lab.h

```
#include <config.h>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <FL/Fl_Cairo_Window.H>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cmath>
Fl_Cairo_Window* makeDrawWindow();
void drawCB(Fl_Cairo_Window* cw,cairo_t* cr);
const int width = 300;
const int height = 300;
extern Fl_Cairo_Window * dw;
struct PSCmd
{
    std::string cmd;
    double x, y;
    double angle1,angleStart,angleEnd;
    double radius;
};
extern PSCmd* cmds;
extern int N; // number of valid commands in ps file
void getPSData(std::string fn);
void parsePSData(std::string fn);
```

- contains all headerfiles, global vairables, and struct

getPSData.cpp

```
//this function will create and fill the cmds array with all the ps
//commands needed to draw the picture in fn
#include "lab.h"
PSCmd* cmds;
void getPSData(std::string fn)
{
    std::ifstream ifs(fn);
    std::string s;
    int count = 0;
    while(getline(ifs,s))
    {
        count++;
    }
    std::cout << count << std::endl;
    cmds = new PSCmd[count];
    ifs.close();
}
```

- counts and copies postscript lines

parsePSData.cpp

```
#include "lab.h"
//fill the cmds array with actual PS comments
int N; void parsePSData(std::string fn)
{std::ifstream ifs(fn); std::string s; int i = 0;
//read lines e.g. 2 1 moveto
    while(getline(ifs,s))
    {std::istringstream iss(s);
        if(s.find("moveto")!= std::string::npos or
            s.find("lineto")!= std::string::npos or
            s.find("scale")!= std::string::npos or
            s.find("translate") != std::string::npos)
            iss >> cmds[i].x >> cmds[i].y >> cmds[i].cmd;
        if(s.find("rotate")!= std::string::npos)
            iss >> cmds[i].angle1 >> cmds[i].cmd;
        if(s.find("arc")!= std::string::npos)
            iss >> cmds[i].x >> cmds[i].y >> cmds[i].radius >>
            cmds[i].angleStart >> cmds[i].angleEnd >> cmds[i].cmd;
        i++;}
    N = i;for(int j = 0; j < i; j++)
    std::cout << cmds[j].cmd << std::endl; ifs.close();
}
```

- takes the postscript and saves each value or command into the struct

makeDrawWindow.cpp

```
#include "lab.h"
Fl_Cairo_Window * dw;

Fl_Cairo_Window * makeDrawWindow(){
    dw = new Fl_Cairo_Window(width,height);
    dw->label("Animate Graphics");
    dw->color(fl_rgb_color(74,189,172));
    return dw;}
```

- makes the drawing window

drawCB.cpp

```
#include "lab.h"
double f(double x){return x+20*sin(x);}
void drawCB(Fl_Cairo_Window* cw,cairo_t* cr)
{static double dx = 0; static double dy = 0; static double dr =
    for(int i = 0; i < N; i++){
    if(cmds[i].cmd == "moveto")
       cairo_move_to(cr,cmds[i].x,height−cmds[i].y);
    if(cmds[i].cmd == "lineto")
       cairo_line_to(cr,cmds[i].x,height−cmds[i].y);
    if(cmds[i].cmd == "arc")
        cairo_arc(cr,cmds[i].x,height−cmds[i].y,cmds[i].radius
        ,cmds[i].angleStart*(M_PI/180),cmds[i].angleEnd*(M_PI/180));
    if(cmds[i].cmd == "translate")
       cairo_translate(cr,dx + cmds[i].x,−(f(dx+cmds[i].x)));
    if(cmds[i].cmd == "scale")
        cairo_scale(cr,cmds[i].x,cmds[i].y);
    if(cmds[i].cmd == "rotate")
        {cairo_translate(cr,0,height);
            cairo_rotate(cr,dr + cmds[i].angle1*(M_PI/180));
            cairo_translate(cr,0,−height);}}
  dx+=5; if(dx>width−20) dx=−40; dy+=5; if(dy>height) dy=40;
  dr+=.1; cairo_stroke(cr);
}
```

- converts postscript to cairo
- animates video

main.cpp

```
#include "lab.h"
void animate(void*)
{
    dw->redraw();
    Fl::add_timeout(1.0/5,animate);
}
int main()
{
    getPSData("drawing.ps");
    parsePSData("drawing.ps");
    makeDrawWindow() -> show();
    dw->set_draw_cb(drawCB);
    Fl::add_timeout(1.0,animate);
    Fl::run();
}
```

- starts the program