软件设计文档

课程：图形学

任课老师：孙正兴

姓名：金鑫

学号：121220307

CONTENT

1.1 编写目的

对软件进行模块级别的详细设计说明，以便编写代码人员参考，也方便维护人员在软件维护过程中的维护工作。

本文档的阅读对象为软件的开发和使用人员。

1.2 背景

项目名称：Curve

项目的提出：孙正兴老师

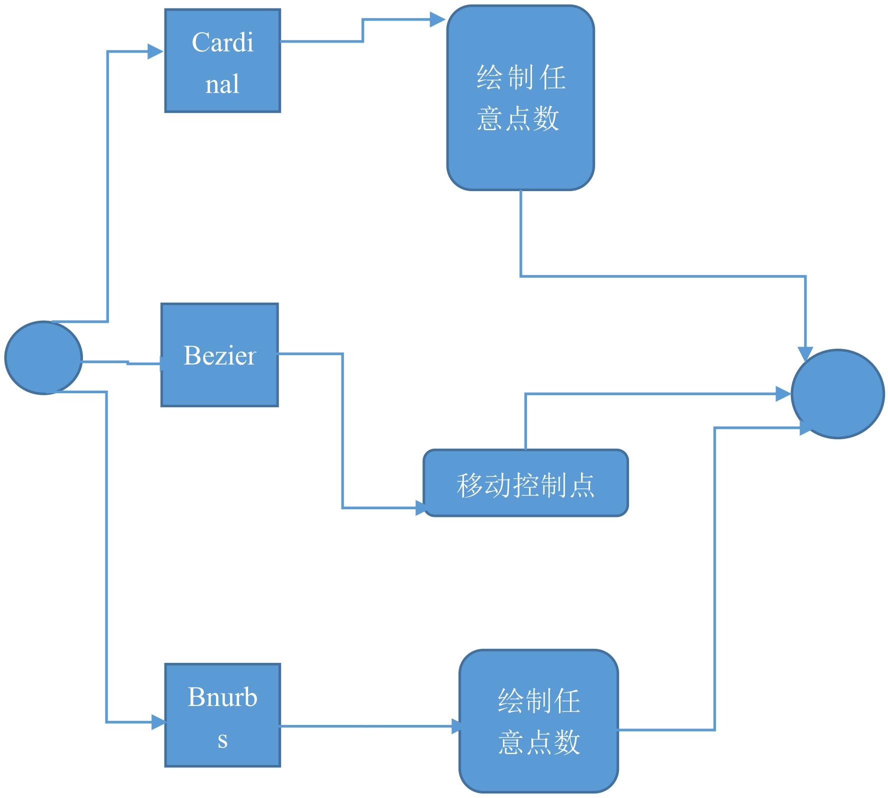
项目的开发者：金鑫

1.3 定义

A. 系统：Curve

B. 用户：使用该系统的用户

2. 程序系统的结构

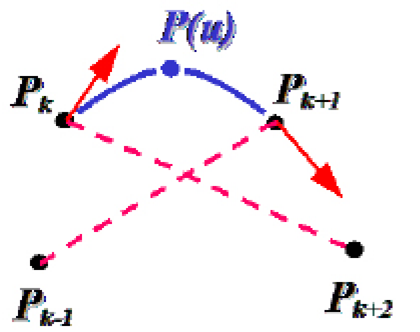


3. 算法描述

3.1 Cardinal 样条曲线

- *Cardinal*样条也是插值分段三次曲线，且每曲线段终点处均指定切线。
- 与*Hermite*样条的区别是：
  - 不一定要给出终点的切线值。
  - 一个控制点处斜率值由两个相邻控制点坐标来计算。
- *Cardinal*样条由四个连续控制点给出(如图)：
  - 中间两个控制点是曲线段端点，
  - 另二个点用来计算终点斜率。

在控制点P_k和P_{k+1}间*Cardinal*样条段的参数向量函数$P(u)$，其端点处的切向量正比于由相邻控制点所形成的弦。



- 用类似Hermite样条的方法，可将边界条件转换成矩阵形式：

$$P(u)=[\,u^3 \quad u^2 \quad u \quad 1\,]\cdot M_C \cdot \begin{bmatrix} P_{k-1} \\ P_k \\ P_{k+1} \\ P_{k+2} \end{bmatrix}$$

- M_C是*Cardinal*矩阵，表示为：

$$M_C = \begin{bmatrix} -t & 2-t & t-2 & t \\ 2t & t-3 & 3-2t & -t \\ -t & 0 & t & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- 将矩阵方程展开成多项式形式，可得到混合函数表达式：

$$P(u)=P_{k-1}(-tu^3+2tu^2-tu)+P_k[(2-t)u^3+(t-3)u^2+1]$$
$$+P_{k+1}[(t-2)u^3-(t-2)u^2+tu]+P_{k+2}(tu^3-tu^2)$$

$$P(u)=P_{k-1}CAR_0(u)+P_kCAR_1(u)+P_{k+1}CAR_2(u)+P_{k+1}CAR_3(u)$$

- 这里多项式$CAR_k(u)$(k=0,1,2,3)称为*Cardinal*混合函数，它们混合了边界约束值(终点坐标和斜率)来得到曲线上每个坐标点位置。

3.2 Bezier 曲线

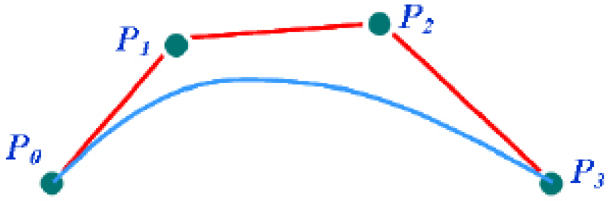
- Bézier曲线可由给定边界条件或特征矩阵决定：$P(u)=G_{BEZ}\cdot M_{BEZ}\cdot U$
- 最方便的是混合函数形式：
  - 给定$n+1$个控制点：$P_k=(x_k,y_k,z_k)$，$(k=0,1,2,\ldots,n)$，
  - 这些点混合产生位置向量$P(u)$，用来描述P_0和P_n间的逼近Bézier多项式函数的路径(Bézier曲线)。

$$P(u)=\sum_{k=0}^{n} P_k BEZ_{k,n}(u),\quad 0\le u\le 1$$

    - 混合函数$BEZ_{k,n}(u)$是Bernstein多项式。
  - 利用Bernstein基函数的降(升)阶公式，可使用递归计算得出Bézier曲线上点的坐标位置。
    - 用递归计算定义Bézier混合函数：

$$BEZ_{k,n}(u)=(1-u)BEZ_{k,n-1}(u)+uBEZ_{k-1,n-1}(u)。$$

    - 其中：$BEZ_{k,k}(u)=u^k$, $BEZ_{0,k}(u)=(1-u)^k$。
- Bézier曲线段可拟合任何数目的控制点：
  - Bézier曲线段逼近这些控制点→控制多边形大致勾画Bézier曲线的形状。



  - 这些控制点的相关位置决定了Bézier多项式的次数。
    - 一条n次Bézier曲线被表示成它的$n+1$个控制顶点的加权和，权是Bernstein基函数。
    - Bézier多项式次数要比控制点个数小1：
      - 三点生成抛物线；四点生成三次曲线；……。
  - 对某些控制点布局，得到退化Bézier多项式。
    - 三个共线控制点生成了一个直线段的Bézier曲线，
    - 具有相同坐标控制点生成Bézier“曲线”是一个点。

3.2 B 样条曲线

- B样条基函数：给定参数u轴上的节点分割

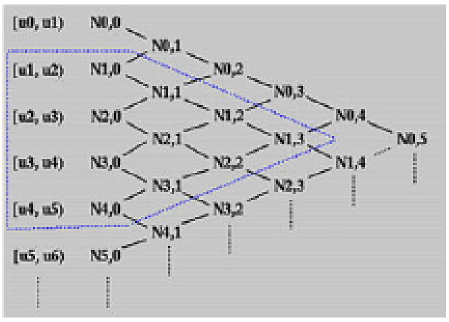
  $U_{n,k+1}=\{u_i\}$ (i=0,1,2,…,n+k+1)，

- 由下列递推关系所确定的$B_{i,k+1}(u)$为$U_{n,k+1}$上的k+1阶(或k次)B样条基函数：

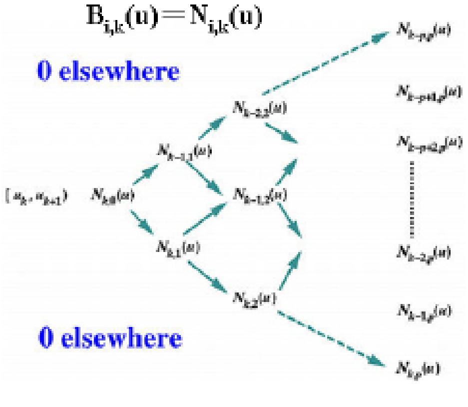
  - deBoox-Cox递推公式：(i=0,1,2,…,n)

$$B_{i,k+1}(u)=\frac{u-u_i}{u_{i+k}-u_i}B_{i,k}(u)+\frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}}B_{i+1,k}(u)$$

  - 递推式中，若遇到0/0则取值为0。
    - $u\in[u_i,u_{i+1})$时，$B_{i,1}(u)=1$；
    - $u\in$其它时，$B_{i,1}(u)=0$。
  - u_i：节点，$U_{n,k+1}$：节点向量。
  - r重节点：若$u_{j-1}<u_j=u_{j+1}=\ldots=u_{j+r-1}<u_{j+r}$，
    - 称：从u_j到u_{j+r-1}的节点为r重节点。



$B_{i,k}(u)=N_{i,k}(u)$



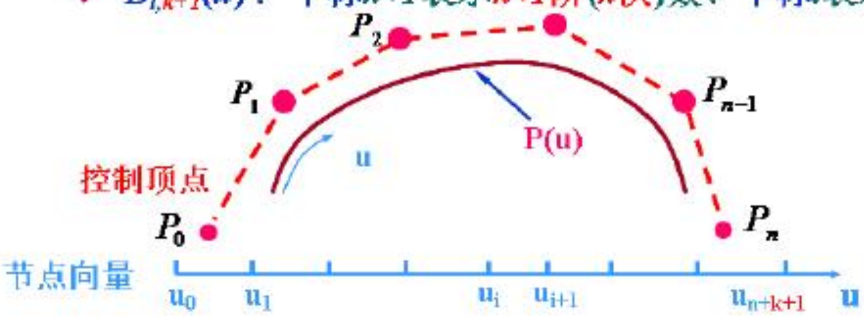
❖ 给定：

- $n+1$个控制顶点$\{P_i\}(i=0,1,\ldots,n)$，$P_0P_1\ldots P_n$为控制多边形
- $n+k+2$个参数节点向量：$U_{n,k}=\{u_i|i=0,1,\ldots,n+k+1,\ u_i\leq u_{i+1}\}$。

❖ 称如下形式的参数曲线$P(u)$为$k+1$阶(k次)B样条曲线：

$$P(u)=\sum_{i=0}^{n}P_iB_{i,k+1}(u),u\in[u_k,u_{n+1}]$$

- 其中：$B_{i,k+1}(u)$为$k+1$阶(k次)B样条基函数。
  - ❖ $B_{i,k+1}(u)$：下标$k+1$表示$k+1$阶(k次)数、下标i表示序号。



4. 程序实现

4.1 Cardinal 曲线

```
	public Point2D cubicCardical(double u, Point2D p1, Point2D p2, Point2D p3, Point2D p4){
		double x, y;
		x = p1.getX() * (-0.5 * u * u * u + u * u - 0.5 * u) + p2.getX() * (1.5 * u * u * u - 2.5 * u * u + 1) + p3.getX() * (-1.5 * u * u * u + 2 * u * u + 0.5 * u)+ p4.getX() * (0.5 * u * u * u - 0.5 * u * u);
		y = p1.getY() * (-0.5 * u * u * u + u * u - 0.5 * u) + p2.getY() * (1.5 * u * u * u - 2.5 * u * u + 1) + p3.getY() * (-1.5 * u * u * u + 2 * u * u + 0.5 * u)+ p4.getY() * (0.5 * u * u * u - 0.5 * u * u);
		Point2D p = new Point2D.Double(x,y);
		return p;
	}
	public void drawCarnical(Graphics g, ArrayList<Point2D> p){
		g.setColor(Color.red);
		for(int i = 0; i < p.size() - 3; i ++){
			for(double t = 0; t < 1; t+=0.002)
			{
				Point2D p1= cubicCardical(t, p.get(i), p.get(i + 1), p.get(i + 2), p.get(i + 3));
				Point2D p2 = cubicCardical(t+0.001, p.get(i), p.get(i + 1), p.get(i + 2), p.get(i + 3));

	g.drawLine((int)p1.getX(),(int)p1.getY(),(int)p2.getX(),(int)p2.getY());
			}
		}
	}
	public void paintComponent(Graphics g)
	{
		Graphics2D g2 = (Graphics2D) g;
		for(int i = 0; i < points.size(); i++)
		{
			double x = points.get(i).getX() - SIZE/2;
			double y = points.get(i).getY() - SIZE/2;
			g.setColor(Color.cyan);
			g2.fill(new Rectangle2D.Double(x, y, SIZE, SIZE));
			g.setColor(Color.MAGENTA);
```

```
                        g.drawString(new
String("("+points.get(i).getX()+","+(points.get(i).getY()-30)+")"),
(int)points.get(i).getX(), (int)points.get(i).getY());
                }

                g.setColor(Color.BLUE);
                for(int i = 0; i < points.size() - 1; i++)
                {
                        g.drawLine((int)points.get(i).getX(),
(int)points.get(i).getY(), (int)points.get(i + 1).getX(), (int)points.get(i
+ 1).getY());
                }
                if(points.size() < 4) return;
                drawCarnical(g, points);
        }
```

4.2 Bezier 曲线

```
        public Point2D cubicBezier(double t, Point2D[] p)
        {
                Point2D[] temp = new Point2D[p.length];
                for(int k=0; k < p.length; k++)
                        temp[k]=p[k];
                for(int i=0; i< 3; i++)
                {
                        for(int j = 0; j < 4-i-1 ; j++)
                        {
                                double x = (1-t)*temp[j].getX() +
t*temp[j+1].getX();
                                double y = (1-t)*temp[j].getY()+
t*temp[j+1].getY();
                                temp[j] = new Point2D.Double(x,y);
                        }
                }
                return temp[0];
        }

        public void drawBezier(Graphics g, Point2D[] p)
        {
                g.setColor(Color.red);
                for(double t = 0; t < 1; t+=0.002)
                {
                        Point2D p1= cubicBezier(t,p);
                        Point2D p2 = cubicBezier(t+0.001,p);
```

```
	g.drawLine((int)p1.getX(),(int)p1.getY(),(int)p2.getX(),(int)p2.getY
());
		}
		g.setColor(Color.blue);
		g.drawLine((int)points[0].getX(), (int)points[0].getY(),
(int)points[1].getX(), (int)points[1].getY());
		g.drawLine((int)points[1].getX(), (int)points[1].getY(),
(int)points[2].getX(), (int)points[2].getY());
		g.drawLine((int)points[2].getX(), (int)points[2].getY(),
(int)points[3].getX(), (int)points[3].getY());
	}

	public void paintComponent(Graphics g)
	{
		if(points == null) return;
		Graphics2D g2 = (Graphics2D) g;
		for(int i = 0; i < points.length; i++)
		{
			double x = points[i].getX() - SIZE/2;
			double y = points[i].getY() - SIZE/2;
			g.setColor(Color.cyan);
			g2.fill(new Rectangle2D.Double(x, y, SIZE, SIZE));
			g.setColor(Color.MAGENTA);
			g.drawString(new
String("("+points[i].getX()+","+(points[i].getY()-30)+")"),
(int)points[i].getX(), (int)points[i].getY());
		}

		drawBezier(g,points);
	}
```

4.3 B 样条曲线

```
public void Calc(double T[], double t, int j, Point2D V)
	{
		int i , r , temp , temp1;
		Point2D Q[] = new Point2D[MAX];
		double lamta;

		temp = j - degree;
		for(i = 0; i <= degree; i ++)
		{
```

```
				Q[i] = new Point2D.Double(points.get(temp+i).getX(), 
points.get(temp+i).getY());
			}
			for( r = 1 ; r <= degree ; ++ r )
			{
				for( i = j ; i >= temp + r ; -- i )
				{
					lamta = (t-T[i])/(T[i + degree - r + 1] - T[i]);
					temp1 = i - temp;
		
	Q[temp1].setLocation(lamta*((double)Q[temp1].getX())+(1.0-
lamta)*((double)Q[temp1-1].getX()), lamta*((double)Q[temp1].getY())+(1.0-
lamta)*((double)Q[temp1-1].getY()));
				}
			}
			V.setLocation(Q[degree].getX(), Q[degree].getY());
	}
	
	public void drawBnurbs(Graphics g, ArrayList<Point2D> p){
		int i , j;
		double deltat , t;
		Point2D V = new Point2D.Double(0.0,0.0),newV = new 
Point2D.Double(0.0,0.0);
		
		g.setColor(Color.red);
		
		deltat = (T[points.size()]-T[degree])/COUNT;
		t = T[degree];
		j = degree;
		
		
		Calc(T,t,j,V);
		
		for(i = 1; i < COUNT ; ++ i)
		{
			t += deltat;
			while(t > T[j+1])
			{
				j++;
			}
			Calc(T,t,j,newV);
```

```
	g.drawLine((int)V.getX(),(int)V.getY(),(int)newV.getX(),(int)newV.getY());
			V.setLocation(newV.getX(), newV.getY());
		}
	}
	public void paintComponent(Graphics g)
	{
		Graphics2D g2 = (Graphics2D) g;
		for(int i = 0; i < points.size(); i++)
		{
			double x = points.get(i).getX() - SIZE/2;
			double y = points.get(i).getY() - SIZE/2;
			g.setColor(Color.cyan);
			g2.fill(new Rectangle2D.Double(x, y, SIZE, SIZE));
			g.setColor(Color.MAGENTA);
			g.drawString(new String("("+points.get(i).getX()+","+(points.get(i).getY()-30)+")"), (int)points.get(i).getX(), (int)points.get(i).getY());
		}

		g.setColor(Color.BLUE);
		for(int i = 0; i < points.size() - 1; i++)
		{
			g.drawLine((int)points.get(i).getX(), (int)points.get(i).getY(), (int)points.get(i + 1).getX(), (int)points.get(i + 1).getY());
		}
		if(points.size() < 3) return;
		drawBnurbs(g, points);
	}
```

5. 操作介绍

5.1 绘制 Cardinal 样条曲线

Step1：选择 Cardinal 选项
Step2：在屏幕上点击你想要点击的任意点的数目，边点边显示曲线

5.2 绘制 Bezier 样条曲线

Step1：选择 Bezier 选项
Step2：选择四个控制点随意移动，观察 Bezier 曲线的任意局部性

5.3 绘制 Bnurbs 样条曲线

Step1：选择 Bnurbs 选项
Step2：在屏幕上点击你想要点击的任意点的数目，边点边显示曲线

6. 类介绍

- 类表

所属包	名称	标识符	数据项	操作	层次关系
Graphics	主界面	Home	comboBox Bezier Cardinal bnurbs	drawimage	继承
graphics	Cardinal 面板	CardinalPanel	points	initPoints cubicCardical drawCardinal paintComponent	
graphics	Bezier 面板	BezierPanel	Points	initPoints cubicBezier drawBezier	

				paintCompnent	
graphics	Bnurbs 面板	BnurbsPanel	Points	initPoints Calc drawBnurbs paintComponent	