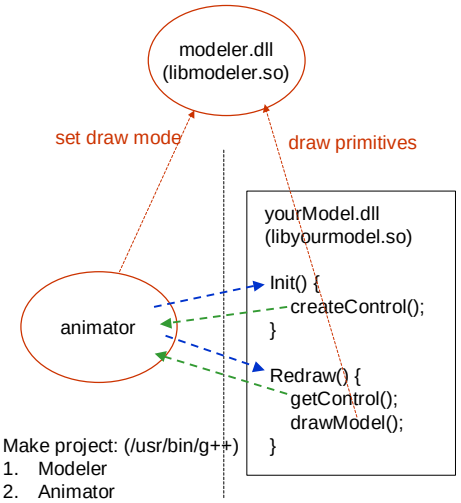
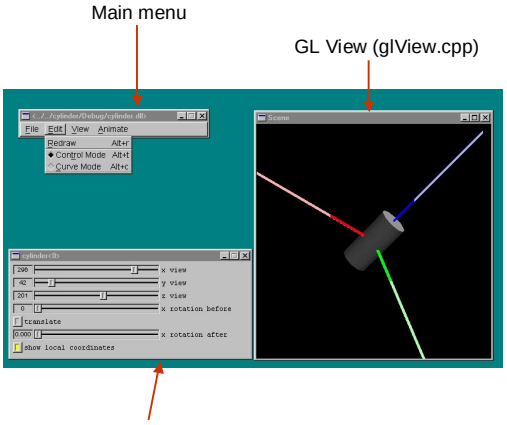
Architecture



Make project: (/usr/bin/g++)
1. Modeler
2. Animator
3. Your model

Execute animator:
1. Copy modeler.dll(libmodeler.so) to the directory where animator sits.
2. setenv LD_LIBRARY_PATH . (Linux only)

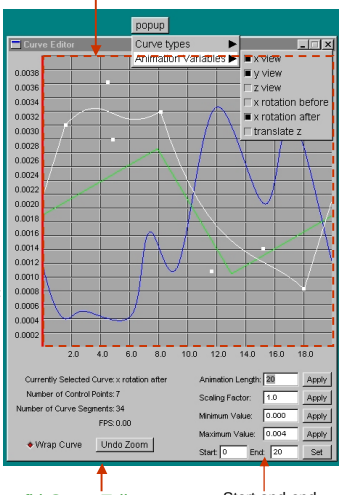
Control mode



Curve mode

curveDisplay.cpp for dispatching events
curveEditor.cpp for drawing

(Alt+Left=Middle)
Left mouse click :
Add a control point.
Left mouse drag :
move a control point.
Ctrl+Left mouse click:
delete ctrl point.
Middle mouse click :
activate curve.
Shift+Left mouse drag :
enlarge region.



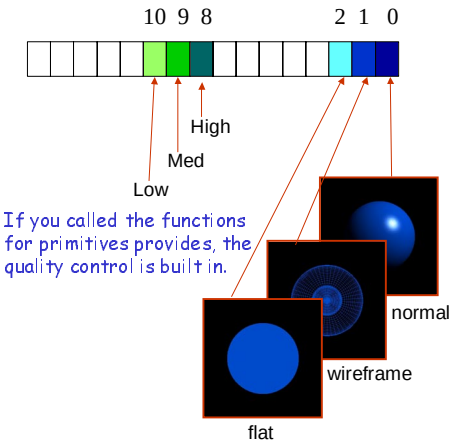
fltkCurveEditor.cpp

Start and end Frmes for movie

What to provide in your model

bool init(void *args) — called once

void redraw(int drawmode) — called whenever system need to redraw



If you called the functions for primitives provides, the quality control is built in.

Interface to animator

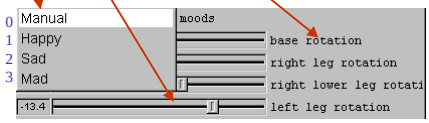
Create control

scale(id, name, min, max, res, start)

checkbox(id, name, start)

menu(interp, id, name, start, ...)

curve(id, name, min, max, start)



Access control value

get_control_d(id) : float

get_control_i(id) : int

get_control_b(id) : bool

get_time() : float (current time in animation)

OpenGL-related functions

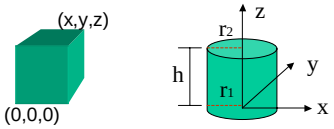
(You don't need to call them if you can program in OpenGL)

sphere(r)

triangle(x1, y1, z1, x2, y2, z2, x3, y3, z3)

box(x, y, z)

cylinder(h, r1, r2)



ambient_color(r, g, b);

diffuse_color(r, g, b);

specular_color(r, g, b);

shininess(s);

```
#include <modelerdll.h>
#define SPHERE_RADIUS   1
MODEL("sphere");        Announce your model name.
                        EXTREMELY IMPORTANT

                        argument string passed in.
bool init(void *args) {  "" for single object.
  scale( SPHERE_RADIUS, "radius", 0.1f, 3.0f, 0.01f, 1.0f );
  return true;                  Create a slider widget
}

void redraw(int drawmode) {
  glMatrixMode( GL_MODELVIEW );
  glLoadIdentity();                    reset modelview matrix
  glTranslatef( 0.0, 0.0, -15.0 );
  ambient_color( 0.0, 0.0, 0.2 );
  diffuse_color( 0.0, 0.3, 0.8 );      set material
  sphere( get_control_d( SPHERE_RADIUS ) );
}
  Draw a sphere     Radius is set by current value
                    Kept by the corresponding widget
```

Add curve types

1. Add the evaluator in CurveEditor.cpp
2. Add a class which inherits from CurveEvaluator

```
void evaluateCurve(const vector<Point> & control_points,
vector<Point> & evaluated_curve_points,
const double & animation_time,
const bool & wrap_control_points);
```