PIC 10A

Lecture 6: Graphics

Why are we studying graphics?

For some time we are going to be using graphics libraries created by the authors of our book.

The books way of creating graphics is not the standard way.

Ok so why do we bother to study it?

- Its fun!
- We get a chance to do some other kind of output than text.
- It gives us a great way of learning how to use classes.
- Its a warm-up for real life graphics programming.

Some changes from the normal

Because we are using a non-standard library a few things will be different from the things we are used to.

For graphics we will not use the black console. Instead we will be using a graphics window.

The procedure for creating your C++ projects will be different. There is going to be a handout on this.

Instead of `cout` we will be using `cwin`.

Instead of `int main` we will have `int ccc_win_main`.

We will not need the line `using namespace std;`

Graphics boiler plate

```
#include "ccc_win.h"
int ccc_win_main()
{

   /* YOUR CODE HERE */

   return 0;
}
```

This will be very clear to you after you work through the graphics handout.

Graphics files

The graphics library files needed are on our class website page. You can fir them here:

http://www.math.ucla.edu/~virtanen/10a.2.13w/labs/lab2/ccc_graphics.zip

The notes for setting up a graphics project is here:

http://www.math.ucla.edu/~virtanen/10a.2.13w/labs/lab2/graphics.pdf

The graphics library

The graphics library has the following graphics features that we can use:

Point

Line

Circle

Message

These graphics features are really classes. Our aim today is to understand the member functions and member variables of these classes and start learning how to use them.

Point class

`Point P(x,y)`	Constructs a point P at location (x,y). x and y are doubles
`P.move(dx,dy)`	Moves the point P by (dx , dy). dx and dy are doubles
`P.get_y ( )`	Returns the y-coordinate of p.
`P.get_x ( )`	Returns the x-coordinate of p.

Point class example

Recall the two ways to construct classes:

1)

```
my_class my_obj(parameters);
```

2)

```
my_class my_obj = my_class(parameters);
```

Lets apply this to our Point class

Point construction

1)
```
my_class my_obj(parameters);
```

Example:

```
Point P(5.0,2.6);
cwin << P; // draws a point at location x=5.0 y=2.6.
```

2)
```
my_class my_obj = my_class(params);
```

This is the expression that we are interested in. The class on the right is not a variable. It is more like an expression.

Example:
```
cwin << Point(5.0,2.6);
```

Moving a point

Example:

```
double x_coord = 3.0 , y_coord = 2.0;
Point P(x_coord,y_coord);

cwin << P;

P.move(.5,-1);//Move the point to right by .5 down by 1

cwin << P;
```

Common mistake

What is wrong with the following?

```
Point P1(3.0,5.0);

Point P2 = P1.move(2.0,2.0);
```

Be careful! What data type does `P1.move` have?

Point example

Say we already have a `Point P` and we want to construct a `Point Q` that is 2 units to the right of `P` and 3 units up from `P`.

```
double x_coord = P.get_x();
double y_coord = P.get_y();

x_coord +=2;
y_coord +=3;

Point Q(x_coord,y_coord);
```

Or a slick way of doing all of it at once

```
Point Q(P.get_x()+2,P.get_y()+3);
```

Line class

`Line L(P , Point Q)`	Constructs a line L joining the points P and Q.
`L.get_start( )`	Returns the starting point of line L.
`L.get_end( )`	Returns the ending point of line L.
`L.move (dx,dy)`	Moves line L by (dx,dy). dx and dy are doubles

Line class example

Line class example

Circle class table

`Circle C(P,r)`	Constructs a circle with center P and radius r. P is a Point and r is a double
`C.get_center( )`	Returns the center point of circle C.
`C.get_radius( )`	Returns the radius of circle C.
`c.move (dx,dy)`	Moves center of the circle C by (dx,dy). dx and dy are doubles

Circle class example

Circle class example

Message class

`Message M(P,s)`	Constructs message M containing text s with starting point P. P is a Point and s is a string.
`Message M(P,x)`	Constructs message M containing number x with starting point P. P is a Point and x is a double.
`M.get_start( )`	Returns starting point of message M.
`M.get_text( )`	Returns text of message M.
`M.move(dx,dy)`	Moves message M by (dx,dy). dx and dy are doubles

Putting it all together

Prof Ryan's computer example