Lecture 6: Introduction to Graphics

PIC 10A

Todd Wittman

Why Study Graphics?

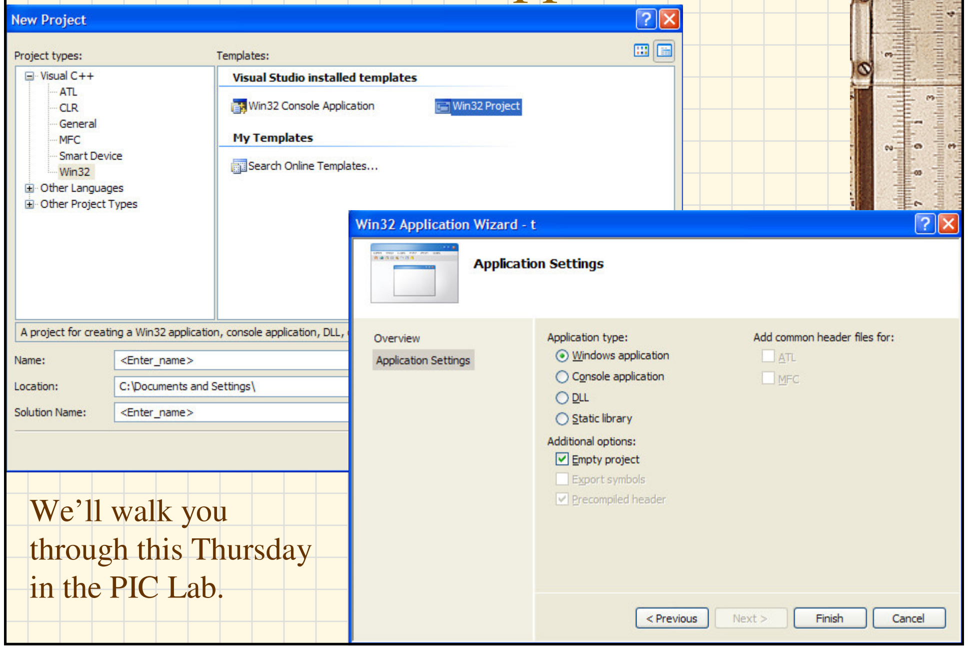
- For the next couple days, we're going to study computer graphics using a special library written by the textbook authors.
- This is not a standard or very useful library, so you probably won't ever see it again.
- C++ was designed for number crunching and data management, not graphics.
- So why are we studying graphics?
  - It's good practice with classes and objects.
  - It's a warm-up for real graphics programming.
  - It's fun.

Sec 3.4: Graphics Applications

- For most of this course, we deal with text-based console applications (cin / cout).
- For just one week, we're going to look at graphics applications which display simple shapes in the graphics **win**dow, not the console.
- So our commands will be quite different. For example, use cwin instead of cout.

```
cout << my_circle;    //Does not draw a circle.
cwin << my_circle;    //Draws a circle.
```

Not A Console Application!


New Project

Project types:
- Visual C++
  - ATL
  - CLR
  - General
  - MFC
  - Smart Device
  - Win32
- Other Languages
- Other Project Types

Templates:

Visual Studio installed templates
- Win32 Console Application
- Win32 Project

My Templates
- Search Online Templates...

A project for creating a Win32 application, console application, DLL,

Name: <Enter_name>

Location: C:\Documents and Settings\

Solution Name: <Enter_name>

Win32 Application Wizard - t

Application Settings

Application type:
- [✓ Windows application]
- Console application
- DLL
- Static library

Additional options:
- [✓ Empty project]
- Export symbols
- [✓ Precompiled header]

Add common header files for:
- ATL
- MFC

We'll walk you through this Thursday in the PIC Lab.

Starting a Graphics Application

- To use the graphics, we have to include the textbook's special library:

  # include "ccc_win.h"
- Start our graphics applications a little differently.

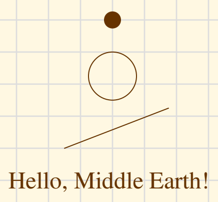
  Instead of: int main( )

  Start with: int ccc_win_main ( )
- We don't need the std namespace, because we're not using the standard libraries.

```
# include "ccc_win.h"
int ccc_win_main ( )  {
     ** YOUR CODE HERE **
     return 0;
}
```

Sec 2.8: The Graphics Classes

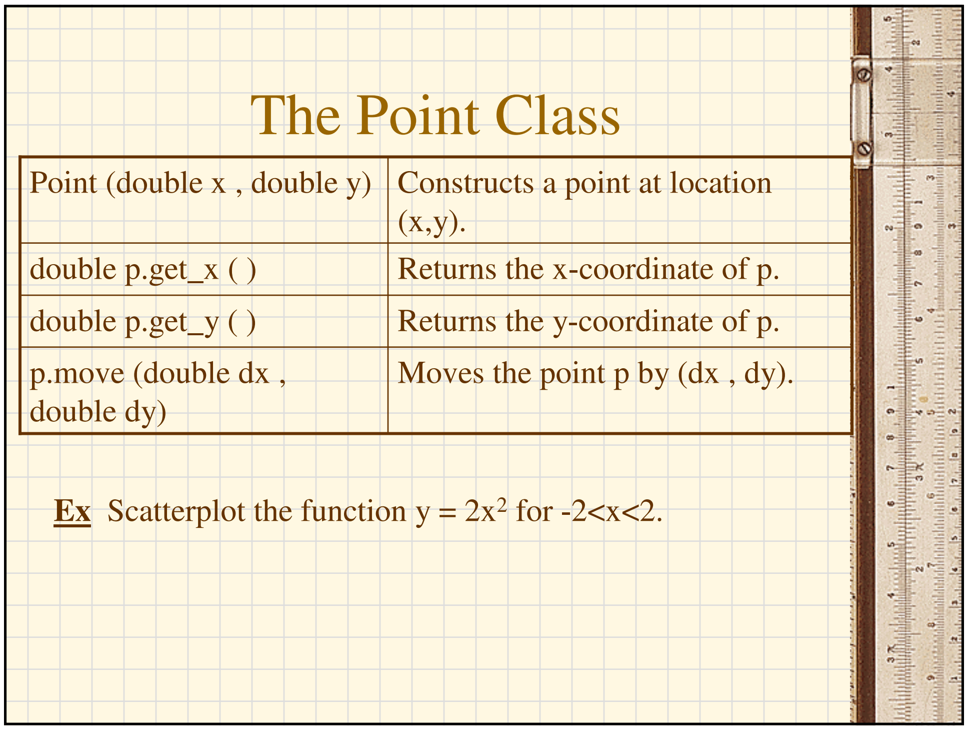
- The library "ccc_win.h" has 4 graphics classes for you to manipulate.
  - Point
  - Circle
  - Line
  - Message



- Any object created by these classes can be drawn on the screen by cwin.
- To draw a dot on the screen at (1,3), type

```
Point my_point (1 , 3);          cwin << Point(1,3);
cwin << my_point;           OR
```

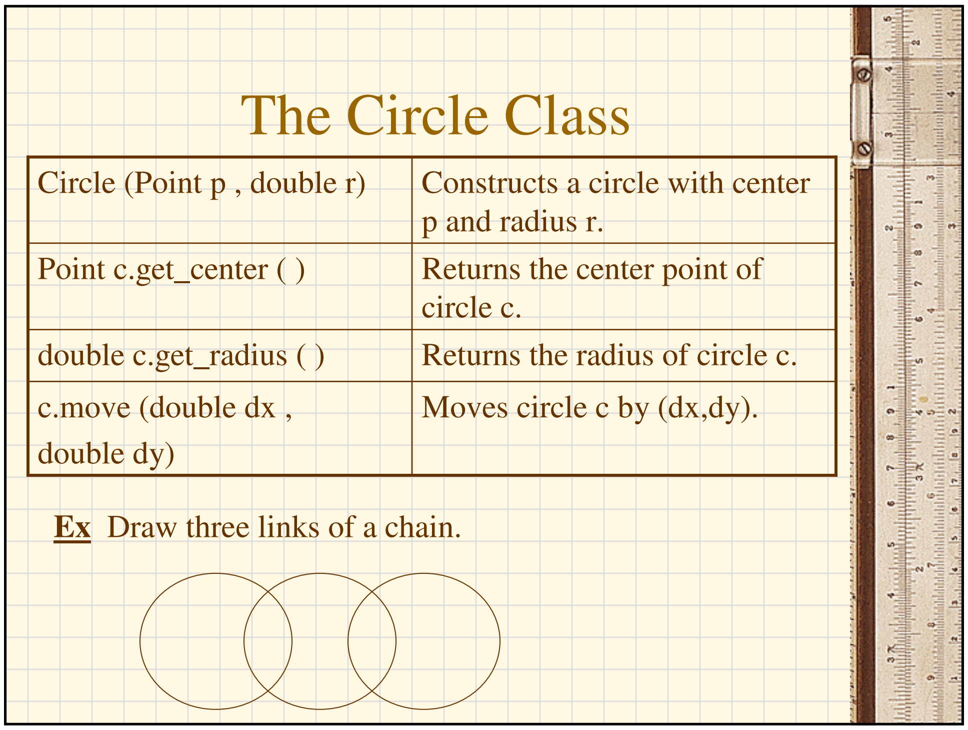
- Draw multiple objects with cwin << p1 << p2;



The Point Class

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

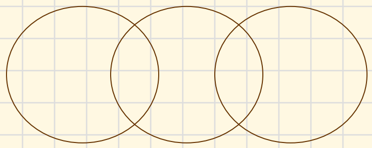
Ex Scatterplot the function $y = 2x^2$ for $-2<x<2$.

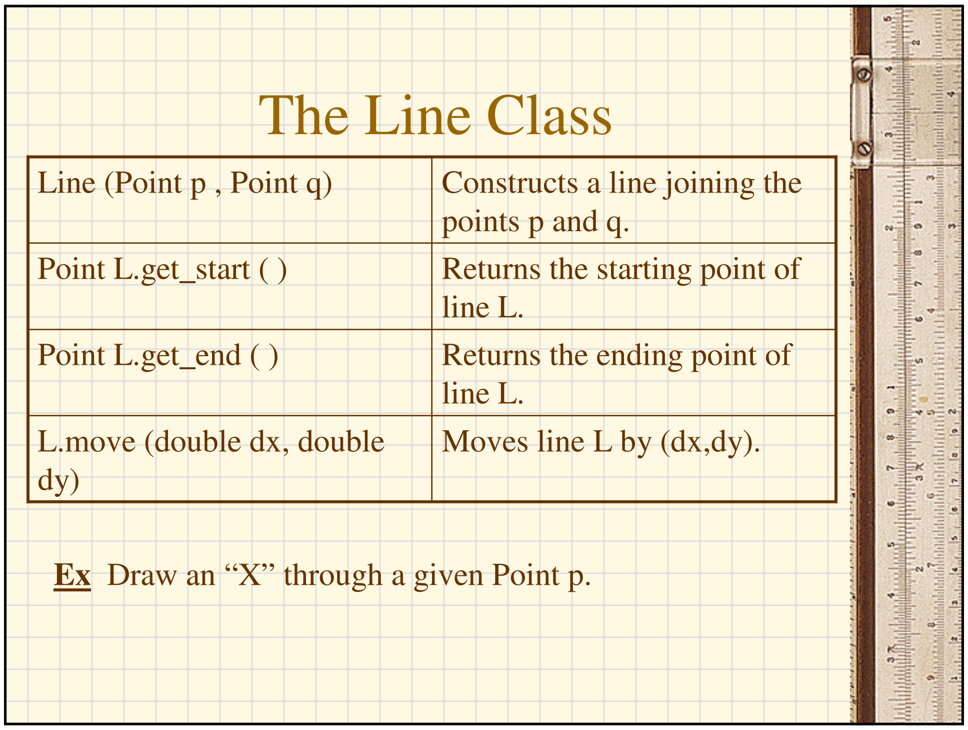


The Circle Class

Circle (Point p , double r)	Constructs a circle with center p and radius r.
Point c.get_center ( )	Returns the center point of circle c.
double c.get_radius ( )	Returns the radius of circle c.
c.move (double dx , double dy)	Moves circle c by (dx,dy).

Ex Draw three links of a chain.

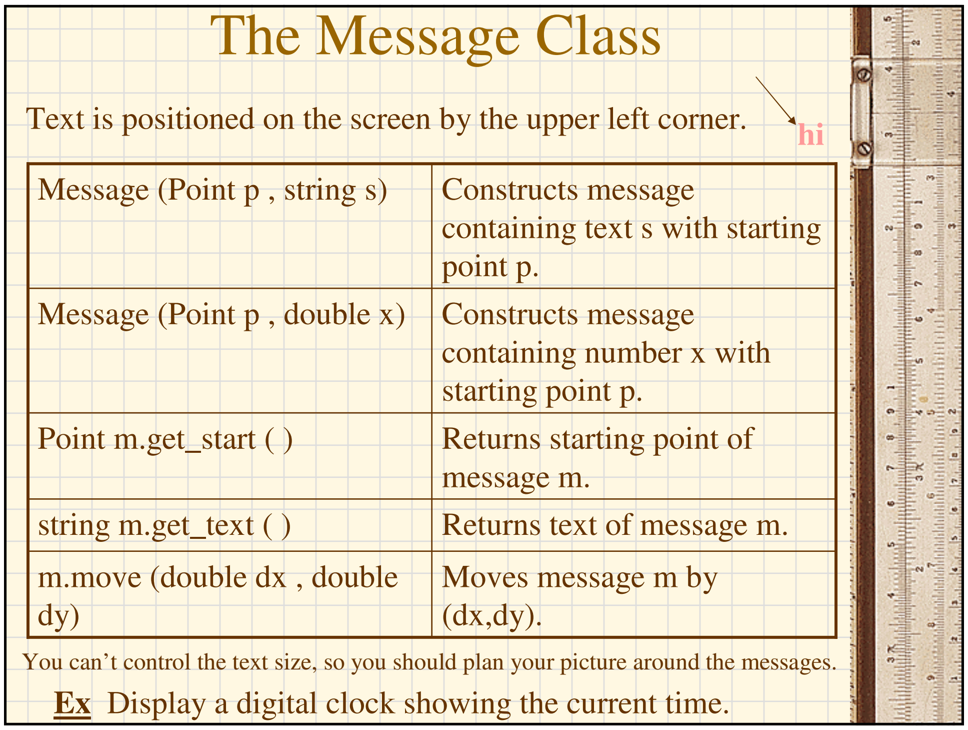




The Line Class

Line (Point p , Point q)	Constructs a line joining the points p and q.
Point L.get_start ( )	Returns the starting point of line L.
Point L.get_end ( )	Returns the ending point of line L.
L.move (double dx, double dy)	Moves line L by (dx,dy).

Ex Draw an “X” through a given Point p.



The Message Class

Text is positioned on the screen by the upper left corner. hi

Message (Point p , string s)	Constructs message containing text s with starting point p.
Message (Point p , double x)	Constructs message containing number x with starting point p.
Point m.get_start ( )	Returns starting point of message m.
string m.get_text ( )	Returns text of message m.
m.move (double dx , double dy)	Moves message m by (dx,dy).

You can’t control the text size, so you should plan your picture around the messages.

Ex Display a digital clock showing the current time.

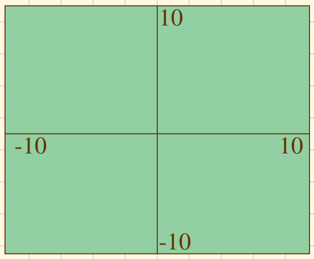
The Viewing Window

- We can adjust the viewing window with

  cwin.coord (x1, y1, x2, y2);

where (x1,y1) is the top left corner and (x2,y2) is the bottom right corner.

- The default window is -10<x<10, -10<y<10.



Ex Scatterplot the curve

$y = 100 \sin(x/3)$

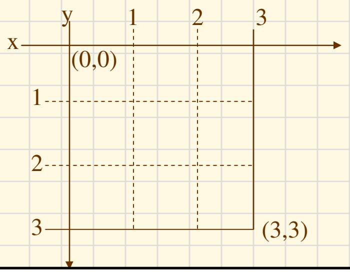
for 0<x<5.

Upside-Down Viewing Window

- We could enter an upper left corner that is lower than the bottom right corner. Huh?

  cwin.coord (0,0, 3,3);

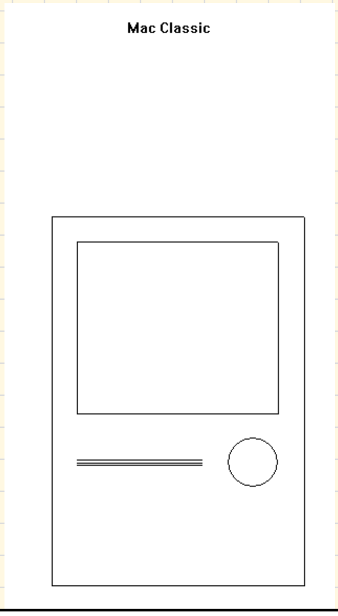
- This flips the y-axis, so that increasing the value of y moves us down.
- It looks a little strange, but this is actually the standard convention used in computer graphics and it is sometimes useful. *(See example on p. 101.)*



Prof. Ryan’s Computer

```
#include "ccc_win.h"
int ccc_win_main() {
    cwin.coord(0,0,3,3);
    Point t(1.3,0.2);
    Message words = Message(t,"Mac Classic");
    Point p(1,1);   Point q(2,1);    Point r(1,2.5);    Point s(2,2.5);
    Line L1 = Line(p,q);   Line L2 = Line(p,r);
    Line L3 = Line(r,s);    Line L4 = Line(s,q);
    Line m1(Point(1.1,1.1),Point(1.9,1.1));
    Line m2(Point(1.1,1.1),Point(1.1,1.8));
    Line m3(Point(1.1,1.8),Point(1.9,1.8));
    Line m4(Point(1.9,1.8),Point(1.9,1.1));
    Circle c(Point(1.8,2),0.1);
    d1(Point(1.1,1.99),Point(1.6,1.99));    Line d2(Point(1.1,2),Point(1.6,2));
    Line d3(Point(1.1,2.01),Point(1.6,2.01));
    cwin << L1 << L2 << L3 << L4 << m1 << m2 << m3 << m4 << c << d1 <<
    d2 << d3;
    cwin << words;
    return 0;
}
```

Prof. Ryan’s Computer


Mac Classic