University of Western Ontario, Computer Science Department
CS3388B, Computer Organization

Assignment 2 **Due: Thursday, February 2, 2022**

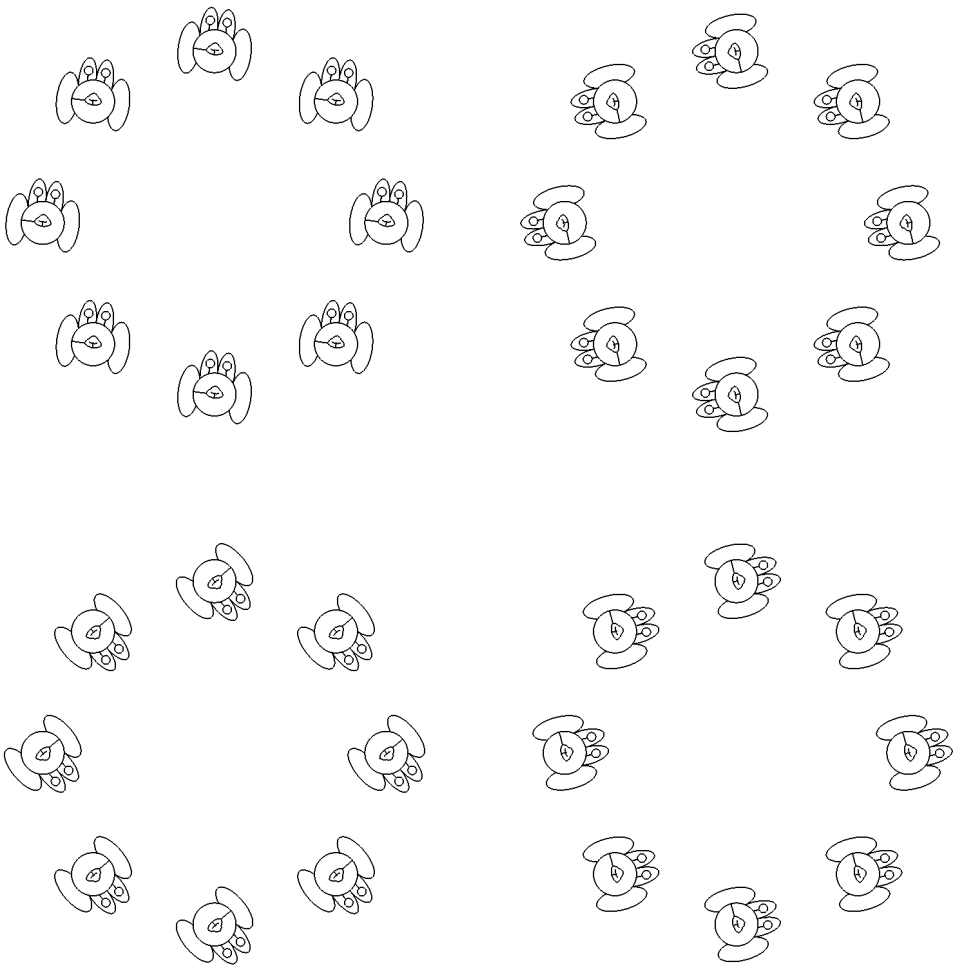
General Instructions: This assignment consists of 4 pages, 2 exercises, and is marked out of 100. For any question involving calculations you must provide your workings. *Correct final answers without workings will be marked as incorrect.* Each assignment submission and the answers within are to be solely your individual work and completed independently. Any plagiarism found will be taken seriously and may result in a mark of 0 on this assignment, removal from the course, or more serious consequences.

Submission Instructions: The answers to this assignment (code, workings, written answers) are to be submitted electronically to OWL. Ideally, any workings or written answers are to be typed. At the very least, clearly *scanned* copies (no photographs) of hand-written work. If the person correcting your assignment is unable to easily read or interpret your written answers then it may be marked as incorrect without the possibility of remarking. Include a **README** with any code.

Exercise 1. In this exercise we will explore transformations and poly-line drawings. On OWL you will find a `dog.txt` file. This is a "space-separated" file of floating point numbers. In this file, the ith number and the $i+1$th number encode the (x, y) position of a vertex. When these vertices are drawn in sequence using a poly-line (line strip), an image is formed.

In this exercise you are to create an OpenGL program which reads in this data from the file and repeatedly draws this poly-line image. Your program should follow these specifications:

- Specifies the viewing volume to be $(left, right, bottom, top) = (0, 60, 0, 60)$
- Set the background to white.
- Around a circle of radius 25, centered at $(30, 30)$, draw, using immediate mode, the dog poly-line image at 8 different points along the circle: 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. Use black as the color.
- For each *frame* that is drawn, rotate the dogs counterclockwise 1° about their point on the circle of radius 25. Each dog should appear to "spin" in place. Consider one frame as one iteration of the loop that ends with `glfwSwapBuffers(window)`.



Exercise 2. In this exercise we will draw a *dot plot* using a very simple algorithm. I give it below in pseudo-code.

1. Let (x_{min}, y_{min}) and (x_{max}, y_{max}) define the opposite corners of a square.
2. Choose a random corner of the square c_0 (remember squares have four possible corners.)
3. Choose a random point p_0 (x, y) such that $x_{min} < x < x_{max}$ and $y_{min} < y < y_{max}$. for i from 1 to N:
   (a) Choose a random corner c_i of the square which is *not* diagonally opposite to c_{i-1}.
   (b) Let p_i be the point halfway between p_{i-1} and c_i.
   (c) Draw a point at p_i.

In this exercise, create an OpenGL program which follows the above pseudo-code to draw a dot plot. Your program should behave as follows:

- As command line arguments, your program takes three parameters: N, screen width, screen height.
- Use `glOrtho` to set the viewing volume to $(left, right, bottom, top) = (-1.1, 1.1, -1.1, 1.1)$
- Set the background to white.
- Your code executes the above algorithm to draw N points in black, and using $x_{min} = y_{min} = -1$, $x_{max} = y_{max} = 1$. Use `glPointSize(2.0f)`.
- Unlike in previous examples, do *not* draw the points *inside* of the `while (!glfwWindowShouldClose(window))` loop. Use immediate mode to draw all the points *outside* the loop, call `glfwSwapBuffers(window)` once, and then enter the loop to keep the window open.
- Your program should support N as large as 5,000,000.

To help, here's a function you can use to create random floats in the range $[-1, 1]$.

```
static inline double frand() {
    double x = ((double) rand()) / (double) RAND_MAX;
    if (rand() % 2) {
        x *= -1.0;
    }
    return x;
}
```

If correct, your program should look like this: