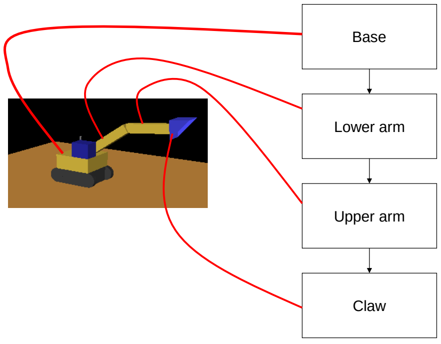
Animator Project

CSE 557

In this project…

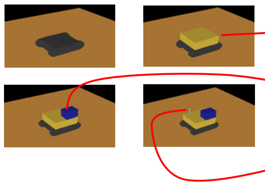
- Create your own hierarchical model
- Create keyframe animations
- Create *smooth* keyframe animations
- Create particle systems
- Create a cool animation artifact!

Hierarchical models



Hierarchical models

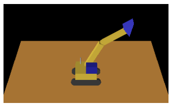
- First requirement: Make your own hierarchical model
  - Inherit from ModelerView class. The main() function in your model cpp file will update the UI.
  - Use simple primitives (box, cylinder, sphere, triangle)
  - Use OpenGL matrix stack commands to control which level of hierarchy you are manipulating
    - glPushMatrix() means “remember where you are”
    - glPopMatrix() means “go back to where you were”
  - Use OpenGL transformation commands to control current position



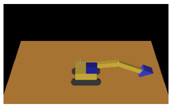
```
void rotation_base(float h) {
    setDiffuseColor( 0.85, 0.75, 0.25 );
    setAmbientColor( 0.95, 0.75, 0.25 );
    glPushMatrix();
        glPushMatrix();
            glScalef(4.0, h, 4.0);
            y_box(1.0f); // the rotation base
        glPopMatrix();
        setDiffuseColor( 0.15, 0.15, 0.65 );
        setAmbientColor( 0.15, 0.15, 0.65 );
        glPushMatrix();
            glTranslatef(-0.5, h, -0.6);
            glScalef(2.0, h, 1.6);
            y_box(1.0f); // the console
        glPopMatrix();
        setDiffuseColor( 0.65, 0.65, 0.65 );
        setAmbientColor( 0.65, 0.65, 0.65 );
        glPushMatrix();
            glTranslatef( 0.5, h, 0.6 );
            glRotatef( -90.0, 1.0, 0.0, 0.0 );
            drawCylinder( h, 0.05, 0.05 ); // the pipe
        glPopMatrix();
    glPopMatrix();
}
```

Keyframe animation

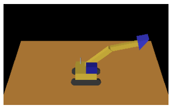
t=0.0



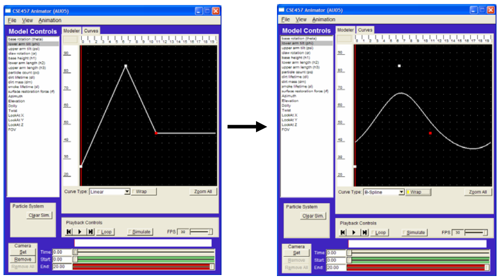
t=6.5



t=11.0



Smooth animation curves



Animation curves

- You will add three kinds of curves: Bezier, Catmull-Rom, and B-spline
- For each, make a subclass of CurveEvaluator
  - Given a set of control points, calculate many sampled points on the curve
  - Allow each curve to be “wrapped” so that the curve is continuous if the animation is looped

Particle systems

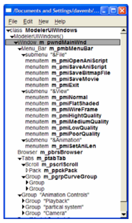
- A particle system is a collection of point-objects with forces acting on them
- We keep track of each particle’s position and velocity. At each timestep, update the position and velocity based on the forces acting on the particle
- You will *bake* your particle system, e.g. you will save the calculated positions of each particle at each timestep so that it can be easily replayed.

Artifact

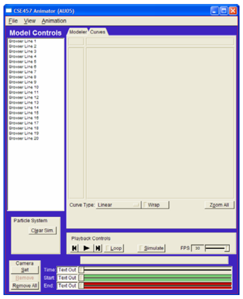
- Animator saves your movie as a set of sequentially-named .bmp files
- To make a movie, use Adobe Premiere (on lab machines) or use your favorite other movie editor (Camtasia, VirtualDub, etc)

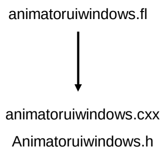
FLUID

- You can add new UI widgets using the FLUID binary



```
class ModelerUIWindows
  ModelerUIWindows()
    Window m_pwndMainWnd
      Menu_Bar m_pmbMenuBar
        submenu "&File"
          menuitem m_pmiOpenAniScript
          menuitem m_pmiSaveAniScript
          menuitem m_pmiSaveBitmapFile
          menuitem m_pmiSaveMovie
          menuitem m_pmiExit
        submenu "&View"
          menuitem m_pmiNormal
          menuitem m_pmiFlatShaded
          menuitem m_pmiWireFrame
          menuitem m_pmiHightQuality
          menuitem m_pmiMediumQuality
          menuitem m_pmiLowQuality
          menuitem m_pmiPoorQuality
        submenu "&Animation"
          menuitem m_pmiSetAniLen
      Browser m_pbrsBrowser
      Tabs m_ptabTab
        Scroll m_pscrlScroll
          Pack m_ppckPack
        Group m_pgrpCurveGroup
          Group
          Group
      Group "Animation Controls"
        Group "Playback"
        Group "partical system"
        Group "Camera"
```





animatoruiwindows.fl

animatoruiwindows.cxx

Animatoruiwindows.h