CSCI 103: Introduction to Programming

Lab 9
Images - shapes

Lab Overview

- Goals
  - Learn to utilize 2D arrays and understand their indexing
  - Practice with image processing by completing a program that allows the user to draw rectangles and ellipses to a BMP image file

Background: 2D arrays

Declare by providing size along both dimensions and access with 2 indices

- Declaration: **unsigned char my_matrix[256][256]**
- Access: **my_matrix[128][128]**

Background: 2D arrays cont

The dimension order does not matter, but we normally treat the first index as **row** and the second index as **column**

- The [0][0] location is in the upper left-hand corner
- We use such layout in this lab

	Col. 0	Col. 1	Col. 2	...	Col. 255
Row 0:	[0][0]	[0][1]	[0][2]	...	[0][255]
Row 1:	[1][0]	[1][1]	[1][2]	...	[1][255]
Row 2:	[2][0]	[2][1]	[2][2]	...	[2][255]
...	...	...	...	...	...
Row 255:	[255][0]	[255][1]	[255][2]	...	[255][255]

The 2D array using [row][column] indexing. The first index is the row (top 0, bottom 255), and the second is the column (left 0, right 255).

Background: Passing 2D arrays

Formal parameter

- Must give dimensions of all but first dimension (and you may give that dimension if you want)

Actual Parameter

- Just the array name (i.e. still only passes the starting address)

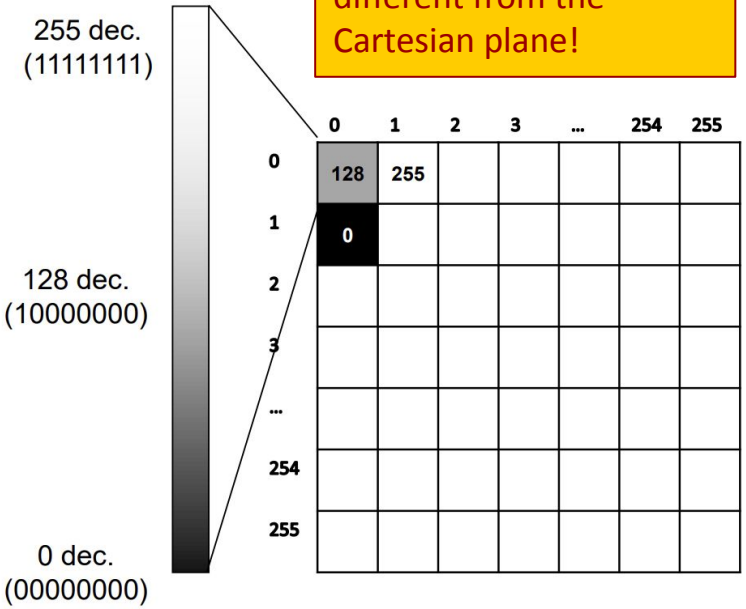
```
void writeImage(unsigned char outputImage[][256])
{
      ...
}

int main()
{
      unsigned char image[256][256];
      ...
      writeImage(image);
      return 0;
}
```

Background: Images

In this lab, we use a **256-by-256 2D array** to represent an image

- `unsigned char [256][256]`
- Each entry in the array represents 1 pixel
- The value of the pixel is a value of 0 – 255 where 0 is black and 255 is white

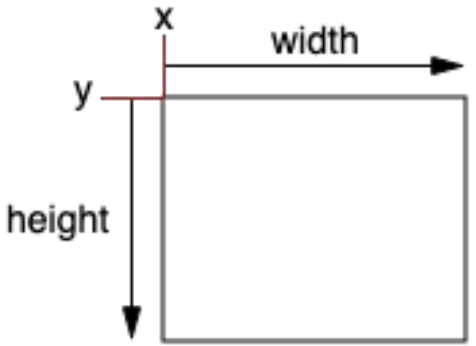
Note: the indexing is different from the Cartesian plane!



Background: Drawing rectangles

Takes 4 input values:

- Starting point (top row, left col)
- Height (#rows it should span)
- Width (#cols it should span)



Draw the sides of the rectangle with colour black (0)

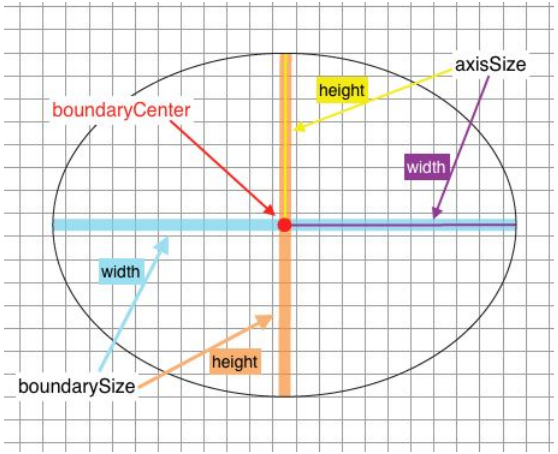
If any portion of the rectangle goes out of our 256,256 bounds, do not wrap around/crash, just don't draw it!

Background: Drawing ellipses

Takes 4 input values:

- Center point (cy, cx)
- Height (total max rows it spans)
- Width (total max cols it spans)

Eg (25 30 50 40) - an ellipse centered at 25,30, with total height 50 (25 each side) and total width 40 (20 each side)



Background: Drawing ellipses cont

Use Polar coordinates for Ellipse!

$x = r_x \cos\theta$

$y = r_y \sin\theta$

Where r_x is W/2 and r_y = is H/2.

And vary θ from 0 to 2*Pi in small increments, dθ, (for this lab use dθ = `0.01`) and apply the conversion to rectangular coordinates

If any pixel of the ellipse border would be out of the image's bounds, just don't draw it (to avoid wrapping or crashing)!

Background: Drawing ellipses cont

Use Polar coordinates for Ellipse!

x=center_col + $r_x \cos\theta$

y=center_row + $r_y \sin\theta$

Where r_x is W/2 and r_y = is H/2.

And vary θ from 0 to 2*Pi in small increments, dθ, (for this lab use dθ = `0.01`) and apply the conversion to rectangular coordinates

If any pixel of the ellipse border would be out of the image's bounds, just don't draw it (to avoid wrapping or crashing)!

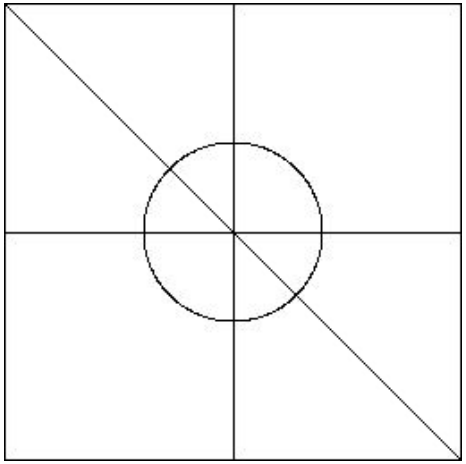
File Structure & Compilation

Included and ready are the files:

- **`bmplib.cpp`**: has `writeGSBMP()` method implemented to create the output '.bmp' file for the image arr
  - `int writeGSBMP(const char filename[], unsigned char outputImage[][SIZE])`
  - check demo.cpp for usage example
- **`demo.cpp`**: an example code that creates `cross.bmp`
- **`Makefile`**: run 'make' to create the executables (./shapes, will also create ./demo)

Demo Program

- Creates the image as shown
  - First creating the central horizontal and vertical lines, then the diagonal and finally the circle by changing the respective indices in the image array to 0
- Calls *writeGSBMP* to create `cross.bmp` output file
- Run `./demo` to see the output



Your Tasks

Shapes.cpp:

- Complete the required implementation of the `draw_rectangle()` and `draw_ellipse()` functions as described
- In `main()`: add logic to take user input and then appropriately call the respective draw_function

Checkoff

- This is an **ungraded lab**. Just enjoy and have fun coding! No checkoff is necessary