计算机动画实验指导书

1 实验目的

设计并实现一个包路径控制曲线，关键帧动画系统(线性插值与矢量线性插值), Free Form Deformation(FFD)，或路径曲线弧长参数化，或自选一个题目。通过上述实验内容，了解动画动态控制的基本原理和方法，提高相关动画编程能力。

2 实验需求

2.1 动画路径控制曲线

在动画中可用样条曲线来表示运动路径以及不规则物体的形状。在本实验中选用 Cardinal 样条曲线，掌握它的表示和算法，了解控制参数对曲线形状的影响。

Cardinal 样条曲线矩阵表示:

$$P(u) = U^T MB$$

这里 U 是幂次最高为 3 的插值变量，M 是 Hermite 多项式矩阵，B 是样条曲线参数。下面是其展开表示。

$$\begin{aligned}
\mathbf{p}(u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \tau(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) \\ \tau(\mathbf{p}_{i+2} - \mathbf{p}_i) \end{bmatrix} \\
&= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\tau & 0 & \tau & 0 \\ 0 & -\tau & 0 & \tau \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} \\
&= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \tau \begin{bmatrix} -1 & 2/\tau - 1 & -2/\tau + 1 & 1 \\ 2 & -3/\tau + 1 & 3/\tau - 2 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 1/\tau & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}
\end{aligned}$$

这里 u∈[0,1], Pi-1, Pi, Pi+1,Pi+2 是控制点，τ 控制曲线的弯曲程度。

2.2 关键帧动画系统(线性插值与矢量线性插值)

关键帧动画技术是计算机动画中的一类重要技术。本实验选取线性插值和矢量线性插值作为实验内容，旨在了解关键帧动画系统的结构，变形算法的思想以及不同算法对应的不同性能。

2.3 Free Form Deformation (FFD)

通过本实验了解 FFD 变形思想. 建立物体顶点坐标以及控制点坐标，计算移动控制点坐标后物体顶点的变形坐标. 用 FFD 设计一简单动画。

2.4 样条路径曲线弧长参数化

假设物体的运动轨迹为一空间参数曲线Q(u),物体的速度曲线为一平面参数曲线V(u)=(T(u),s(u))。为了得到动画序列,我们必须对Q(u)等间隔采样,以求得物体在每一帧的位置。通过本实验加深理解弧长参数化的概念，掌握样条路径曲线弧长参数化的思想及其算法。

3 设计指导

3.1 路径控制曲线

3.1.1 任务概述

1。找出 Cardinal 样条曲线的矩阵表示和程序之间的对应关系。

2。给定若干控制点的位置（这些控制点可以大致描述某个运动路径的形状），用上述程序计算出控制点之间的插值点，显示出样条曲线。

3。改变曲线弯曲程度的参数 τ∈[0，1]大小，观察曲线形状的变化。

4。在实验报告中简述 Cardinal 样条曲线，附上图形结果并用文字讨论。

3.1.2 算法流程（参见 3.1.3 代码提示）

1. 输入控制点，每两控制点之间插入的新点数量 grain，控制样条曲线控制点处平滑程度的 tension。
2. 用给定 tension=a1 初始化 Cardinal 矩阵。

   m[0]=-a1; m[1]=2.-a1; m[2]=a1-2.; m[3]=a1; m[4]=2.*a1; m[5]=a1-3.; m[6]=3.-2*a1; m[7]=-a1;
   m[8]=-a1; m[9]=0.; m[10]=a1; m[11]=0.; m[12]=0.; m[13]=1.; m[14]=0.; m[15]=0.;
3. 调用 Cardinal 矩阵对输入数据进行样条曲线插值。

```
Matrix(float a, float b, float c, float d, float alpha)
{
    float p0,p1,p2,p3;
    p0=m[0]*a+m[1]*b+m[2]*c+m[3]*d;
    p1=m[4]*a+m[5]*b+m[6]*c+m[7]*d;
    p2=m[8]*a+m[9]*b+m[10]*c+m[11]*d;
    p3=m[12]*a+m[13]*b+m[14]*c+m[15]*d;
    return(p3+alpha*(p2+alpha*(p1+alpha*p0)));
}
```

4. 将插值结果记录保存。

3.1.3 算法代码提示

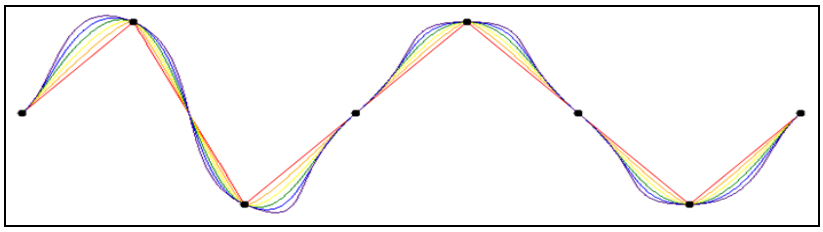
```
// Spline.h : header file
struct CPT { double x; double y;};
#define    CPt CPT

class CSpline : public CEdit
{
// Construction
public:
 CSpline(double x[100], double y[100],    int grain, double tension, int n);

// Attributes
public:

// Operations
public:
```

```
// Overrides
 // ClassWizard generated virtual function overrides
 //{{AFX_VIRTUAL(CSpline)
 //}}AFX_VIRTUAL

// Implementation
public:
 int last;
 CPt Spline[1024];
 void CubicSpline(int n, CPt *knots, int grain, double tension);

 virtual ~CSpline();

 // Generated message map functions
protected:
 //{{AFX_MSG(CSpline)
        // NOTE - the ClassWizard will add and remove member functions here.
 //}}AFX_MSG

 DECLARE_MESSAGE_MAP()
private:
   CPt *knots0;
   int n0;
   double m[16];
   double Matrix(double a, double b, double c, double d, double alpha);
   void GetCardinalMatrix(double a1);
};
//end of head file

CSpline::CSpline(double x[100], double y[100], int grain, double tension, int n)
{
//x[100], y[100]用来存放特征控制点坐标，n 为控制点数量
//grain 控制每两个相邻控制点之间样条插值后的添加点数，一般 grain<20 可以满足大部分应用要求
// tension ∈[0.0，1.]控制样条插值曲线通过控制点处的曲率，实验中应改考察 tension 数值变化对拟合曲线形状带来什么影响；

 int i,np;      n0=n; np=n0;    CPt jd[100]; CPt *knots;

 knots0=knots;
//将控制点坐标赋值到数组 jd 中
 for(i=1;i<=np;i++) {
            jd[i].x = x[i-1];    jd[i].y = y[i-1];
       }

      jd[0].x = x[0];              jd[0].y = y[0];
      jd[np+1].x = x[np-1];    jd[np+1].y = y[np-1];
      //思考：为什么这里这样赋值？
```

```
        np=np+2;    knots=jd;
        CubicSpline(np, knots, grain, tension);
}

CSpline::~CSpline()
{
}
////////////////////////////////////////////////////////////////////////////
// CSpline message handlers
void CSpline::CubicSpline(int n, CPt *knots, int grain, double tension)
{
 CPt *s, *k0, *kml, *k1, *k2;
 int i,j;
 double alpha[50];

 GetCardinalMatrix(tension);

 for(i=0; i<grain; i++) alpha[i] =((double)i)/grain;
 s = Spline;
 kml = knots;
 k0=kml+1; k1=k0+1; k2=k1+1;
 for(i=1; i<n-1; i++) {
         for(j=0; j<grain;j++) {
                 s->x = Matrix(kml->x,k0->x,k1->x,k2->x,alpha[j]);
                 s->y = Matrix(kml->y,k0->y,k1->y,k2->y,alpha[j]);
                 s++;
         }
         k0++; kml++; k1++; k2++;
 }
}

void CSpline::GetCardinalMatrix(double a1)
{
        m[0]=-a1;      m[1]=2.-a1;    m[2]=a1-2.;      m[3]=a1;
       m[4]=2.*a1; m[5]=a1-3.;    m[6]=3.-2*a1; m[7]=-a1;
       m[8]=-a1;      m[9]=0.;         m[10]=a1;          m[11]=0.;
       m[12]=0.;      m[13]=1.;        m[14]=0.;          m[15]=0.;
}

double CSpline::Matrix(double a, double b, double c, double d, double alpha)
{
        double p0,p1,p2,p3;
          p0=m[0]*a+m[1]*b+m[2]*c+m[3]*d;
         p1=m[4]*a+m[5]*b+m[6]*c+m[7]*d;
        p2=m[8]*a+m[9]*b+m[10]*c+m[11]*d;
         p3=m[12]*a+m[13]*b+m[14]*c+m[15]*d;
```

return(p3+alpha*(p2+alpha*(p1+alpha*p0)));　　}

附参考图例：



3.2　关键帧变形动画系统（线性插值与矢量线性插值）

3.2.1　任务概述

系统包括三个部分：

(1) 输入数据：包括初始形状数据和终止形状数据，一般为事先定义好的整型变量数据,如简单的几何物体形状（苹果，凳子，陶罐）以及简单的动物形状（大象，马）等。也可以设计交互界面，用户通过界面交互输入数据。

(2) 插值算法，包括线性插值和矢量线性插值。线性插值：对于初始和终止形状上每个点的坐标 Pi 进行线性插值得到物体变形的中间形状；矢量线性插值：对初始形状和终止形状上每两个相邻点计算其对应的长 Li 和角度 θi，然后对 Li 和 θi 进行线性插值得到中间长度和角度， 顺序连接插值后定义的各个矢量得到中间变化形状。

插值变量变化范围是[0，1], 插值变量等于 0 时对应于初始形状，插值变量等于 0 时对应于终止形状；数据类型为 float。

(3) 插值结果输出。

3.2.2　插值算法

(1) 线性插值：

指定两幅关键画面图形（最简单的是大小不同的两个矩形，分别由 4 个点构成。学生也可以自己构造更复杂的图形，如由若干点构成的手图形），然后计算两幅图对应点的线性距离来得到它们的中间画面图形。

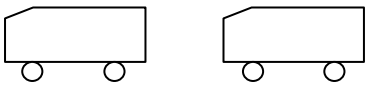
设图形上有 N 个点，（xi，yi), i=1,...N; 初始图形的点记为（x0i，y0i），终止图形记为（x1i，y1i），生成的中间图形记为（xti，yti），设生成 M 个画面，则有

```
for(j=1, j<M, j++) {
    t=(float)j/M;   //这里 t 是时间，其区间在[0，1]范围内
    for(i=1,i<N, i++)    {
        xti=(1-t)*x0i+t*x1i ，
        yti=(1-t)*y0i+t*y1i ，}
     画出（xti，yti）中间图形；
}
```

计算得到（xti，yti）之后，把初始图形，中间图形和终止图形显示在一张图上，也可以分别写入不同的图像 Im_i 上，然后用 Photoshop 的 Image ready 作成 gif 画观看结果。

进一步了解线性差值算法的性能，比如考察矩形在初始和终止画面上具有不同角度会带来什么影响。如下所示两个简化图形。

(1) 简化汽车图形



初始图形　　终止图形

(2) 简化铅笔图形

初始图形　　终止图形

（3）铅笔对向放置

初始图形　　终止图形

(2) 矢量线性插值：

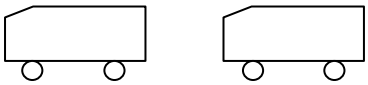
与线性差值框架类似，但插值变量不再是线性插值中的点坐标表（x,y), 而是把图形曲线上每两个邻近点看成一个矢量，这样就能把由 N 个点构成的曲线分解成 N-1 个矢量；

预处理：计算初始图像和终止图像物体曲线的极坐标，分别得到（r0,i，θ0,i）和（r1,i，θ1,i).

```
for(j=1,j<=M,j++) {
    t=(float)j/M;  //这里 t 是时间，其区间在[0，1]范围内
    for(i=1,i<N-1,i++)  {
      rt,i=(1-t)*r0,i+ t*r1,i,
      θt,i=(1-t)*θ0,i+t*θ1,i
     }
     把（rt,i，θt,i）定义的各个矢量首尾相接，画出中间图形；
}
```

进一步了解线性差值算法的性能，比如考察矩形在初始和终止画面上具有不同角度会带来什么影响。如下所示两个简化图形。

(1) 简化汽车图形



初始图形　　终止图形

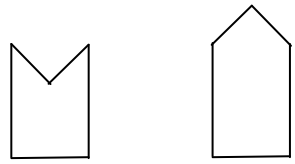
(2) 简化铅笔图形

初始图形　　终止图形

（3）铅笔对向放置

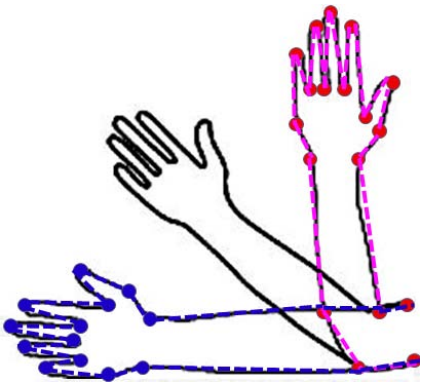
初始图形　　终止图形

(4) 凸凹对应多边形



初始图形　　终止图形

(5) 手臂旋转



粉色：初始图形，蓝色：终止图形

3.2.3 插值结果输出

模式 1：在屏幕窗口中直接显示

中间形状输出模块将插值后得到的物体在不同时间上的各个中间形状进行显示输出. 在 MFC 设备场景类中使用画笔和画刷实现绘制线条和形状，并对屏幕上的区域着色。会其它绘制方法的学生也可以用其他方法绘制。

1. 画笔类：创建画笔类 CPen 来指定屏幕上绘制的线条的颜色和宽度。

```
// Create the device context
CDC do (this)
// Create the pen
CPen 1Pen(PS_SOLID, 1, RGB(0,0,0));
// Select the pen as the current drawing pen
dc.SelectObject (&|Pen);
```

2. 画刷类：创建画刷类 CBrush 来定义区域如何被填充的画刷。
3. 创建应用程序外壳：在应用程序外壳中生成一个窗口作为画布， 打开一个新的 AppWizard 项目并提供一个适当的名称;
4. 添加图形能力。创建 OnPaint 函数，把所有的绘制图形的功能模块都加入这个函数，并显示图形。
5. 绘制线条: 先创建颜色表

```
const COLORREF m_Colors[8] = {
    RGB(0,0,0) // Black          RGB(0,0,255) //Blue
    RGB(0,255,0) //Green        RGB(0,255,255) //Cyan
    RGB(255,0,0) //Red          RGB(255,0,255) //Magenta
    RGB(255,255,0) // Yellow    RGB(255,255,255) // White }
```

然后添加绘制线条函数 DrawLine(Cpaint *pdc, int iCOlor).

模式 2：将各个中间图形用图像保存并转换成 AVI 动画文件。

特征点插值： 对物体形状上的特征点进行插值获得定义中间形状的特征点，再用样条曲线插值特征控制点得到最终的中间物体形状。

3.3 三维变形——Free Form Deformation（FFD）

3.3.1 任务概述

FFD的思想是不直接操作物体, 而是将物体嵌入一空间, 当所嵌的空间变形时, 物体也随之变形。

3.3.2 算法思想

(1) 确定物体的顶点(或控制顶点)在网格空间的位置。建立FFD块的局部坐标系:

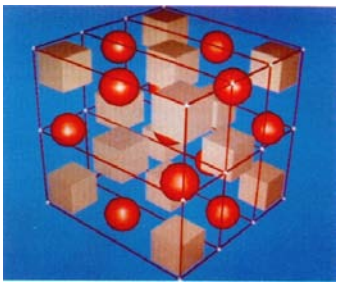
$$X(s,t,u)=X0 + sS+t\ T+uU$$

其中 $X0$ 为局部坐标原点, 且

$$s=\frac{T\times U\cdot(X-X_0)}{T\times U\cdot S},\ t=\frac{S\times U\cdot(X-X_0)}{S\times U\cdot T},\ u=\frac{S\times T\cdot(X-X_0)}{S\times T\cdot U}$$

$$0<s<1,\ 0<t<1 \text{ and } 0<u<1.$$

(2) 引进网格控制点Pijk，在S方向设置l+1个平面，T方向m+1个平面，U方向n+1个平面。如右图示，l=1, m=2, n=3. 平面形成的网格交点即控制点Pijk，用白色示意。

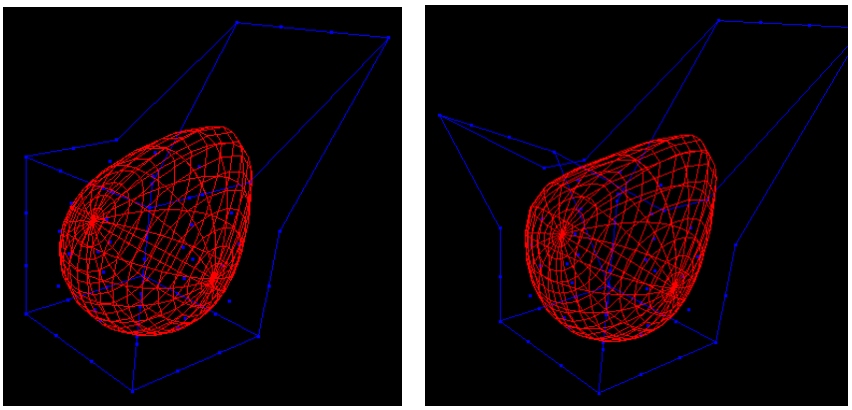


(3) 变形 FFD 块。根据动画设计需要移动控制顶点 Pijk,即改变控制顶点的坐标. 可以交互控制,也可以自动控制. 未变形时的 X 点，在控制点位置变化后，其对应变化的坐标 Xffd 计算如下，即把物体顶点 X 的局部坐标(s,t,u)代入下式:

$$\mathbf{X}_{ffd}=\sum_{i=0}^{l}\binom{l}{i}(1-s)^{l-i}s^{i}\left[\sum_{j=0}^{m}\binom{m}{j}(1-t)^{m-j}t^{j}\left[\sum_{k=0}^{n}\binom{n}{k}(1-u)^{n-k}u^{k}\mathbf{P}_{i,j,k}\right]\right]$$

参阅《计算机动画的算法基础》第 6.5.1 节，pp315-361。

附参 FFD 变形考图例：



3.4 路径曲线弧长参数化

3.4.1 任务概述

1。实现样条曲线弧长参数化。

2。在弧长参数化后的路径曲线上对某物体（如汽车）进行运动控制。

3。（可选）设计不同的速度曲线，如先从 0 开始加速，到一定速度 v 时再匀速，然后再开始减速至 0。

3.4.2 算法流程

设弧长函数为 s=A(u). Q(u)的弧长参数化是从 Q(u)到 $Q(A^{-1}(s))$。一般条件下 $A^{-1}(s)$无法解析表达，故采用数值求解。求解过程分为两步：

步骤1 采用二分查找法, 对于给定的弧长s_a，计算所对应的参数值u_a。

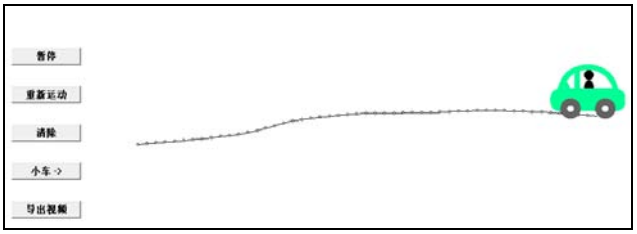
步骤2 给定某一参数u, 计 算弧长函数A(u)。由于A(u）是一个积分方程, 需采用数值积今的方法对其进行计算。

（参阅《计算机动画的算法基础》4.2节，pp183-186。）

3.4.3 代码提示

参阅《计算机动画的算法基础》4.2 节，pp186-188（注意：书中给出的代码有少量符号印刷错误，希望在编程实现过程中将错误纠正)。

附参考图例：



4 文档要求和评分标准

本实验系统设计开发完成后，需要经过现场运行验收才算通过。

1、设计文档包括：

（1）实验报告封面 （见下页）

（2）总体设计报告（同一个组里各同学该报告相同，报告里要注明组内分工)，

（3）模块设计报告（本人承担的部分，各人不同)，

（4）其他设计开发文档（如测试报告，开发体会、小结等)，

（5）源代码文件（组长提供全部源代码，其余同学只包含自己开发的模块的源代码)。

要求每个同学将设计文档打包为一个文件（文件名为“学号姓名.rar”）

2、实验采用“优、良、中、及格、不及格”五级评分制，具体的评分标准如下：

（1）实验最后得分由总体设计报告得分、详细设计报告得分和系统现场运行验收得分组成，其比例分别 30%、30%、40% 。

（2）报告及时提交，则根据报告的质量给“优、良、中”中相应分级，未及时提交，则在报告质量分级基础上降一级，未提交报告或报告为抄袭，相应的报告得分为“不及格”。

（3）若程序编写工作基本完成，但无法运行或无法进行测试，则根据程序质量给验收得分为“中、及格”

（4）若基本上未编写程序或程序纯属抄袭，验收得分为“不及格”。

5. 实验进度安排:

1. 路径控制曲线（ 2 次课）
2. 关键帧动画系统（2 次课）
3. 三维 Free Form Deformation （2 次课）
4. 路径曲线弧长参数化（2 次课）

浙江大学

本科实验报告

课程名称：

姓　　名：

学　　院：

　　系：

专　　业：

学　　号：

指导教师：

年　　月　　日