in [t_1, t_2]$$

- 可以通过调节多项式系数来生成不同的曲线。

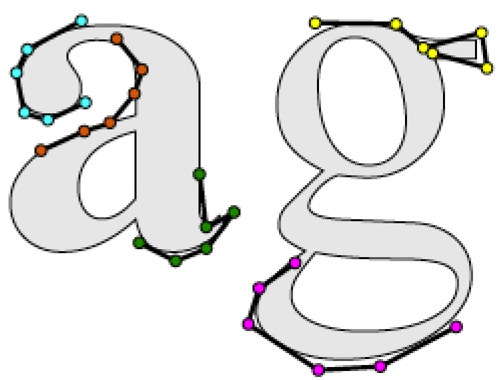



$$a_0 =?, a_1 =?, a_2 =?$$
$$b_0 =?, b_1 =?, b_2 =?$$

多项式系数的选取与曲线的最终形状缺乏直观联系！

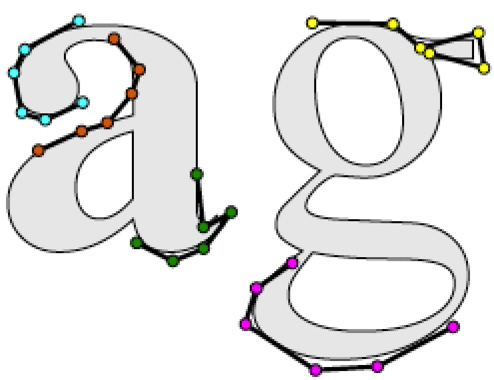
# 曲线造型技术

- 需要直观、灵活的曲线造型技术



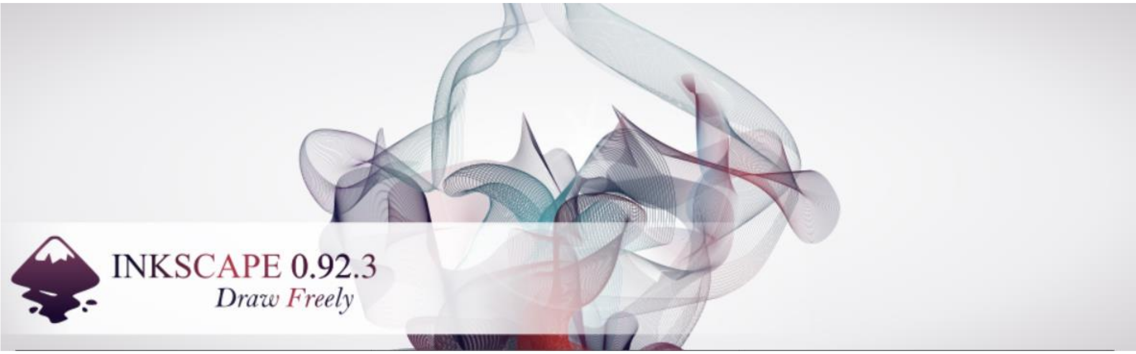
# 曲线造型技术

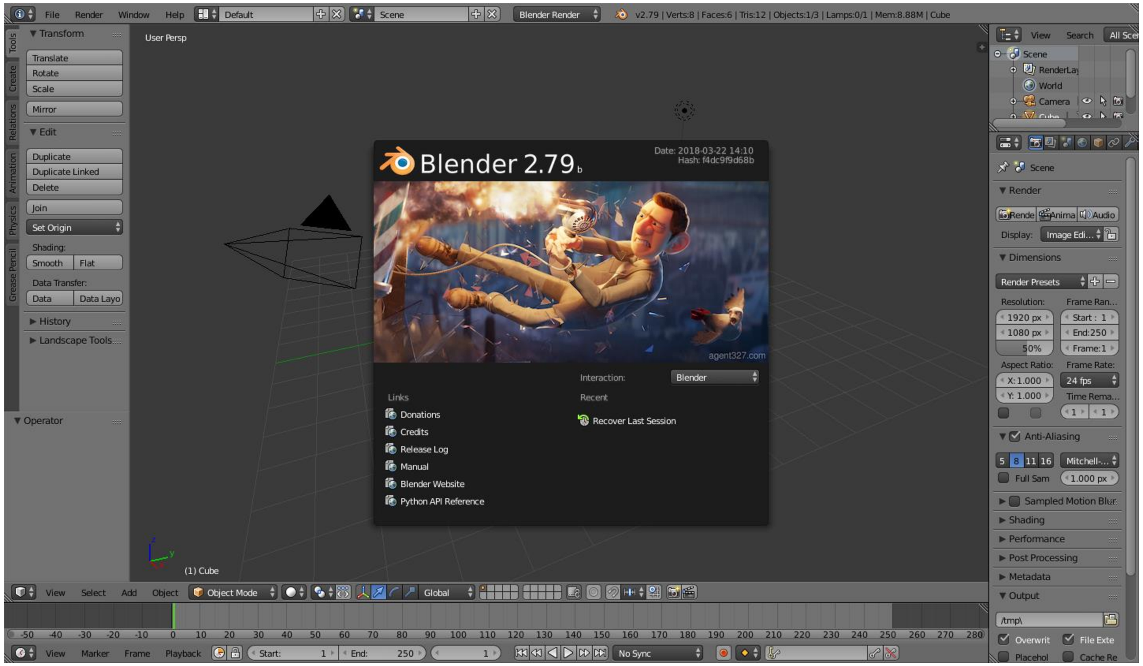
- 需要直观、灵活的曲线造型技术
  - Bézier曲线
  - B样条曲线



# Bézier曲线

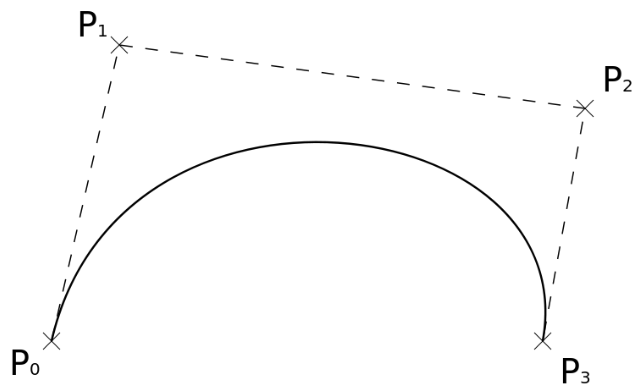
- 应用广泛
- 图形学相关
  - 建模
  - 动画
- 矢量图
- 交互界面
- 机器人

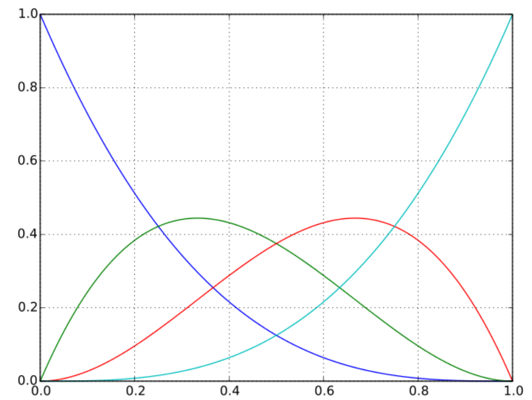


# Bézier曲线

- 应用广泛
- 图形学相关
  - 建模
  - 动画
- 矢量图
- 交互界面
- 机器人

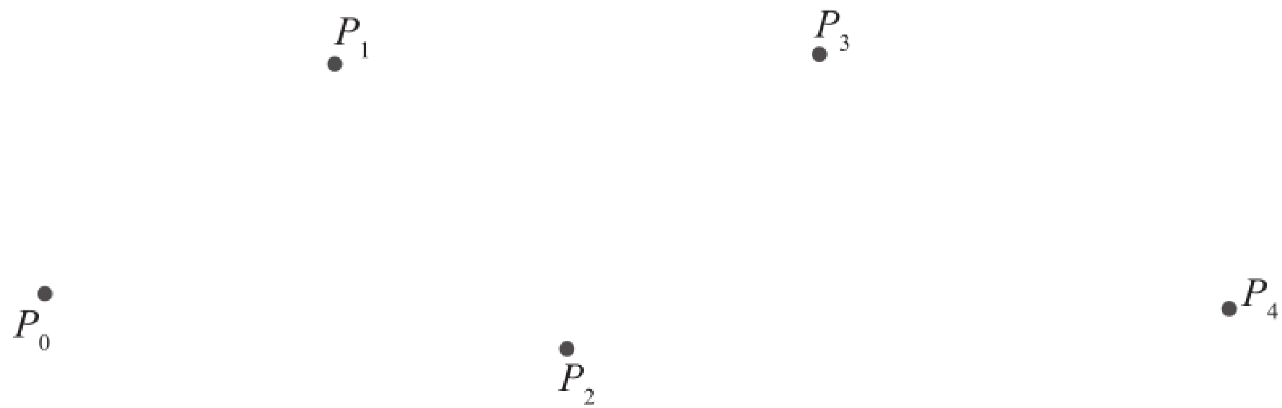



在1960年左右，由法国的两位汽车工程师以及数学家**Pierre Bézier**和**Paul de Casteljau**分别独立提出。

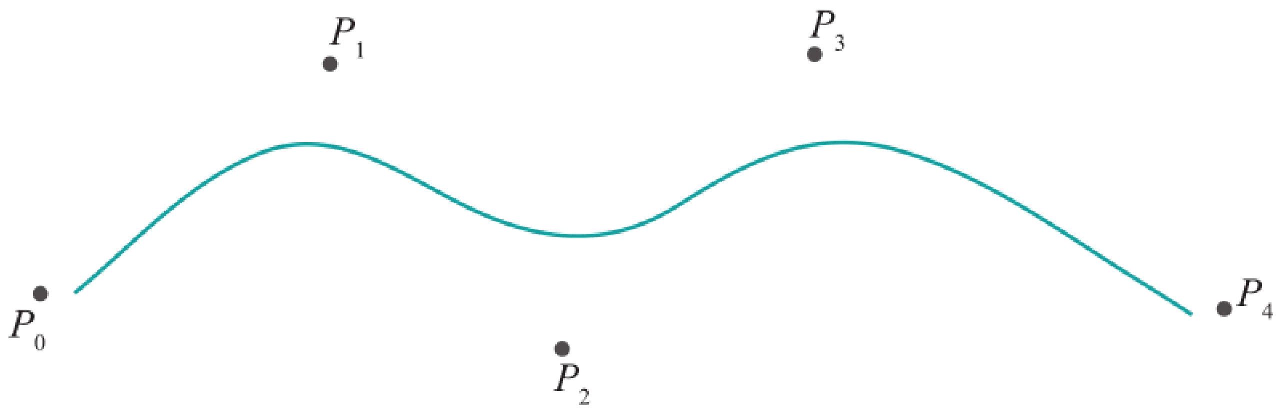
# Bézier曲线

- 概览



# Bézier曲线

- 概览



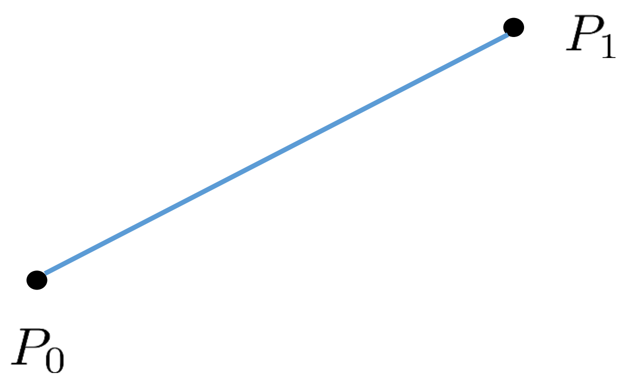

$P_0, P_1, P_2, P_3$ 称为控制顶点

# 一次Bézier曲线

- 两个控制顶点

# 一次Bézier曲线

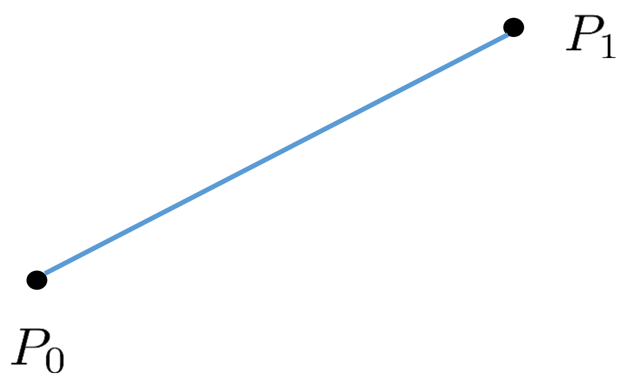
- 两个控制顶点




$$c(u) = (1-u)P_0 + uP_1,\ \ u \in [0,1]$$

# 一次Bézier曲线

- 两个控制顶点



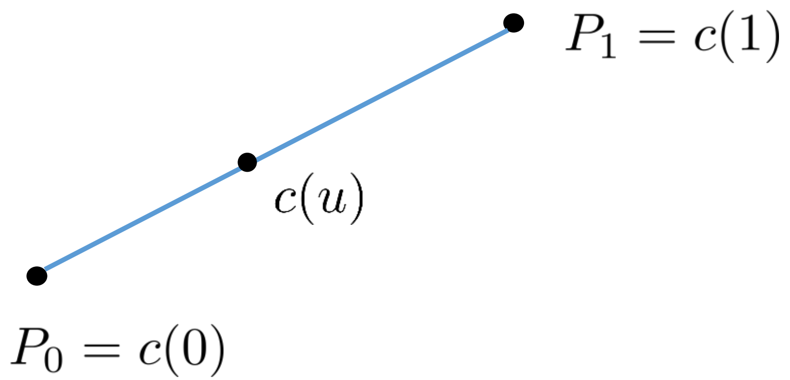

$$c(u) = (1-u)P_0 + uP_1,\ \ u \in [0,1]$$

**小练习**：

请计算由控制点**(5, 1)**和**(-1, 0)**确定的Bézier曲线方程，并计算u分别为**0.0**, **0.3**以及**1.0**时所对应曲线上点的坐标。

# 一次Bézier曲线

- 两个控制顶点




$$c(u) = (1-u)P_0 + uP_1,\ \ u \in [0,1]$$

# 一次Bézier曲线

$$c(u) = (1-u)P_0 + uP_1,\ u \in [0,1]$$

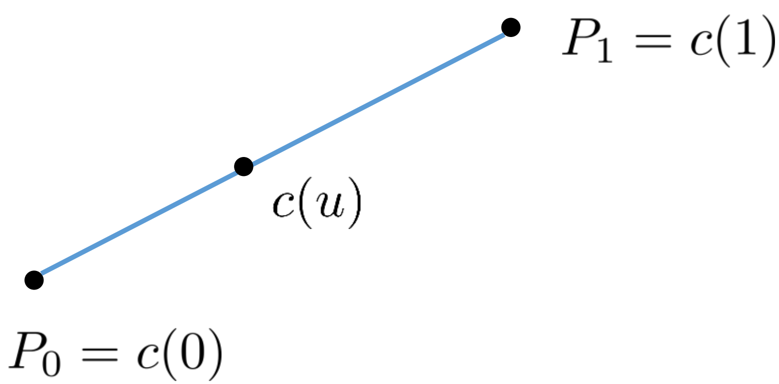
- 几点观察：
- c(u)是u的线性函数;
- 点c(u)是点$P_0$和$P_1$的加权和，权重分别为(1-u)以及u；上述权重称为混合函数或者基函数;
- (1-u)以及u为Bernstein一次多项式，表示为$B_{0,1}(u), B_{1,1}(u)$；
- $B_{0,1}(u)$和$B_{1,1}(u)$取值都在0和1之间，且满足：

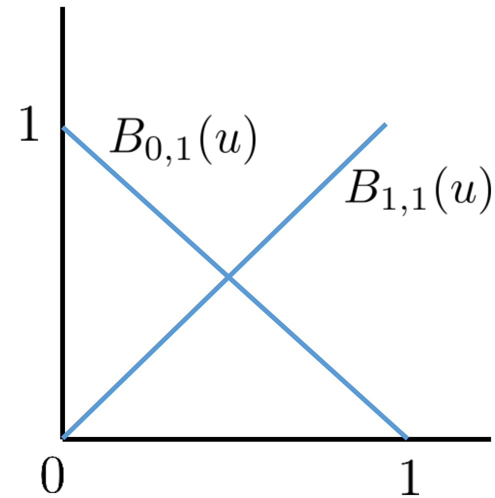
$$B_{0,1}(u) + B_{1,1}(u) = 1,\ \forall u \in [0,1]$$

- $P_0$ 对应的u=0, $P_1$ 对应的u=1，称曲线在点 $P_0$和$P_1$处进行了插值.

# 一次Bézier曲线

- Bernstein一次多项式函数

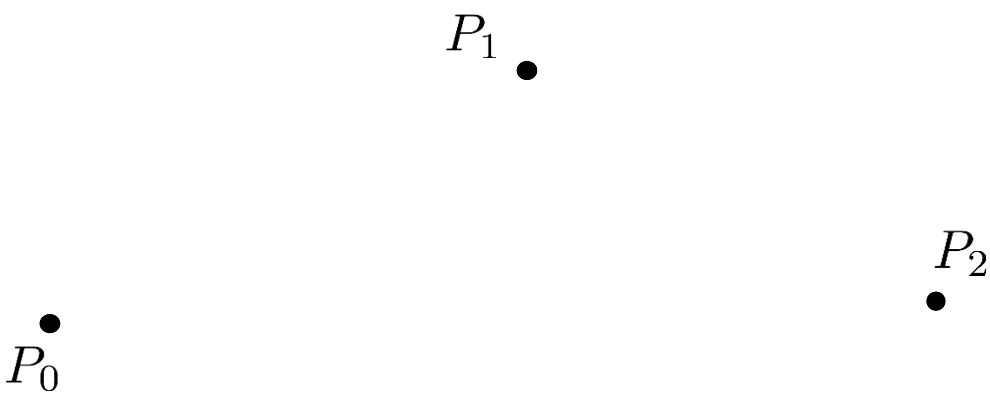



$$c(u) = B_{0,1}(u)P_0 + B_{1,1}(u)P_1,\ \ u \in [0,1]$$

# 二次Bézier曲线

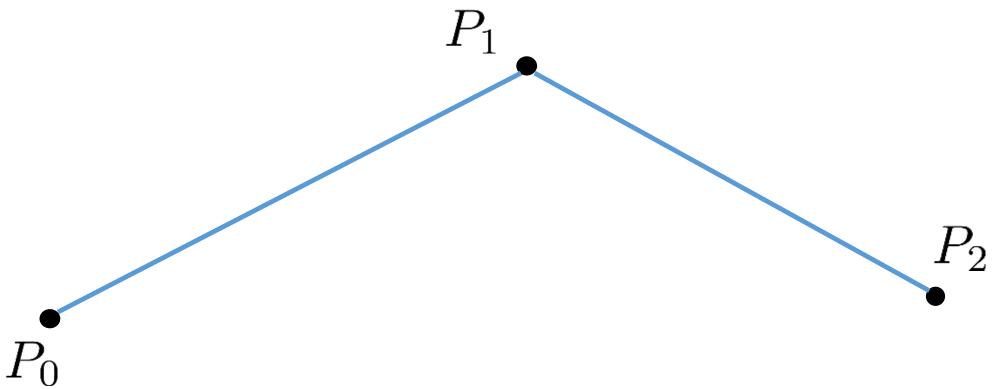
- 三个控制顶点




*是否能通过类似线性插值的方式得到基于三个控制点的光滑曲线？*

# 二次Bézier曲线

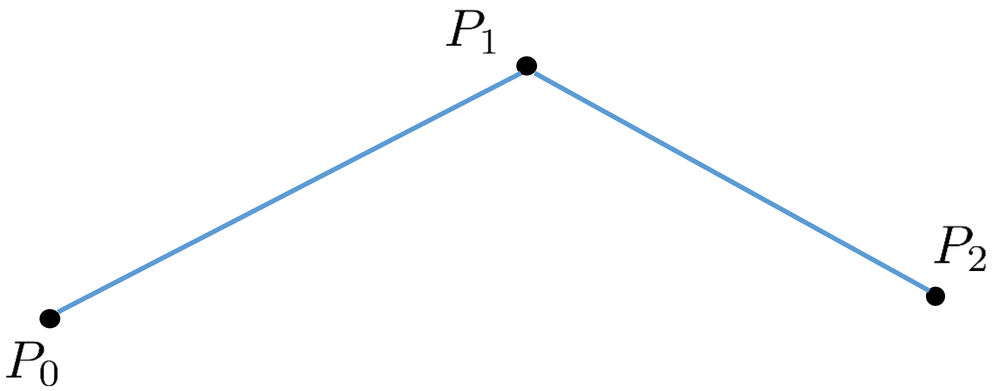
- 三个控制顶点




*尝试1：直接用直线连接各个顶点*

# 二次Bézier曲线

- 三个控制顶点

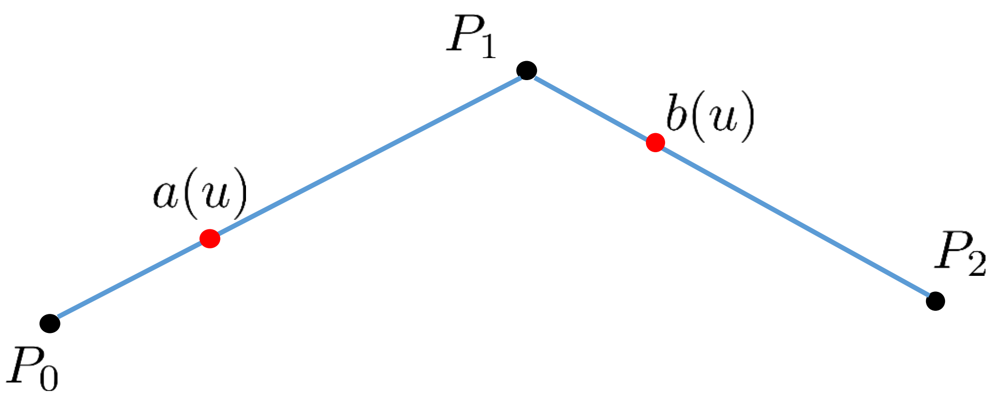



*尝试1：直接用直线连接各个顶点*

*缺点：不够光滑，在点P1处的插值并不理想*

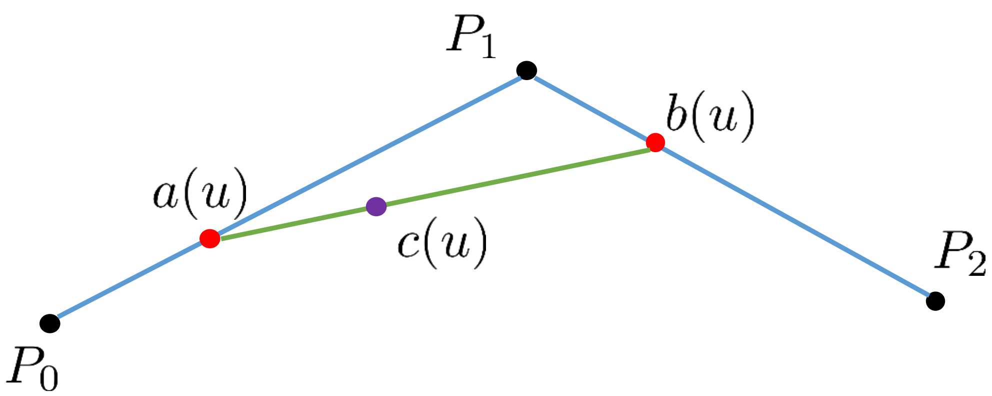
# 二次Bézier曲线

- 三个控制顶点



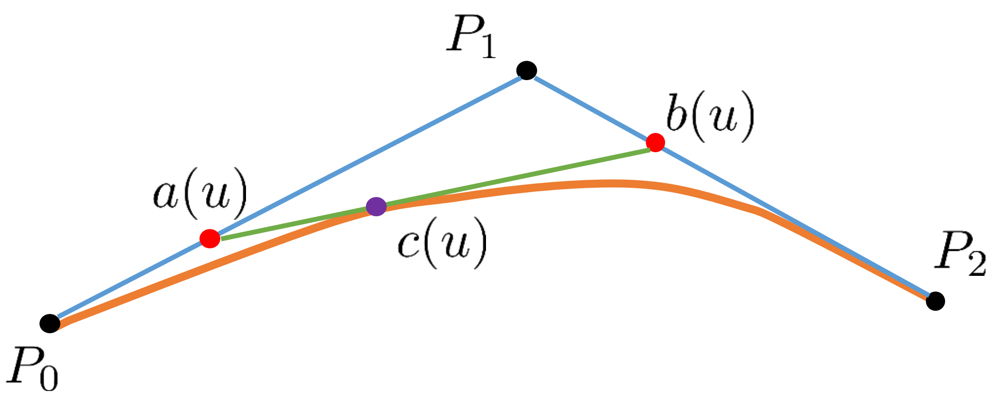
# 二次Bézier曲线

- 三个控制顶点



# 二次Bézier曲线

- 三个控制顶点

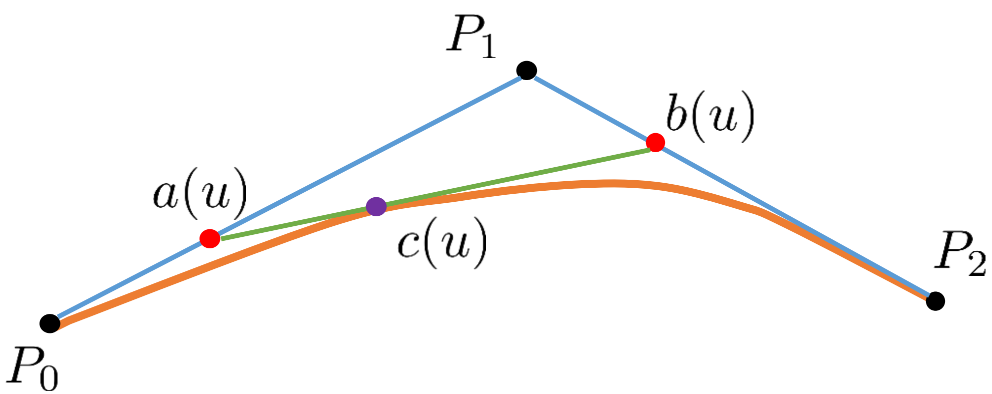



*尝试2：经过两次插值，将得到一条经过起点P0和终点P2的光滑曲线。该方法由* ***Paul de Casteljau*** *提出，称为* ***de Casteljau*** *算法。*

# 二次Bézier曲线

- 三个控制顶点

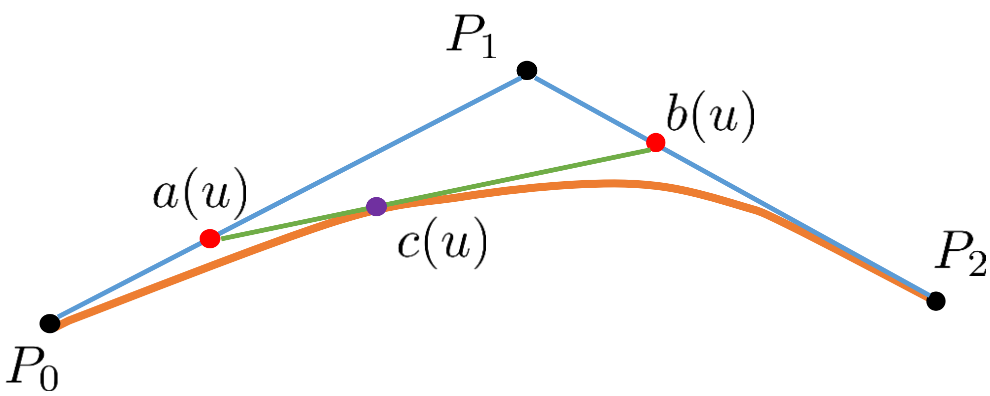
OpenGL Demo演示




*尝试2：经过两次插值，将得到一条经过起点P0和终点P2的光滑曲线。该方法由* ***Paul de Casteljau****提出，称为****de Casteljau****算法。*

# 二次Bézier曲线

- 三个控制顶点

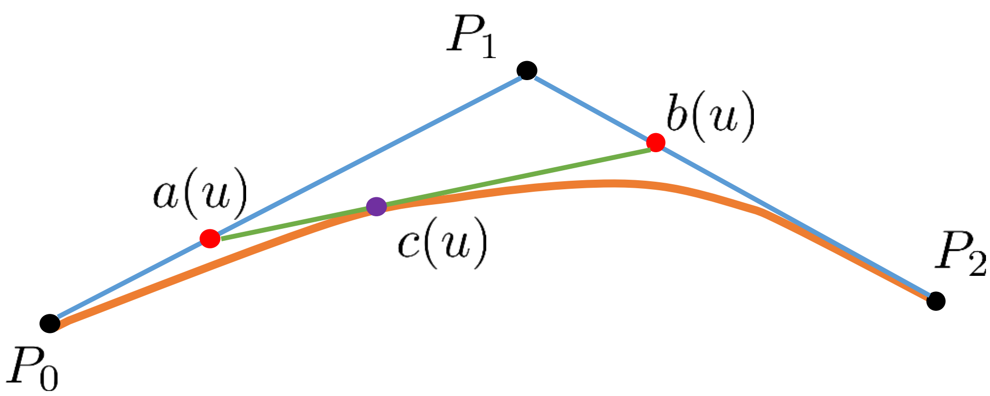



$$a(u) = (1-u)P_0 + uP_1$$

$$b(u) = (1-u)P_1 + uP_2$$

# 二次Bézier曲线

- 三个控制顶点



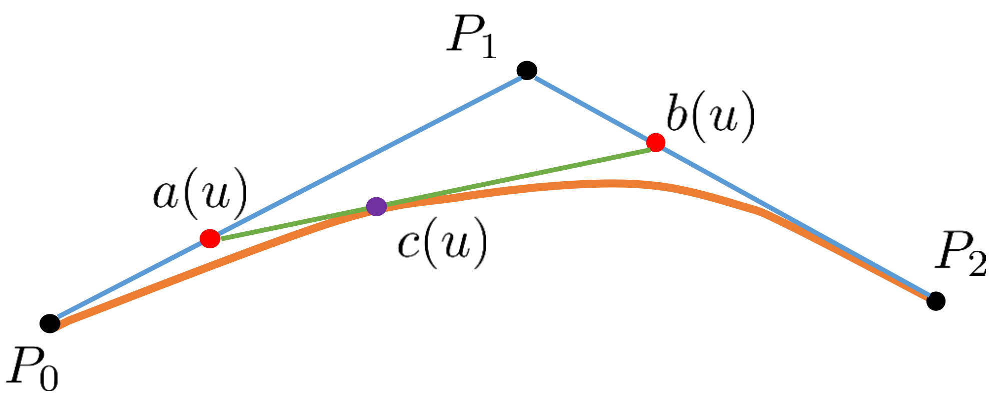

$$a(u) = (1-u)P_0 + uP_1$$

$$b(u) = (1-u)P_1 + uP_2$$

$$c(u) = (1-u)a(u) + ub(u)$$

# 二次Bézier曲线

- 三个控制顶点




$$a(u) = (1-u)P_0 + uP_1$$

$$b(u) = (1-u)P_1 + uP_2$$

$$0 \le u \le 1$$

$$\begin{aligned} c(u) &= (1-u)a(u) + ub(u) \\ &= (1-u)^2 P_0 + 2(1-u)uP_1 + u^2 P_2 \end{aligned}$$

# 二次Bézier曲线

$$c(u) = (1-u)^2 P_0 + 2(1-u)uP_1 + u^2 P_2$$
$$0 \le u \le 1$$

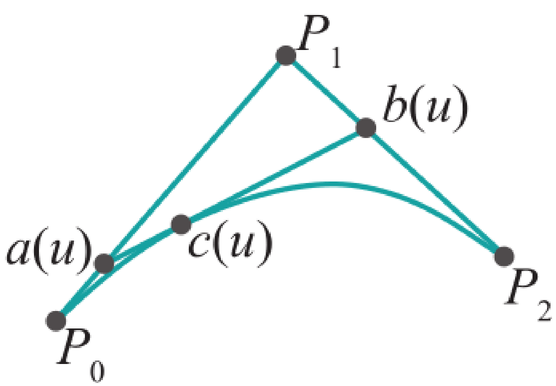
- 几点观察：
- c(u)是u的二次函数;
- 点c(u)是点$P_0$, $P_1$ 和$P_2$的加权和，混合函数分别为$(1-u)^2$, $2(1-u)u$ 和 $u^2$;
- 上述混合函数为Bernstein二次多项式，记为$B_{0,2}(u)$, $B_{1,2}(u)$ 和$B_{2,2}(u)$;
- $B_{0,2}(u)$, $B_{1,2}(u)$ 和$B_{2,2}(u)$ 取值都在0和1之间，且满足：

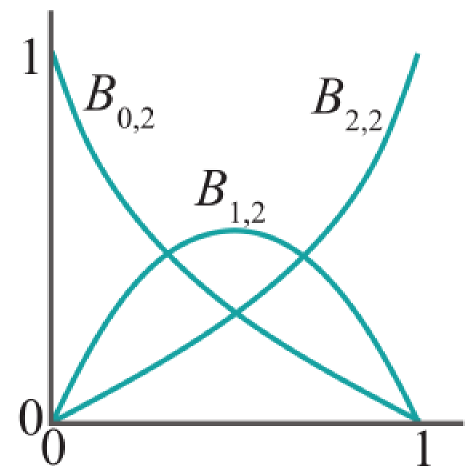
$$B_{0,2}(u) + B_{1,2}(u) + B_{2,2}(u) = 1, \ \ \forall u \in [0,1]$$

- $P_0$ 对应的u=0, $P_2$ 对应的u=1，称曲线在点 $P_0$和$P_2$处进行了插值.

# 二次Bézier曲线

- 二次Berstein多项式

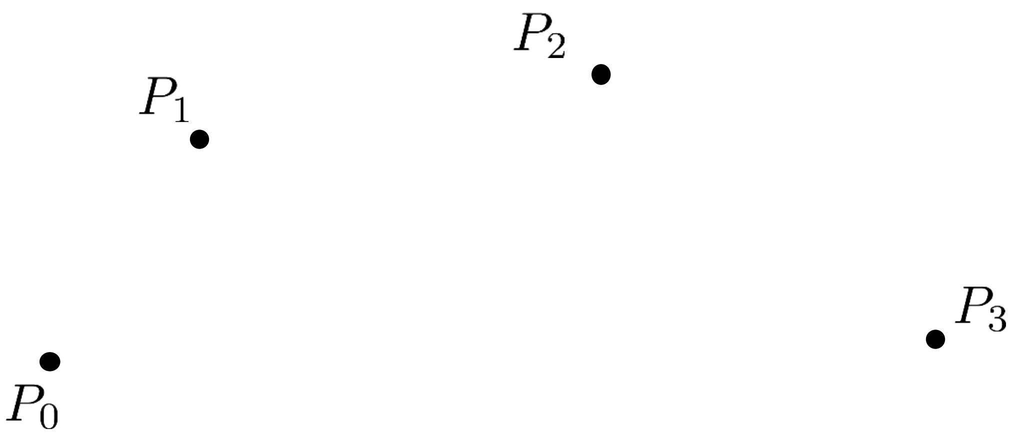



$$c(u) = B_{0,2}(u)P_0 + B_{1,2}(u)P_1 + B_{2,2}(u)P_2$$

# 三次Bézier曲线

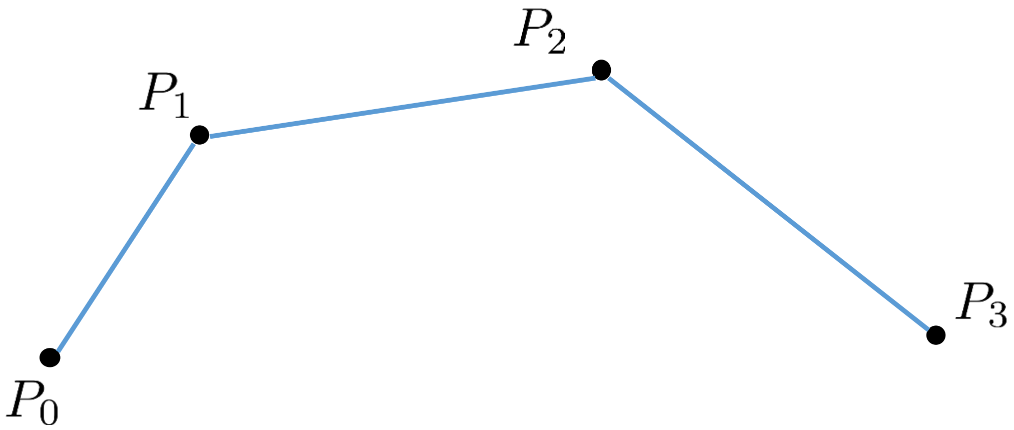
- 四个控制顶点




*如何基于**de Casteljau**算法生成对应的Bézier曲线？*

# 三次Bézier曲线

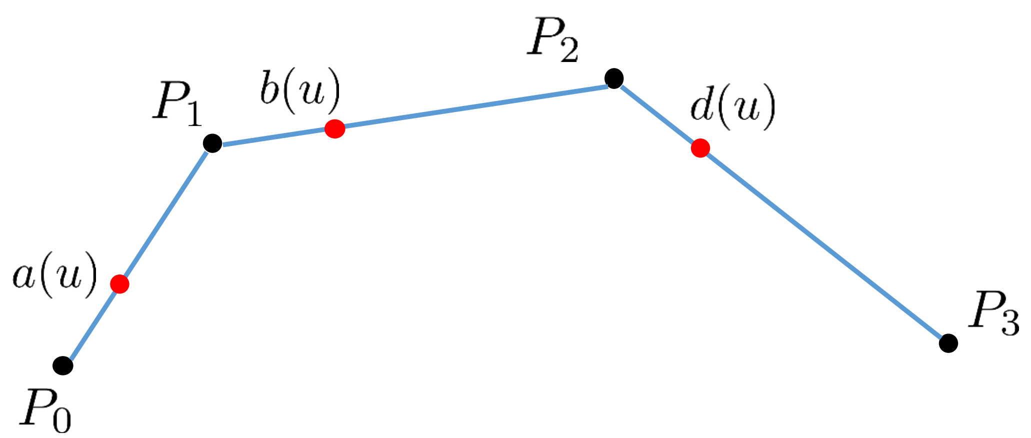
- 四个控制顶点




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

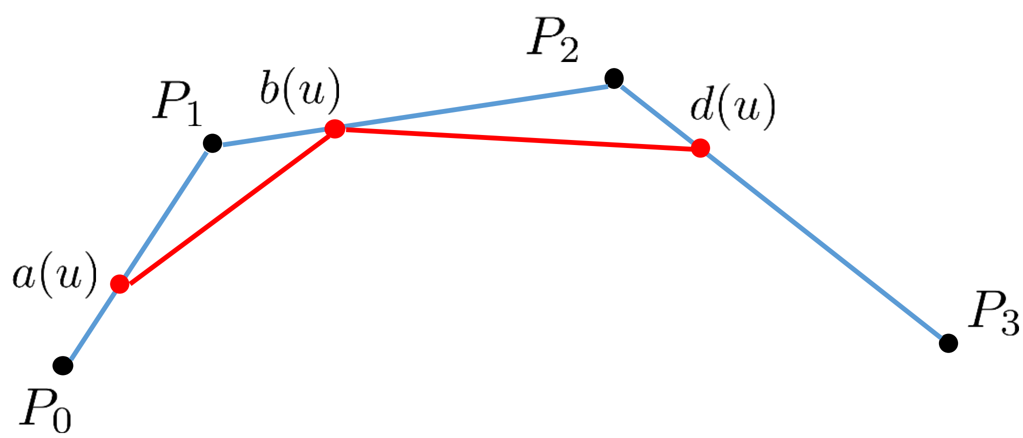
- 四个控制顶点




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

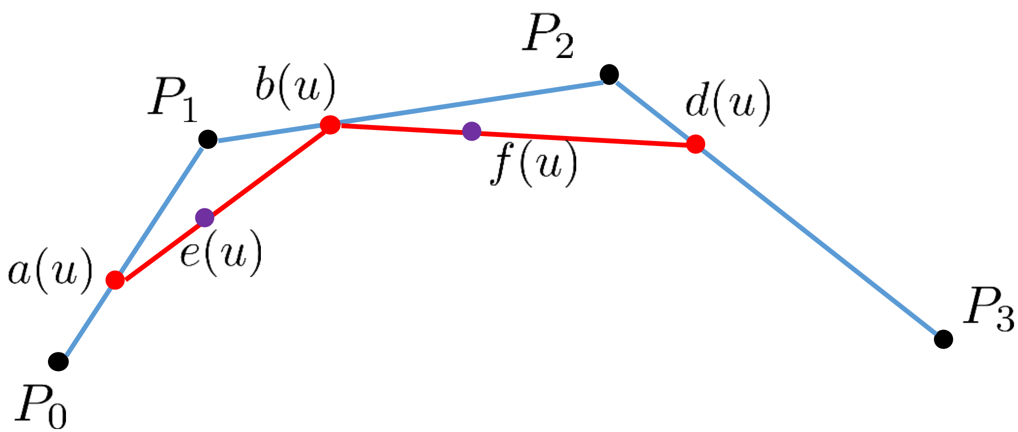
- 四个控制顶点




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

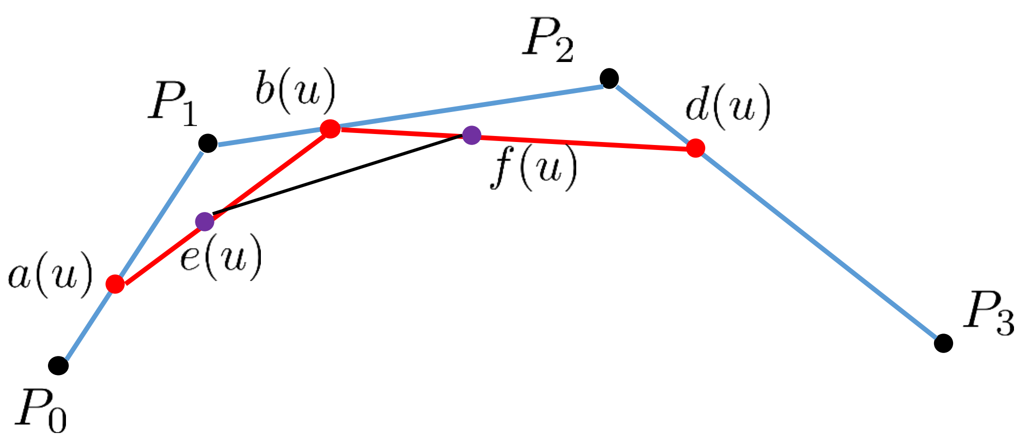
- 四个控制顶点




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

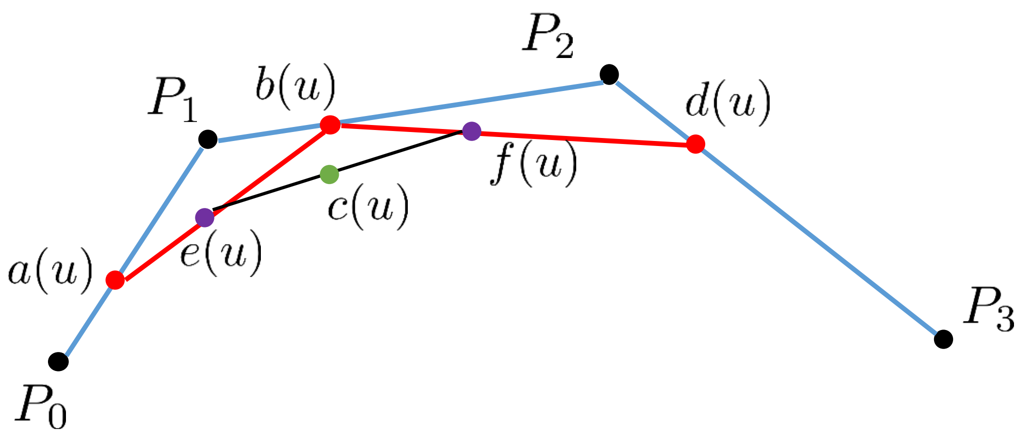
- 四个控制顶点




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

- 四个控制顶点

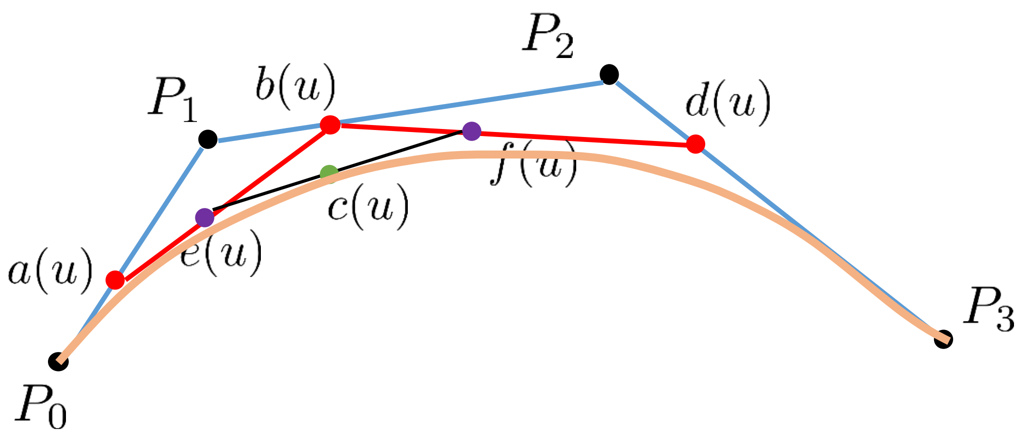



*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

- 四个控制顶点

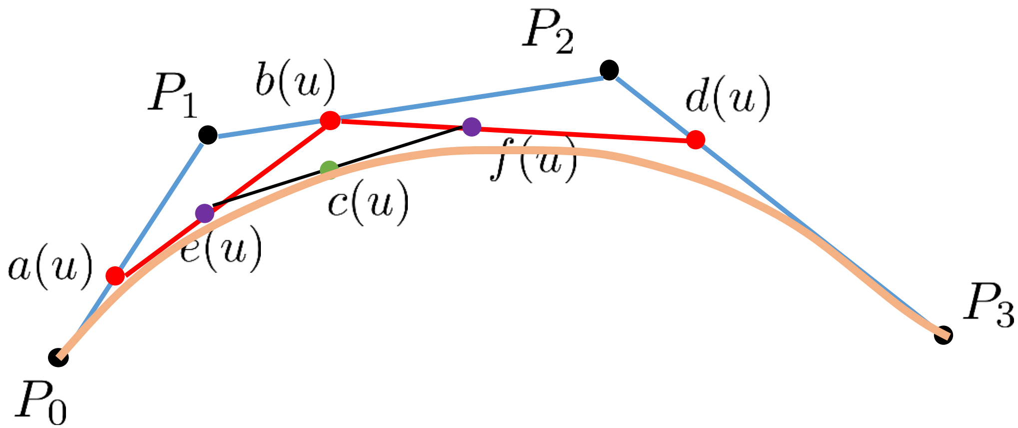
OpenGL Demo演示




*应用**de Casteljau**算法生成三次Bézier曲线*

# 三次Bézier曲线

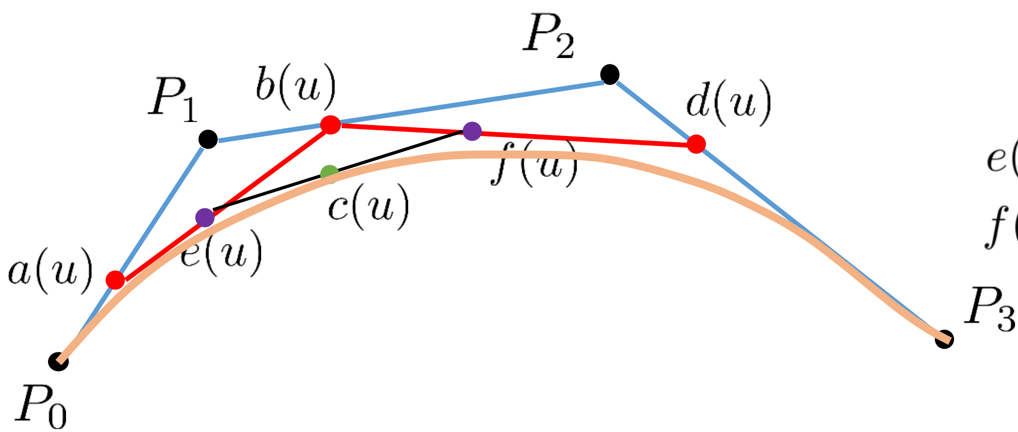
- 四个控制顶点




$$a(u) = (1-u)P_0 + uP_1$$
$$b(u) = (1-u)P_1 + uP_2$$
$$d(u) = (1-u)P_2 + uP_3$$

# 三次Bézier曲线

- 四个控制顶点




$$a(u) = (1-u)P_0 + uP_1$$
$$b(u) = (1-u)P_1 + uP_2$$
$$d(u) = (1-u)P_2 + uP_3$$
$$e(u) = (1-u)a(u) + ub(u)$$
$$f(u) = (1-u)b(u) + ud(u)$$

# 三次Bézier曲线

- 四个控制顶点

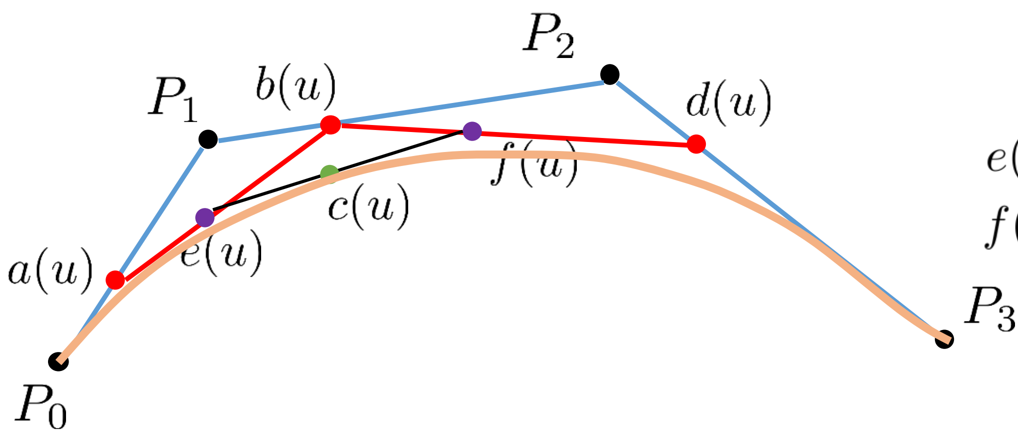
$$a(u) = (1-u)P_0 + uP_1$$

$$b(u) = (1-u)P_1 + uP_2$$

$$d(u) = (1-u)P_2 + uP_3$$

$$e(u) = (1-u)a(u) + ub(u)$$

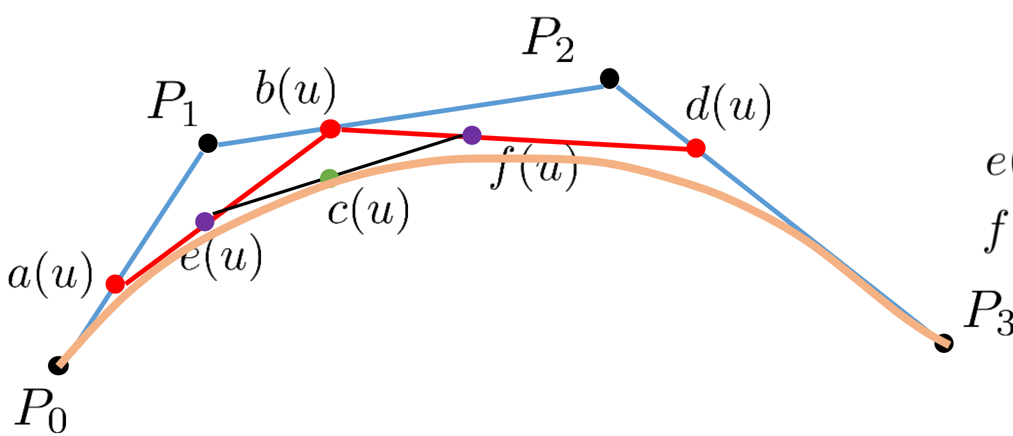
$$f(u) = (1-u)b(u) + ud(u)$$




$$c(u) = (1-u)e(u) + uf(u)$$

# 三次Bézier曲线

- 四个控制顶点

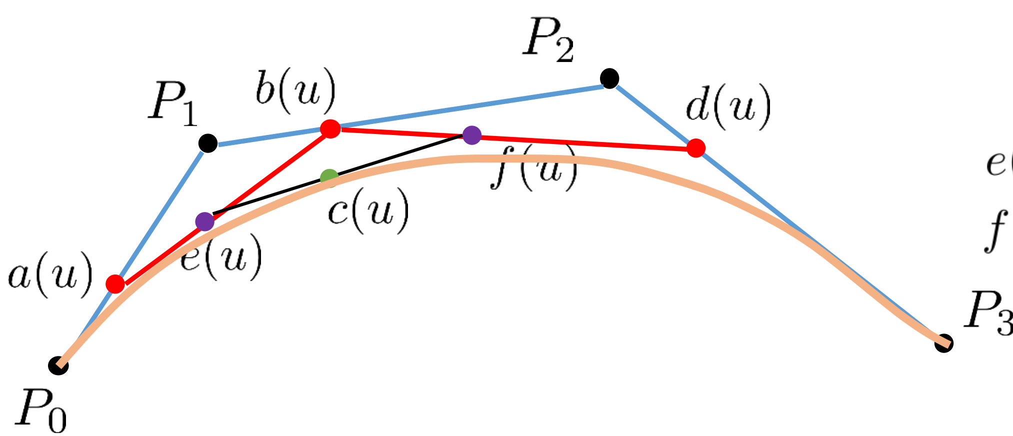



$$a(u) = (1-u)P_0 + uP_1$$
$$b(u) = (1-u)P_1 + uP_2$$
$$d(u) = (1-u)P_2 + uP_3$$
$$e(u) = (1-u)a(u) + ub(u)$$
$$f(u) = (1-u)b(u) + ud(u)$$

$$\begin{aligned} c(u) &= (1-u)e(u) + uf(u) \\ &= (1-u)^2 a(u) + 2(1-u)ub(u) + u^2 d(u) \end{aligned}$$

# 三次Bézier曲线

- 四个控制顶点




$$a(u) = (1-u)P_0 + uP_1$$
$$b(u) = (1-u)P_1 + uP_2$$
$$d(u) = (1-u)P_2 + uP_3$$
$$e(u) = (1-u)a(u) + ub(u)$$
$$f(u) = (1-u)b(u) + ud(u)$$

$$\begin{aligned} c(u) &= (1-u)e(u) + uf(u) \\ &= (1-u)^2 a(u) + 2(1-u)ub(u) + u^2 d(u) \\ &= (1-u)^3 P_0 + 3(1-u)^2 uP_1 + 3(1-u)u^2 P_2 + u^3 P_3 \end{aligned}$$

$$0 \le u \le 1$$

# 三次Bézier曲线

$$c(u) = (1-u)^3 P_0 + 3(1-u)^2 u P_1 + 3(1-u) u^2 P_2 + u^3 P_3$$

$$0 \le u \le 1$$

- 几点观察：
- c(u)是u的三次函数;
- 点c(u)是点$P_0$, $P_1$ ,$P_2$ 和$P_3$的加权和，对应的混合函数为:

$$(1-u)^3,\ 3(1-u)^2 u,\ 3(1-u)u^2,\ u^3$$

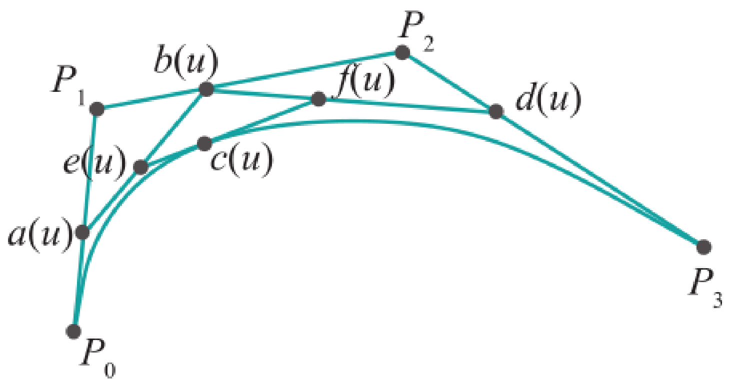
- 上述混合函数为Bernstein三次多项式,记为$B_{0,3}(u)$ ,$B_{1,3}(u)$ ,$B_{2,3}(u)$ 及$B_{3,3}(u)$
- $B_{0,3}(u)$, $B_{1,3}(u)$ ,$B_{2,3}(u)$ 和$B_{3,3}(u)$取值都在0和1之间，且满足：

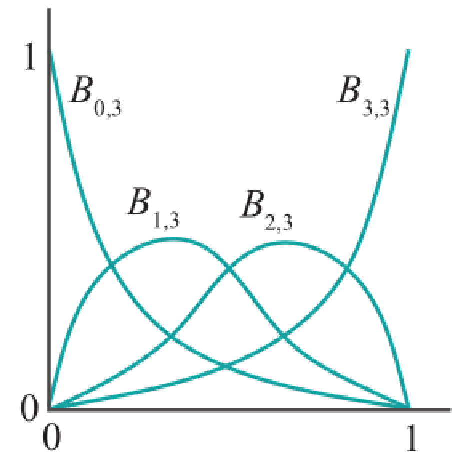
$$B_{0,3}(u) + B_{1,3}(u) + B_{2,3}(u) + B_{3,3}(u) = 1,\ \ \forall u \in [0,1]$$

- $P_0$ 对应的u=0, $P_3$ 对应的u=1，称曲线在点 $P_0$和$P_3$处进行了插值.

# 三次Bézier曲线

- 三次Bernstein多项式

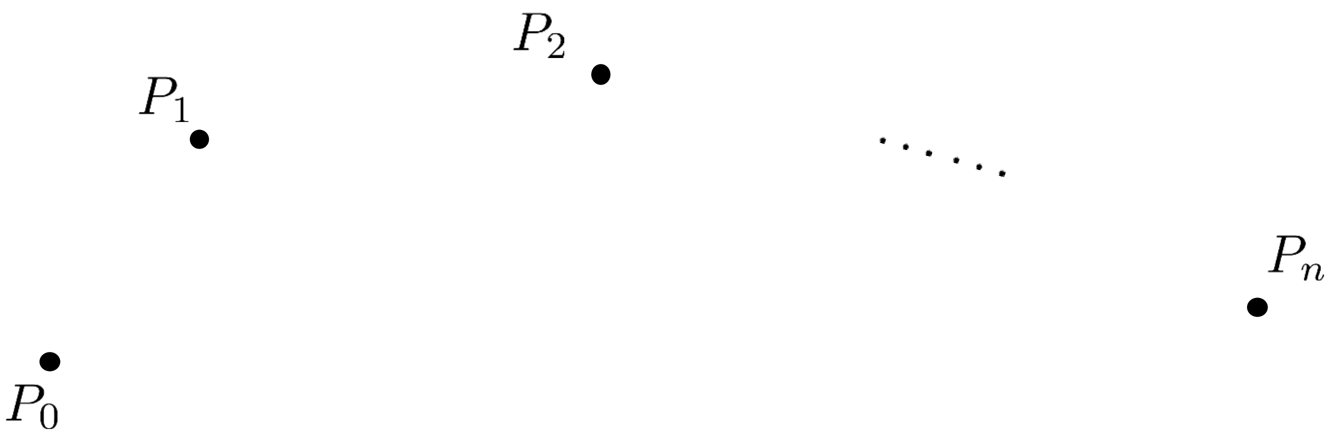



$$c(u) = B_{0,3}P_0 + B_{1,3}P_1 + B_{2,3}P_2 + B_{3,3}P_3$$
$$0 \leq u \leq 1$$

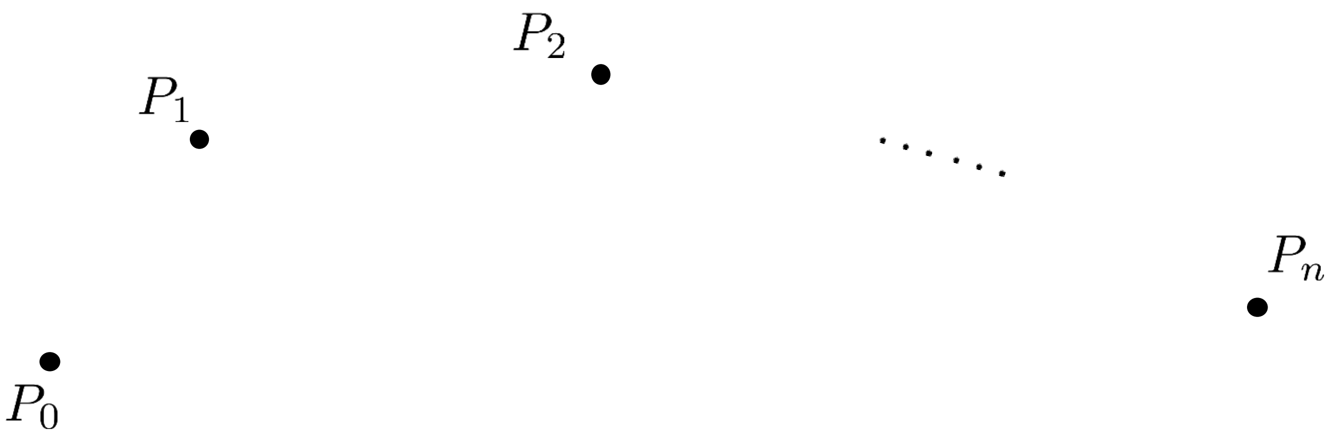
# 广义Bézier曲线

- n个控制顶点



# 广义Bézier曲线

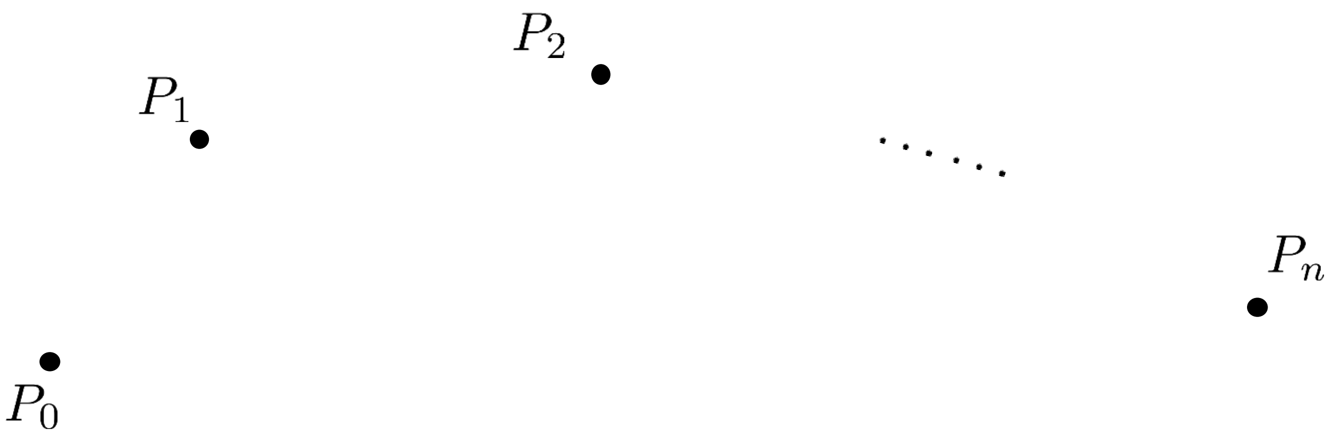
- n个控制顶点




$$c(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i,\ \ 0 \le u \le 1$$

# 广义Bézier曲线

- n个控制顶点



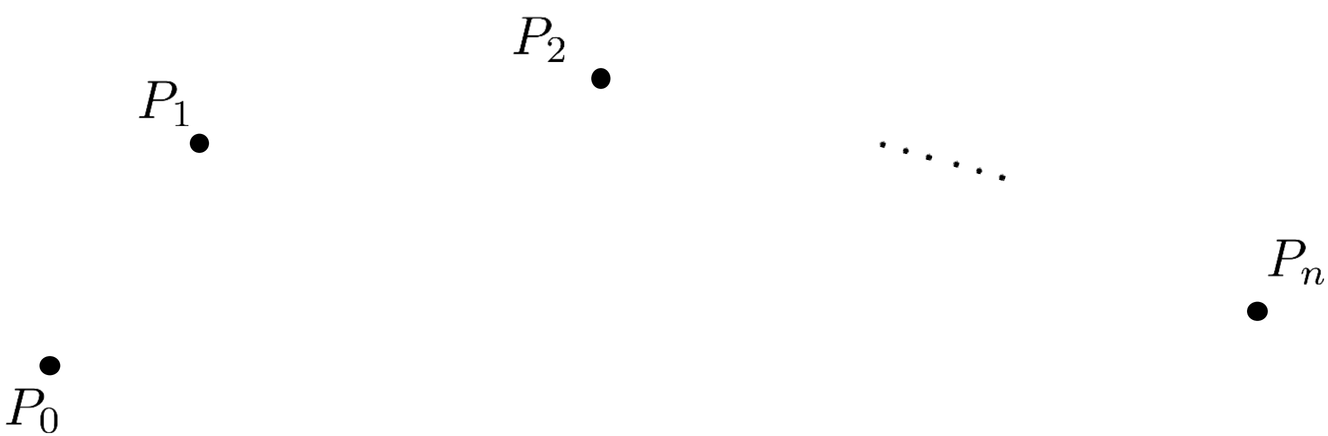

$$c(u) = \sum_{i=0}^{n} \overset{?}{\boxed{B_{i,n}(u)}} P_i, \ 0 \le u \le 1$$

# 广义Bézier曲线

- n个控制顶点

二项式系数

$$\boxed{B_{i,n}(u)=\begin{pmatrix} n \\ i \end{pmatrix}(1-u)^{n-i}u^{i}}$$




$$c(u)=\sum_{i=0}^{n} B_{i,n}(u)P_i,\ \ 0\le u\le 1$$

# 广义Bézier曲线

$$c(u) = \sum_{i=0}^{n} B_{i,n}(u)P_i,\ 0 \le u \le 1$$

- de Casteljau递归算法
- 基于n+1个控制顶点的Bézier曲线可以由如下递归公式生成：

$$c(u) = (1-u)c_0(u) + uc_1(u),\ 0 \le u \le 1$$

$$c_0(u) = \sum_{i=0}^{n-1} B_{i,n}P_i$$

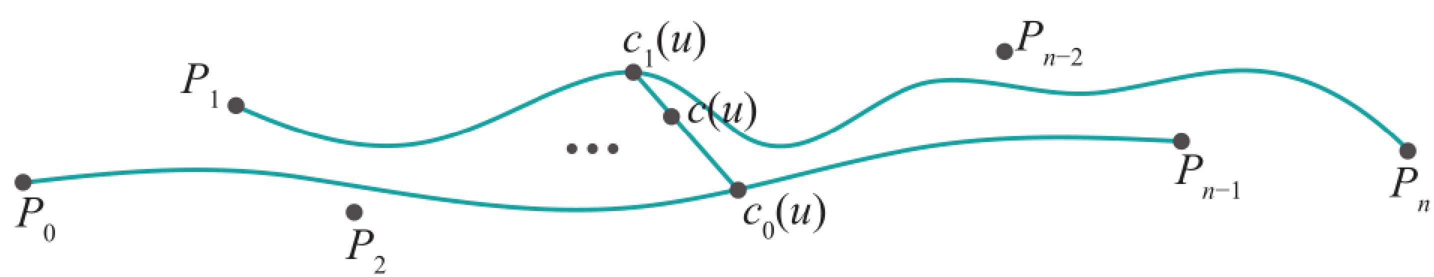
$$c_1(u) = \sum_{i=1}^{n} B_{i,n}P_i$$

# 广义Bézier曲线

$$c(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i,\ 0 \le u \le 1$$

- de Casteljau递归算法
- 基于n+1个控制顶点的Bézier曲线，可以由如下递归公式定义：

$$c(u) = (1-u)c_0(u) + uc_1(u),\ 0 \le u \le 1$$



# 广义Bézier曲线

- 起始点以及终点处切线特性：
- 曲线c在$P_0$点处的切线方向由$P_0$指向$P_1$
- 曲线c在$P_n$点处的切线方向由$P_{n-1}$指向$P_n$

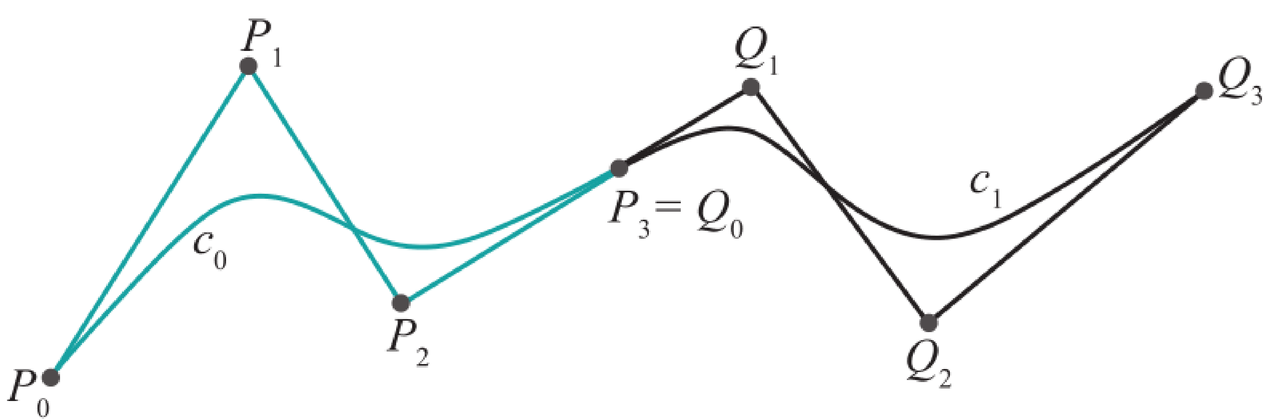
# 广义Bézier曲线

- 起始点以及终点处切线特性：
- 曲线c在$P_0$点处的切线方向由$P_0$指向$P_1$
- 曲线c在$P_n$点处的切线方向由$P_{n-1}$指向$P_n$

可以利用上述切线特性拼接两条Bézier曲线

# 广义Bézier曲线

- 起始点以及终点处切线特性：
- 曲线c在$P_0$点处的切线方向由$P_0$指向$P_1$
- 曲线c在$P_n$点处的切线方向由$P_{n-1}$指向$P_n$



# OpenGL实例

```
static float controlPoints[6][3] =
{
    { -4.0, -2.0, 0.0}, { -3.0, 2.0, -5.0}, { -1.0, -1.0, 2.0},
    {0.0, 2.0, -2.0}, {3.0, -3.0, 1.0}, { 4.0, 0.0, -1.0}
};
```


```
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

```
void display(void)
{
   int i;

   glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

   // Draw the Bezier curve by approximating with a line strip.
   glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINE_STRIP);
      for (i = 0; i <= 50; i++) glEvalCoord1f( (float)i/50.0 );
   glEnd();

   // Draw the control points as dots.
   /* ... */

   glFlush();
}
```

# OpenGL实例

```c
static float controlPoints[6][3] =
{
    { -4.0, -2.0, 0.0}, { -3.0, 2.0, -5.0}, { -1.0, -1.0, 2.0},
    {0.0, 2.0, -2.0}, {3.0, -3.0, 1.0}, { 4.0, 0.0, -1.0}
};
```

```c
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```


```c
void display(void)
{
   int i;

   glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

   // Draw the Bezier curve by approximating with a line strip.
   glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINE_STRIP);
      for (i = 0; i <= 50; i++) glEvalCoord1f( (float)i/50.0 );
   glEnd();

   // Draw the control points as dots.
   /* ... */

   glFlush();
}
```

# OpenGL实例

```
void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);

    // Specify and enable the Bezier curve.
    glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
    glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

# OpenGL实例

```
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

GL_MAP1_VERTEX_3:
生成三维顶点

# OpenGL实例

```
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

GL_MAP1_VERTEX_3:
生成三维顶点

t1, t2: u的取值范围

# OpenGL实例

```c
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

GL_MAP1_VERTEX_3:
生成三维顶点

stride:控制顶点数组中
相邻两个顶点之间的浮
点值数目

t1, t2: u的取值范围

# OpenGL实例

```c
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

GL_MAP1_VERTEX_3:
生成三维顶点

t1, t2: u的取值范围

stride:控制顶点数组中
相邻两个顶点之间的浮
点值数目

```c
static float controlPoints[6][3] =
{
   { -4.0, -2.0, 0.0}, { -3.0, 2.0, -5.0}, { -1.0, -1.0, 2.0},
   {0.0, 2.0, -2.0}, {3.0, -3.0, 1.0}, { 4.0, 0.0, -1.0}
};
```

| | | | | | |
|---|---|---|---|---|---|
| -4.0 | -2.0 | -0.0 | -3.0 | 2.0 | -5.0 |

# OpenGL实例

```
void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```

OpenGL一维Bézier求值器：

**glMap1f(target, t1, t2, stride, order, *controlPoints)**

GL_MAP1_VERTEX_3: 生成三维顶点

stride:控制顶点数组中相邻两个顶点之间的浮点数数目

t1, t2: u的取值范围

order: 控制顶点数目

# OpenGL实例

```
void display(void)
{
    int i;

    glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

    // Draw the Bezier curve by approximating with a line strip.
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINE_STRIP);
       for (i = 0; i <= 50; i++) glEvalCoord1f( (float)i/50.0 );
    glEnd();

    // Draw the control points as dots.
   /* ... */

    glFlush();
}
```

**glEvalCoord1f(u): 计算Bézier曲线上指定参数u所对应点的坐标**

# OpenGL实例

```
static float controlPoints[6][3] =
{
    { -4.0, -2.0, 0.0}, { -3.0, 2.0, -5.0}, { -1.0, -1.0, 2.0},
    {0.0, 2.0, -2.0}, {3.0, -3.0, 1.0}, { 4.0, 0.0, -1.0}
};

void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```


```
void display(void)
{
   int i;

   glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

   // Draw the Bezier curve by approximating with a line strip.
   glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINE_STRIP);
      for (i = 0; i <= 50; i++) glEvalCoord1f( (float)i/50.0 );
   glEnd();

   // Draw the control points as dots.
   /* ... */

   glFlush();
}
```

# OpenGL实例

```
static float controlPoints[6][3] =
{
    { -4.0, -2.0, 0.0}, { -3.0, 2.0, -5.0}, { -1.0, -1.0, 2.0},
    {0.0, 2.0, -2.0}, {3.0, -3.0, 1.0}, { 4.0, 0.0, -1.0}
};

void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);

   // Specify and enable the Bezier curve.
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, controlPoints[0]);
   glEnable(GL_MAP1_VERTEX_3);
}
```


```
void display(void)
{
   int i;

   glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

   // Draw the Bezier curve by approximating with a line strip.
   glColor3f(0.0, 0.0, 0.0);

   glMapGrid1f(50, 0.0, 1.0);
   glEvalMesh1(GL_LINE, 0, 50);

   // Draw the control points as dots.
   /* ... */

   glFlush();
}
```

# OpenGL实例

```
void display(void)
{
    int i;

    glClear(GL_COLOR_BUFFER_BIT);
   /* ... */

    // Draw the Bezier curve by approximating with a line strip.
     glMapGrid1f(50, 0.0, 1.0);
     glEvalMesh1(GL_LINE, 0, 50);

    // Draw the control points as dots.
   /* ... */

    glFlush();
}
```

**glMapGrid1f(n, t1, t2): 在参数区间[t1,t2]内均匀采样n个点**

**glEvalMesh1(mode, p1, p2): 结合上述采样点，按mode以p1为初始采样点开始绘制直到p2结束**

# 作业5-1

- 已知4个控制点的坐标如下：

$$P_0 = [-2, 2]^T \qquad P_1 = [0, -3]^T$$

$$P_2 = [3,\ 4]^T \qquad P_3 = [7,\ 0]^T$$

- 1. 请给出对应的三次Bézier曲线的方程；
- 2. 计算曲线上u=0.0, 0.5以及1.0处点的坐标；
- 3. 计算曲线上点 $P_0$ 和 $P_3$ 处切线对应的方向向量。

# 作业5-2

- 已知二次Bézier曲线：

$$c(u) = (1-u)^2 P_0 + 2(1-u)uP_1 + u^2 P_2$$

- 可以表示为如下等价的矩阵形式：

$$c(u) = [P_0 \ P_1 \ P_2] M [u^2 \ u \ 1]^T$$

- 请计算矩阵M。
- 提示：M是一个3X3矩阵。