```
//
//  MamRendererPlugin_FixedEllipse.cpp
//  iOSMamPlayer
//
//  Created by Stephen Malinowski on 4/16/11.
//  Copyright 2011 Music Animation Machine. All rights reserved.
//

#include <math.h>
#include <assert.h>

#include "MamRendererPlugin_FixedEllipse.h"
#include "MamUtil.h"
#include "MamMemoryDiagnostics.h"
#include "RendererControl.h"
#include "glcheck.h"
#include "TimeValue.h"

MamRendererPluginInstance_FixedEllipse::MamRendererPluginInstance_FixedEllipse()
{
}
MamRendererPluginInstance_FixedEllipse::~MamRendererPluginInstance_FixedEllipse()
{
}

void MamRendererPluginInstance_FixedEllipse::setPartDefaults(MamPartInstance*theMamp,MidiPartInstance*theMidip,double theNowMidi)
{
        // these are things that don't change from note to note
        mamPart=theMamp;
        midiPart=theMidip;
        nowMidi=theNowMidi;
}

// make these configurable in the future
static double kSpiralOuter=0.25;
static double kSpiralInner=0.005;
static double kSoundingMultiple=3;

extern double firstNoteOnTimeGlobal,lastNoteOnTimeGlobal;

double MamRendererPluginInstance_FixedEllipse::thetaShiftAtTime(double x,double t)
{
        dRange range=mamPart->thetaRange;
        double theta=HALF_PI-interpolate(0,x,1,range.min,range.max)*TWO_PI;
        double tThetaShift;
//      MamPackage *package=_r->script->package;
        if(mamPart->timeThetaSet)
        {
           tThetaShift=mamPart->timeThetaSet->valueAtTime(t)*TWO_PI;
        }
        else
        {
           tThetaShift=interpolate(9,nowMidi,217,0,EIGHT_PI)+QUARTER_PI;
        }
        theta+=tThetaShift;
        return theta;
}


dPoint MamRendererPluginInstance_FixedEllipse::eventToZeroBasedPoint(cMidiEvent*event,double&theta)
{
        if(!event)
        {
           theta=NAN;
           return origin;
        }

    dPoint pLinear=eventStartToPoint(event);
    pLinear.y=interpolate(-1,pLinear.y,1,0,1);        // recast Y range from {-1,1} to {0,1}
    pLinear.x=interpolate(-1,pLinear.x,1,0,1);        // recast X range from {-1,1} to {0,1}

        // convert from linear to radial
        theta=thetaShiftAtTime(pLinear.x,nowMidi);

        double radius=pLinear.y;
        dPoint pRadial(cos(theta)*radius,sin(theta)*radius);
        if(0)
        {
           // make circular
           pRadial.x*=g->xscale();
        }
        return pRadial;
}

void MamRendererPluginInstance_FixedEllipse::fillCircle(dPoint p,double r,MamColor c,double add3Doutline,bool sounding)
{
        if(add3Doutline)
        {
           MamColor background=mamPart->renderer->script->backgroundFillColor;
           MamColor backgroundAlphaZero=alphaZero(background);

           const int kNumBands=8;
           const double kFadePower=0.3;
           for(double i=0;i<kNumBands;++i)
           {
                      double rInner=i/kNumBands;
                      double rOuter=(i+1)/kNumBands;
                      double fadeProgressInner=pow(rInner,kFadePower);
                      double fadeProgressOuter=pow(rOuter,kFadePower);
                      MamColor cInner=interpolateColor(0,fadeProgressInner,1,background,backgroundAlphaZero);
                      MamColor cOuter=interpolateColor(0,fadeProgressOuter,1,background,backgroundAlphaZero);
                      g->fillCircleRing(p,r+add3Doutline*rInner,r+add3Doutline*rOuter,cInner,cOuter);
           }
        }
        else
        {
           MamColor fadeToColor=mamPart->renderer->script->backgroundFillColor;
           fadeToColor.a=c.a;
           if(mamPart->fadeEdgeToBlack)
           {
           }
           else
```

```
		{
			fadeToColor=c;
		}

		const int kNumBands=8;
		const double kFadePower=3;
		for(double i=0;i<kNumBands;++i)
		{
			double rInner=i/kNumBands;
			double rOuter=(i+1)/kNumBands;
			double fadeProgressInner=pow(rInner,kFadePower);
			double fadeProgressOuter=pow(rOuter,kFadePower);
			MamColor cInner=interpolateColor(0,fadeProgressInner,1,c,fadeToColor);
			MamColor cOuter=interpolateColor(0,fadeProgressOuter,1,c,fadeToColor);
			g->fillCircleRing(p,r*rInner,r*rOuter,cInner,cOuter);
		}
	}
}

void MamRendererPluginInstance_FixedEllipse::renderEvent(cMidiEvent*ePrev,
																		cMidiEvent*eCurr,
																		cMidiEvent*eNext,
																		double theNow,
																		double add3Doutline)
{
	// fake event that points to where the note would be if it were moving
    cMidiEvent eNow;
    eNow.eventTimeS=theNow;
    eNow.pitch=eCurr->pitch;

	// fake event for where current note ends
    cMidiEvent eFakeNext;
    eFakeNext.eventTimeS=eNext?eNext->eventTimeS:eCurr->eventTimeS+eCurr->eventDurationS;
    eFakeNext.pitch=eCurr->pitch;

	double thetaPrev;
	double thetaCurr;
	double thetaNow;
    dPoint pPrev=eventToZeroBasedPoint(ePrev,thetaPrev);
    dPoint pCurr=eventToZeroBasedPoint(eCurr,thetaCurr);
    dPoint pNow =eventToZeroBasedPoint(&eNow,thetaNow );
	MamColor cOffPast=mamPart->colors.offColorPastForNote(mamPart->coloringMode,eCurr->pitch,eCurr->velocity);
	MamColor cOn=mamPart->colors.onColorForNote(mamPart->coloringMode,eCurr->pitch,eCurr->velocity);
	double tStart=eCurr->eventTimeS;
	double tEnd=tStart+eCurr->eventDurationS;
	double tNext=eNext?eNext->eventTimeS:tEnd;

    double height=mamPart->pitchHeight();
    double tFactor=min(eCurr->eventDurationS,mamPart->referenceTime);
    double ptFactor=pow(tFactor,mamPart->ballPowerFactor);
    height=interpolate(0.0,ptFactor,1.0,height*mamPart->barHeightMin,height*mamPart->barHeightMax);
	MamPackage *package=_r->script->package;
	double sizeAtTime;
	if(mamPart->timeSizeSet)
	   sizeAtTime=mamPart->timeSizeSet->valueAtTime(theNow);
	else
	   sizeAtTime=interpolate(package->songStartTime,theNow,package->songEndTime,mamPart->sizeRangeInTime.min,mamPart->sizeRangeInTime.max);
	height*=sizeAtTime;

	dPoint pPrevExtend=pPrev;
	dPoint pCurrExtend=pCurr;
	dPoint pNowExtend=pNow;
	if(mamPart->extendCursor)
	{
	   pPrevExtend *= sqrt(2.0)/pPrevExtend.distTo(g->xscale(),g->yscale());
	   pCurrExtend *= sqrt(2.0)/pCurrExtend.distTo(g->xscale(),g->yscale());
	   pNowExtend  *= sqrt(2.0)/pNowExtend.distTo(g->xscale(),g->yscale());
	}

	// show position spiral dot
	dPoint pCurrSpiral;
	if(mamPart->showCompass && mamPart->showOnlyCursor==false)
	{
	   double radius=interpolate(package->songStartTime,eCurr->eventTimeS,package->songEndTime,kSpiralInner,kSpiralOuter);
	   pCurrSpiral=dPoint(cos(thetaCurr)*radius*g->xscale(),sin(thetaCurr)*radius);
	   if(!add3Doutline)
	   {
			if(theNow < tStart)
				g->fillCircle(pCurrSpiral,0.003,cOffPast);
			else
			{
				g->fillCircle(pCurrSpiral,0.007,interpolateColor(0,0.2,1,cOffPast,cOn));
			}
	   }
	}
	else
	{
	   pCurrSpiral=origin;
	}

	// draw it
	if(mamPart->showFuture)
	{
	   // show all events (special case)
	   fillCircle(pCurr,height,cOffPast,add3Doutline,false);
	}
	else if(theNow < tStart)
	{
	   // future, just draw circle
//	   if(!add3Doutline  && mamPart->showOnlyCursor==false)
//			g->drawCircle(pCurrSpiral,height,cOff);
	}
	else
	{
	   // not in future; connect to previous point
	   if(ePrev && fabs(thetaCurr-thetaPrev < 0.1) && mamPart->showConnectingLines)
	   {
			g->drawLine(pPrev,pCurr,cOffPast);
	   }

	   if(theNow < tEnd)
	   {
```

```
                        // sounding, fade color
                        MamColor cFade=interpolateColor(tStart,theNow,tEnd,cOn,cOffPast);

                        // and interpolate height
                        height=interpolate(tStart,theNow,tEnd,height*kSoundingMultiple,height);
                        if(mamPart->showOnlyCursor==false)
                                fillCircle(pCurr,height,cFade,add3Doutline,true);

                        // cursor
                        if(eNext)
                        {
                                // there is a next note to move toward
                                if(1)
                                {
                                        // cursor
                                        if(mamPart->showCursor && mamPart->showNowLine)
                                                g->drawLine(pCurrSpiral,pNowExtend,cOn);
                                }
                                else
                                {
                                        MamColor cBackgroundFade=interpolateColor(tStart,theNow,tEnd,cOn,alphaZero(cOn));
                                        if(mamPart->fillCursor)
                                                g->fillTriangle(pCurrSpiral,pCurrExtend,pNowExtend,cBackgroundFade);
                                        MamColor cLineFade=interpolateColor(tStart,theNow,tEnd,cOn,alphaZero(cOn));
                                        if(!add3Doutline)
                                                g->drawLine(pCurrSpiral,pCurrExtend,cLineFade);
                                        if(!add3Doutline && mamPart->showNowLine)
                                                g->drawLine(pCurrSpiral,pNowExtend,cOn);
                                }
                        }
                        else
                        {
                                // cursor
                                MamColor cLineFade=interpolateColor(tStart,theNow,tEnd,cOn,alphaZero(cOn));
                                if(mamPart->showCursor)
                                        g->drawLine(pCurrSpiral,pCurrExtend,cLineFade);
                        }
                }
                else if(theNow < tNext)// && ((tNext-tEnd)<1.0))
                {
                        if(mamPart->showOnlyCursor==false)
                        {
                                // before next note, still show line
                                if(mamPart->fadeOutDuration)
                                {
                                        double dt=theNow-tEnd;
                                        double tFull=tEnd+mamPart->fadeOutDuration;
                                        if(tFull<dt)
                                                cOffPast=alphaZero(cOffPast);
                                        cOffPast=interpolateColor(tEnd,theNow,tFull,cOffPast,alphaZero(cOffPast));
                                }
                                fillCircle(pCurr,height,cOffPast,add3Doutline,true);
                        }
                        if(!add3Doutline && mamPart->showCursor)
                                g->drawLine(pCurrSpiral,pCurrExtend,cOffPast);
                }
                else
                {
                        // in past
                        if(mamPart->showOnlyCursor==false)
                        {
                                if(mamPart->fadeOutDuration)
                                {
                                        double dt=theNow-tEnd;
                                        double tFull=tEnd+mamPart->fadeOutDuration;
                                        if(tFull<dt)
                                                cOffPast=alphaZero(cOffPast);
                                        cOffPast=interpolateColor(tEnd,theNow,tFull,cOffPast,alphaZero(cOffPast));
                                }
                                fillCircle(pCurr,height,cOffPast,add3Doutline,false);
                        }
                }
        }
}

void MamRendererPluginInstance_FixedEllipse::renderPart(MamPartInstance*theMamp,
                                                                                                                MidiPartInstance*theMidip,
                                                                                                                double theNow,
                                                                                                                double add3Doutline)
{
        setPartDefaults(theMamp,theMidip,theNow);
        assert(theMamp->colors.finalized);

        // get the first NOTE_ON event we're interested in
        cMidiEvent*event=getStartingEvent(theMamp,theMidip,theNow);
        cMidiEvent*next=NULL;
        cMidiEvent*prev=NULL;

        bool doneOne=false;

        if(mamPart->showRangeSet)
        {
           vector <dRange> *rs=&mamPart->renderer->rendererPlugin->rendererInstance->parent->package->rangeSet;
           if(rs->size())
           {
                        const double radiusInner=0.05;
                        const double radiusOuter=1.0;
                        const double dThetaMin=TWO_PI/360;
                        MamColor cOff=mamPart->colors.offColorForNote(mamPart->coloringMode,60,63);
                        MamColor cOn=mamPart->colors.onColorForNote(mamPart->coloringMode,60,63);
                        for(int i=0;i<rs->size();++i)
                        {
                                double xfrom=(*rs)[i].min;
                                double xto=(*rs)[i].max;
                                double thetaTo=thetaShiftAtTime(xfrom,theNow);        // n.b. angles go in opposite direction
                                double thetaFrom=thetaShiftAtTime(xto,theNow);
                                dPoint P[2]={origin,origin};
                                dPoint R[2]={dPoint(radiusInner,radiusInner),dPoint(radiusOuter,radiusOuter)};
                                dPoint DP[2]={origin,origin};
                                dPoint DR[2]={origin,origin};
                                double dTheta=thetaTo-thetaFrom;
                                int nSlices=(int)ceil(dTheta/dThetaMin);
```

```
                                double dThetaSlice=dTheta/(double)nSlices;
                                for(int j=0;j<nSlices;++j)
                                {
                                        double from=j   *dThetaSlice;
                                        double to= (j+1)*dThetaSlice;
                                        MamColor c=interpolateColor(0.0,(double)j,(double)nSlices,cOff,cOn);
                                        MamColor C[2]={c,alphaZero(c)};
                                        double THETA[2]={thetaFrom+from,thetaFrom+to};
                                        g->fillVEllipseRing(P,R,DP,DR,C,THETA);
                                }
                        }
                }
                return;
        }

        // walk through NOTE_ON events, drawing the bars
        while(event)
        {
                next=theMamp->nextOn(event);

                if(theMamp->showFuture)
                {
                        // draw events in both past and future
                        renderEvent(prev,event,next,theNow,add3Doutline);
                }
                else
                {
                        // is this a note we actually need to draw?
                        if(event->eventTimeS<theMamp->xy.leftTimeMargin())
                        {
                                // no, don't bother with ones that are too early
                        }
                        else if (theMamp->xy.rightTimeMargin()<event->eventTimeS)
                        {
                                // no, stop if we're past the last note of interest
                                break;
                        }
                        else
                        {
                                // yes
                                // if it's the first, save this for our starting point for next frame
                                if(!doneOne)
                                {
                                        theMamp->firstRendered=event;
                                        doneOne=true;
                                }

                                renderEvent(prev,event,next,theNow,add3Doutline);
                        }
                }
                prev=event;
                event=next;
        }
}

void MamRendererPluginInstance_FixedEllipse::drawFrame(MidiData*midi,
                                                       MamRendererInstance *r,
                                                       double t,
                                                       double tSmoothed,
                                                       double alpha,
                                                       unsigned int width,
                                                       unsigned int height)
{
        _r=r;
    r->g->setWidthAndHeight(width,height);
        g->startBatch(kBatchFillCircle);
        int nParts=r->part.size();
        for(int partNum=0;partNum<nParts;++partNum)
        {
           MamPartInstance *pi=r->part[partNum];
           int bits=pi->selectionMask.min;
           int mask=pi->selectionMask.max;
           extern int globalFrameCount;
           int maskedBits=globalFrameCount&mask;
           if(maskedBits==bits)
           {
                        g->setGlobalAlpha(pi->respectGlobalAlpha?alpha:1.0);
                        MidiPartInstance *mp=midi->part[r->part[partNum]->partNumber];
                        if(pi->add3Doutline)
                                renderPart(pi,mp,t,pi->add3Doutline);
                        renderPart(pi,mp,t,0);
           }
        }
        g->finishBatch();
}

void MamRendererPluginInstance_FixedEllipse::finalizeData(MamRendererInstance*ri,MamRenderer*r)
{
        rendererInstance=ri;
        renderer=r;
        g=ri->g;
    int nParts=ri->part.size();
    for(int i=0;i<nParts;++i)
        ri->part[i]->colors.finalizeData(ri->part[i]);
}
```