浙江大学

课程名称：计算机动画

姓　　名：**

学　　院：计算机学院

专　　业：数字媒体技术

学　　号：31601*****

指导教师：于金辉

2018 年　11 月　5　日

浙江大学实验报告

课程名称：计算机动画 实验类型：综合

实验项目名称：关键帧变形动画系统

学生姓名：** 专业：数字媒体技术 学号：31601*****

同组学生姓名：无 指导老师：于金辉

实验地点： 实验日期：2018 年 11 月 5 日

一、 任务概述

系统组成：

1. 输入数据：包括输入数据：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 θ i，然后对 Li 和 θ i 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。

二、 实验内容和原理

1. 插值算法

   a）线性插值：

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;
}
```

b) 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（r0,i， 0,i）和（r1,i， 1,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,
        t,i=(1-t)*  0,i+t*  1,i
    }
```

把（rt,i， t,i）定义的各个矢量首尾相接，画出中间图形；

```
}
```

2. 插值结果的输出：

屏幕直接输出，中间形状输出模块将插值后得到的图案在不同时间上的各个中间形状进行显示输出，采用 QPen 类来指定画笔的颜色，粗细和样式

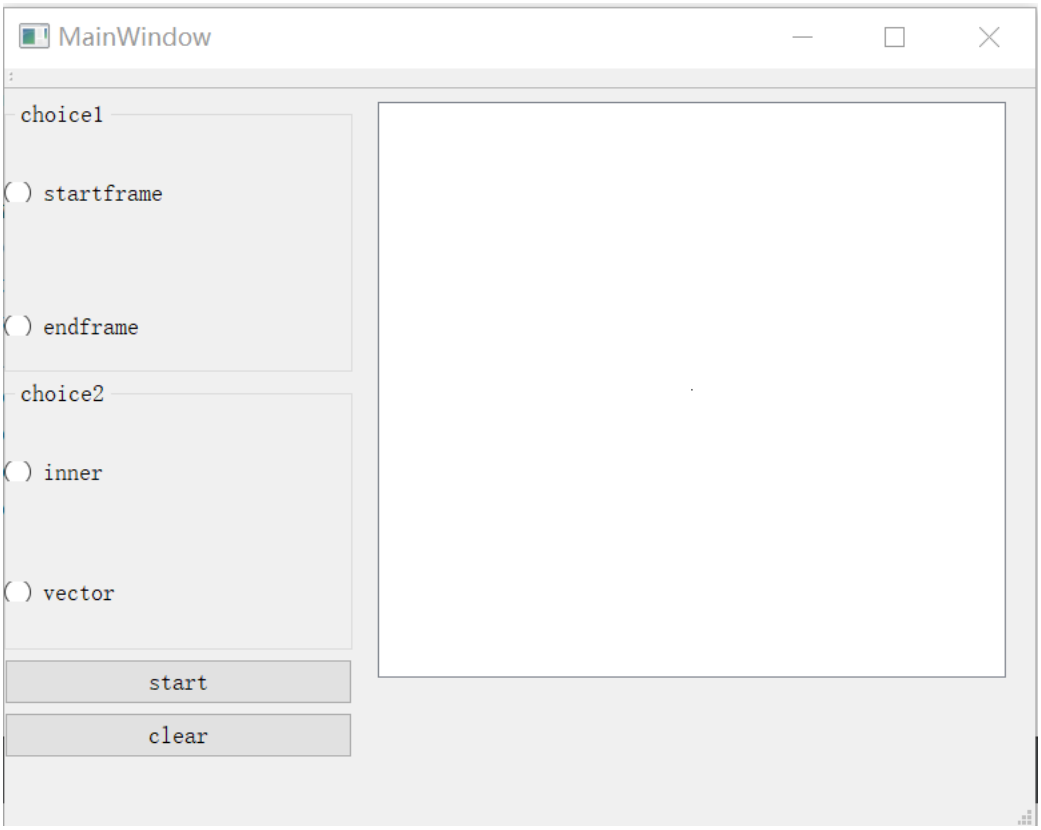
三、 实验器材

Qt creator 4.6.2　　Win10

四、 实验步骤

1. 界面设计：

   在 Qt creator 的 ui 设计视图中完成标签，按钮等的设置与布局。



2. 鼠标事件处理：

   在每次点击时直接生成一个点，调用对象的 addpoint 函数，存入坐标，实时绘制点和连线

```
void MainWindow::mousePressEvent(QMouseEvent *event)
{
    linee.addPoint(event->x()-5,event->y()-5);
}
```

```
void line::addPoint(int _x, int _y)
{
    numm = num[linetype];
    x[linetype][numm]=_x;
    y[linetype][numm]=_y;
    num[linetype]++;
    QGraphicsEllipseItem *point = new QGraphicsEllipseItem(_x,_y,5,5,Line);
    point->setPen(pen);
    if(num[linetype]==1)
        return;
    QGraphicsLineItem *linee = new QGraphicsLineItem(x[linetype][numm-1]+5,y[linetype][numm-1]+5,_x,_y,Line);
    linee->setPen(pen);
    return;
}
```

3. 绘制中间过程：线性插值实现（addline 函数中）：

```
for(int i=0;i<50;i++)
{
    for(int j=0;j<num[0];j++)
    {
        inx[i][j] = 1/50.0*i*x[1][j]+1/50.0*(50-i)*x[0][j];
        iny[i][j] = 1/50.0*i*y[1][j]+1/50.0*(50-i)*y[0][j];
    }
}
```

4. 绘制中间过程，矢量线性插值实现（addline 函数中）：

```
if(drawtype==1)
{
    int number;
    int x1,y1,x2,y2;
    double t,t1,t2,l,l1,l2;
    number=50;
    for(int j=0;j<50;j++){
        inx[j][0]=1/50.0*j*x[1][0]+1/50.0*(50-j)*x[0][0];
        iny[j][0]=1/50.0*j*y[1][0]+1/50.0*(50-j)*y[0][0];
    }
    for(int i=1;i<num[0];i++){
        x1=x[0][i]-x[0][i-1];
        y1=y[0][i]-y[0][i-1];
        x2=x[1][i]-x[1][i-1];
        y2=y[1][i]-y[1][i-1];
t1=asin(y1/sqrt(qPow(x1,2)+qPow(y1,2)));
t2=asin(y2/sqrt(qPow(x2,2)+qPow(y2,2)));
if(x1<0)
    t1=(PI-t1);
if(x2<0)
    t2=(PI-t2);
l1=qPow(x[0][i]-x[0][i-1],2)+qPow(y[0][i]-y[0][i-1],2);
l2=qPow(x[1][i]-x[1][i-1],2)+qPow(y[1][i]-y[1][i-1],2);
l1=qSqrt(l1);
l2=qSqrt(l2);
while(qAbs(t1-t2)>PI)
{
    if(t1>t2)
    {
        t1-=PI*2;
    }
    else
    {
        t2-=PI*2;
    }
}
```

```
for(int j=0;j<50;j++){
    t=1/50.0*j*t2+1/50.0*(50-j)*t1;
    l=1/50.0*j*l2+1/50.0*(50-j)*l1;
    inx[j][i]=inx[j][i-1]+l*qCos(t);
    iny[j][i]=iny[j][i-1]+l*qSin(t);
}
```

5. 界面清除：

   点击清除按钮后，画布被清空。

```
void line::clear()
{
    num[0]=num[1]=0;
    scene.removeItem(Line);

    timer->stop();
}
```

6. 动画开始：

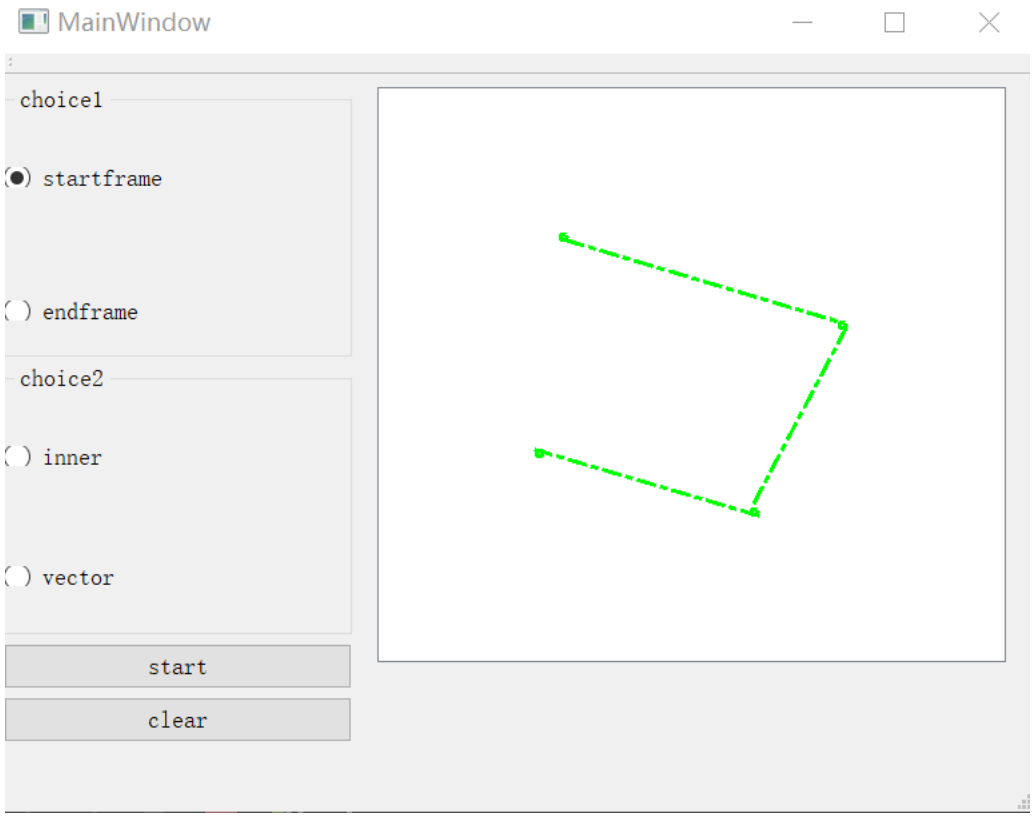
   调用 addline 函数，绘制中间过程。绑定 nextflame 函数到 timer，删除已显示过的状态，绘制下一帧，实现动画效果。

```
void line::start()
{
    if(num[0]!=num[1])
    {
        return;
    }
    addLine();
    inNum=0;
    timer->start(100);
}
```

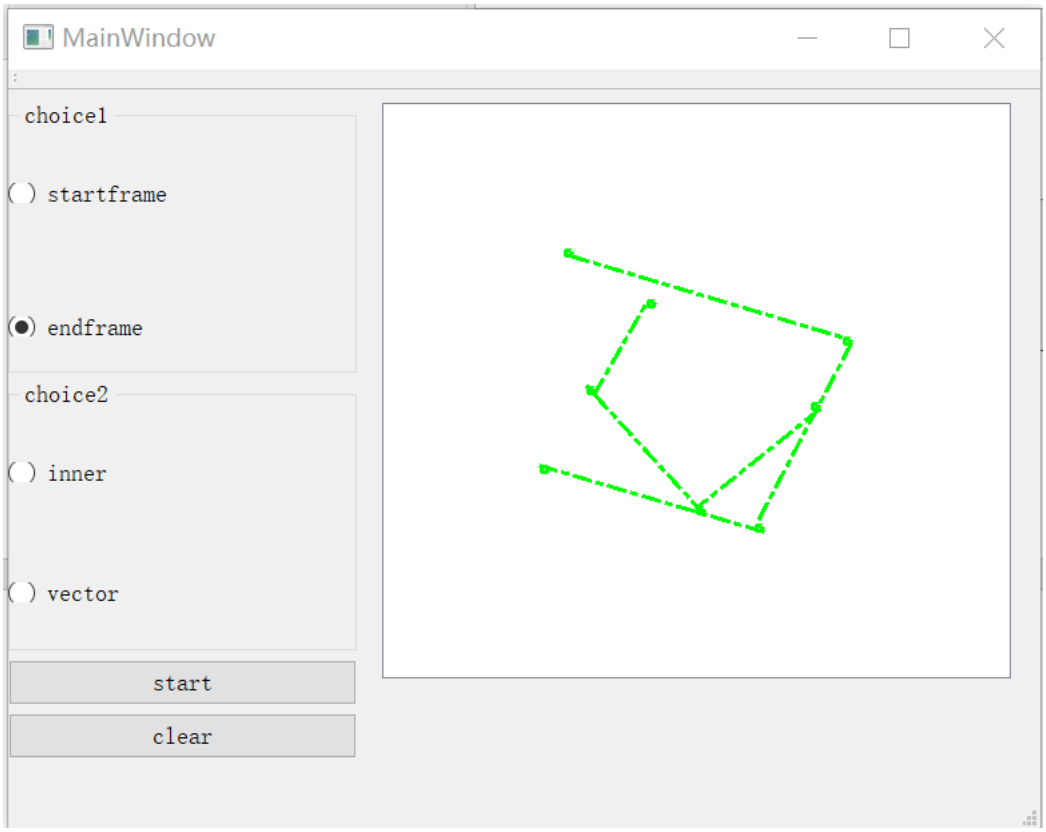
```
void line::nextflame()
{
    QList<QGraphicsItem *> list_line = inLine->childItems();
    for ( int i=0; i!=list_line.size(); i++ )
    {
        scene.removeItem(list_line.at(i));
    }

    if(inNum==50)
        return;

    for(int i=0;i<num[0];i++)
    {
        QGraphicsEllipseItem *point = new QGraphicsEllipseItem(inx[inNu
        point->setPen(pen);
        if(i==0)
            continue;
        QGraphicsLineItem *linee = new QGraphicsLineItem(inx[inNum][i-1
                iny[inNum][i],inLine);
        linee->setPen(pen);
    }
    inNum++;
}
```

五、 实验结果分析

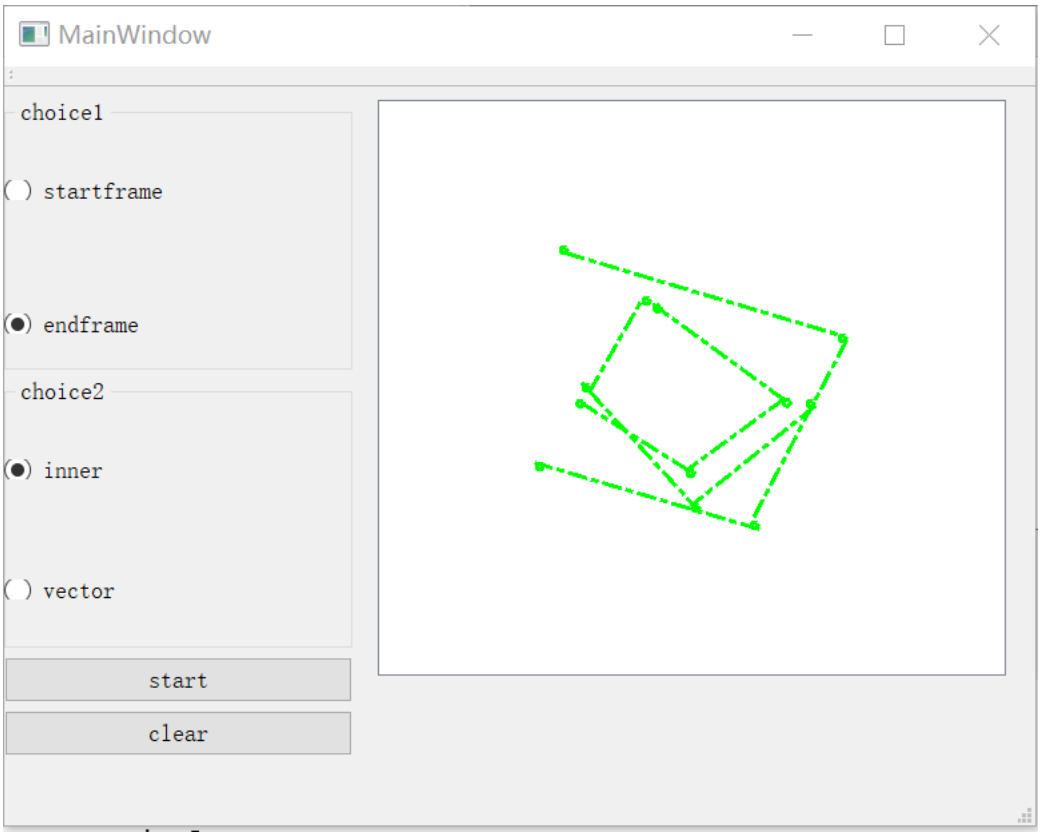
1. 选择 startflame，绘制开始帧



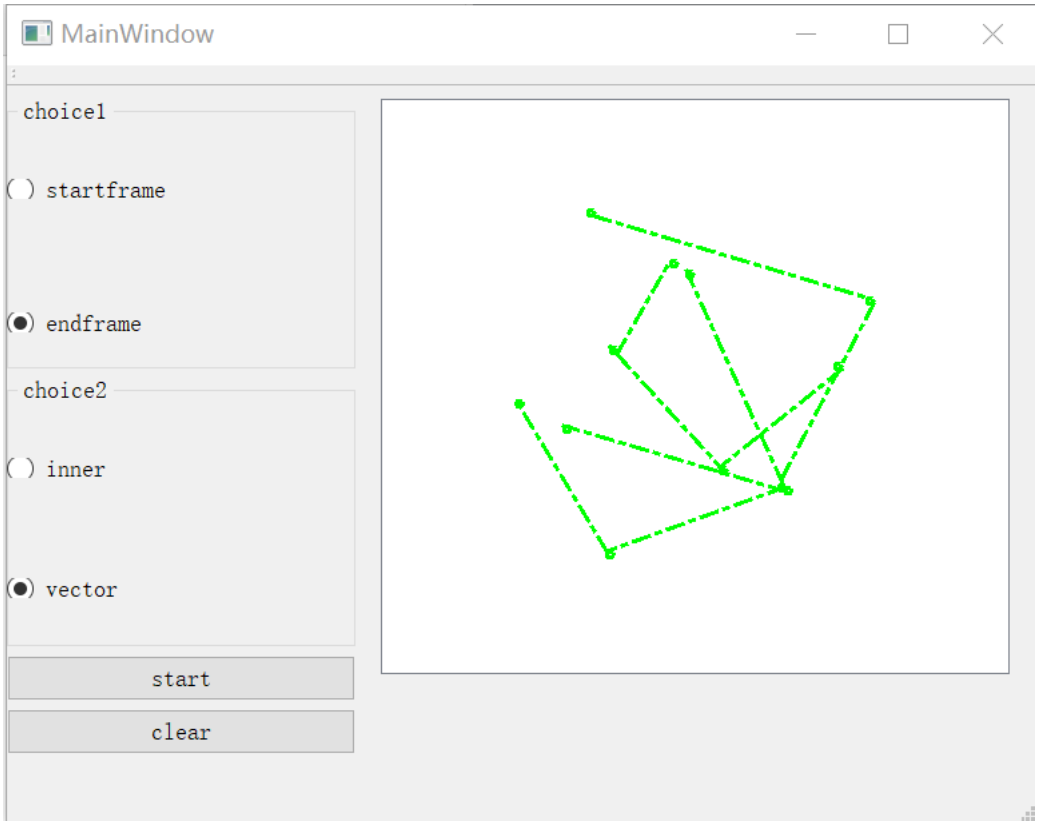
选择 endflame 绘制结束帧



选择 inner，进行线性插值（详情见录屏）



选择 vector，进行矢量线性插值（详情见录屏）



详细情况见录屏

六、 实验感悟与问题

通过这次实验，我更深入的了解了 Qt 的界面设计，学会了 QGraphicsScene，item 的使用。在实验过程中，我对线性插值和矢量线性插值的特征与区别有了更进一步的了解，实现了他们的算法。。但是由于能力有限，实验的结果并不完美，还有一些功能尚未实现，界面的设计也有待提升。希望能在对 Qt 及计算机动画的算法有更深的了解以后，再能来实现他们。