# Computer Programming: Skills & Concepts (INF-1-CP)

## Intro to graphics with `descartes`

15th, 19th & 20th October, 2015

# Lab 4

- The `descartes` graphics routines.
- Example: Square-drawing example using `descartes` routines.

# descartes

`descartes` *is a set of small* functions *or* routines *which perform basic graphics tasks through a primitive graphics drawing tool.*

- What is a *function* (in programming)?
  It is an encapsulated and named section of code, which takes a number of parameters (or certain declared *types*), performs a sequence of C-statements, and returns a value of a declared *type*.

# descartes.h - structured data types

descartes.h contains *type* declarations for the (non-native) *structured data types* and *functions* of descartes.

```
/* A point is specified by its x- and y-coordinates. */

typedef struct {int x, y;} point_t;

/* A line segment is specified by its endpoints. */

typedef struct {point_t initial, final;} lineSeg_t;
```

Two special datatypes

- These are *structured* data types (notice `struct`) composed of more than one previously-defined data type.
- We will learn about `typedef` and `struct` in week 7 of CP.

# descartes.h - function prototypes

Function prototypes are *not* code . . . only describe "shape" of a function.

```
/* Waits until the user clicks the left mouse button, then
 * returns the point that the user input.  If the middle mouse
 * button is clicked the value returned * is (-1, -1). */

point_t GetPoint(void);

/* Creates a point with given coordinates. */
point_t Point(int a, int b);

/* Returns the x-coordinate of the point given as argument. */
int XCoord(point_t p);

/* Returns the y-coordinate of the point given as argument. */
int YCoord(point_t p);

/* Creates a line segment with given endpoints. */
lineSeg_t LineSeg(point_t p1, point_t p2);
```

# descartes.h - function prototypes cont'd

```
/* Returns one endpoint of a line segment... */
point_t InitialPoint(lineSeg_t l);

/* ... returns the other endpoint. */
point_t FinalPoint(lineSeg_t l);

/* Returns the length of a line segment. */
float Length(lineSeg_t l);

/* Draws a line segment. */
void DrawLineSeg(lineSeg_t l);

/* Opens and initialises the graphics window */
void OpenGraphics(void);

/* Closes the graphics window - (waits for right-mouse-click) */
void CloseGraphics(void);
```

# Function prototypes vs Function use

Consider the following *function prototype*:

```
/* Creates a point with given coordinates. */
point_t Point(int a, int b);
```

- This function prototype declares the "shape" of the `Point` function.
  - `Point` is a function which takes two input arguments (each of type `int`) and `returns` a result of type `point_t`.
- It tells us how we must *use/call* the function `Point`
  - `origin = Point(50, 100);`
    (must have already declared `origin` as a variable of type `point_t`)
  - `q = Point(XCoord(q), YCoord(q)+50);`
    (`p, q` must have already been declared as `point_t` variables)
- The actual code to implement `Point` is elsewhere - for `descartes` we "link" to Object code at compile time.

# Today's lab (Lab 4)

- Use the pre-programmed implementations of the functions of `descartes.h`. The code for these has been compiled already and that executable is available in `descartes.o`.
- You will need to download `descartes.h` (to use with `#include`) and `descartes.o` from the course webpage:

  `http://www.inf.ed.ac.uk/teaching/courses/cp`

  You will also find the example program `sqDraw.c` there.
- The file `lab4.tar` from the course webpage contains templates for the programs `segment.c`, `rectangle.c` and `polygon.c`:
  - Do not edit `descartes.h` or `descartes.o` *under any circumstances*!!
  - Your C programs for this lab should be written in `segment.c`, `rectangle.c` and `polygon.c`.

# line segments: `segment.c`

*Write a program which reads two points in the plane (specified as clicks on the graphics window), draws the line connecting these points, and calculates the distance between them.*

*Discuss:* Which functions from `descartes.h` will be useful?

# drawing a rectangle: `rectangle.c`

*Write a program which reads in two points from the plane (given as clicks on the graphics window), and then:*

(i) *draws the implied rectangle,*
(ii) *computes the length of its diagonal,*
(iii) *classifies the shape of the rectangle as* almost square, wide *or* tall*.*

*Discuss:* Which functions from `descartes.h` will be useful?

# drawing a polygon: `polygon.c`

*Write a program which reads in a sequence of points from the plane (given as clicks on the graphics window), and computes the perimeter of the polygon defined by those points.*

*Discuss:* Which functions from `descartes.h` will be useful?

# `descartes` example: Drawing a Square

*Write a program which uses the* `descartes` *functions to load the graphics window, read one point (specified by a click) from this window, and draw a square of side-length 100 which has this point as its North-West corner.*

*Which* `descartes` *functions will we need? Discuss.*
*What variables will we define?*

# Drawing a Square

Steps of our program:

- Start up the Graphics window.
- Read in a point from that window.
- Draw the 4 edges of the square.
- Close the graphics window.

# sqDraw.c - outline

```
#include <stdlib.h>
#include <stdio.h>
#include "descartes.h"

int main(void)
{
   point_t  p, q;     /*  Two point variables, */
   lineSeg_t pq;      /*  One line segment variable */
   int x, y;          /*  Two integers.  */

   OpenGraphics();    /*  Load graphics window. */
   printf("Indicate NW corner by clicking left mouse button.\n");
   p = GetPoint();    /* p stores point where the user clicked. */
   ........        /* Draw 4 line segs - LineSeg(,), DrawLineSeg(,) */
   CloseGraphics();
   return EXIT_SUCCESS;
}
```

# sqDraw.c

```
#include <stdlib.h>
#include <stdio.h>
#include "descartes.h"
/* Draws a square, of side 100, with given NW corner */

int main(void)
{
   point_t  p, q;    /*  Two points,         */
   lineSeg_t pq;     /*  a line segment      */
   int x, y;         /*  and two integers.   */
   OpenGraphics();

   printf("Indicate NW corner by clicking left mouse button.\n");
   p = GetPoint();   /* p stores the point where the user clicked. */
   x = XCoord(p);          /* We can take a point apart            */
   y = YCoord(p);          /* into its two coordinates...          */
   q = Point(x + 100, y);  /* and then reassemble.                 */
   pq = LineSeg(p, q);     /* Two points define a line segment.    */
   DrawLineSeg(pq);        /* Let's have a look at what we've got. */
```

# sqDraw.c cont'd

```
    p = q;                          /* Start where we left off.*/
    x = XCoord(p);
    y = YCoord(p);

    q = Point(x, y - 100);
    pq = LineSeg(p, q);
    DrawLineSeg(pq);

    /*  We can construct these shifted points more tersely... */

    p = q;
    q = Point(XCoord(p) - 100, YCoord(p));
    DrawLineSeg(LineSeg(p, q));

    p = q;
    q = Point(XCoord(p), YCoord(p) + 100);
    DrawLineSeg(LineSeg(p, q));

    CloseGraphics();
    return EXIT_SUCCESS;
}
```

# compiling and linking

We have *already* pre-compiled the `descartes.c` code; the "executable" for the `descartes` functions is in `descartes.o`.

In compiling your own graphics programs, you must "link" to this executable as follows:

```
gcc -Wall sqDraw.c descartes.o -lSDL -lm
```

The `-lSDL` is used because our `descartes` functions are themselves making use of the `SDL` graphics library.