CS 106 Winter 2019

Lab 04: Advanced Shapes

Due: Wednesday, January 30th, 11:59pm

Summary

This lab will allow you to practice advanced shapes. Each question is on a separate page.

SAVE each sketch as “L04Q01”, “L04Q02, “L04Q03”, etc.

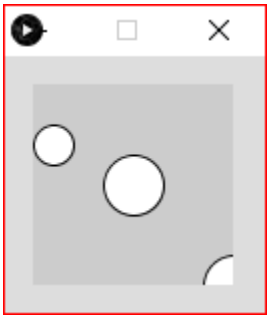
QUESTION ONE

You are given the starter code below.

Write a function called `circle()` that takes three floating-point numbers as arguments, representing the *x* and *y* coordinates of the centre of a circle and a radius for that circle, and draws the circle. The function should not return a value.

```
void setup() {
  circle(10, 30, 20);
  circle(width / 2, height / 2, 30);
  circle(width, height, 30);
}
```

With the above starter code and your function `circle()`, the result should look like the following.



Note: Your function `circle()` only needs one line of code.

Save your solution as L04Q01.

QUESTION TWO

You are given the starter code below.

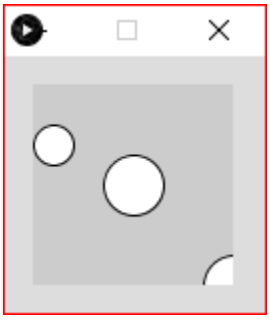
As an extension to question 1 above, write a second version of `circle()` that takes two arguments: a single `PVector` containing the centre of the circle, and a radius for that circle, and draws that circle. The function should not return a value.

```
void setup() {

  PVector p1 = new PVector(10, 30);
  PVector p2 = new PVector(width / 2, height / 2);
  PVector p3 = new PVector(width, height);

  circle(p1, 20);
  circle(p2, 30);
  circle(p3, 30);
}
```

With the above starter code and your function `circle()`, the sketch should look like the following.



Note: Your function `circle()` only needs one line of code.

Save your solution as L04Q02.

QUESTION THREE

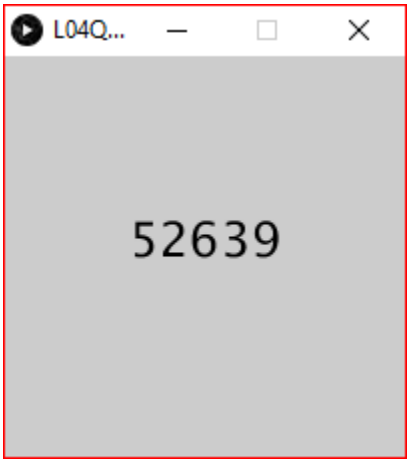
You are given the starter code below.

Write a function `secondsToday()` that takes no arguments. It uses the built-in functions `hour()`, `minute()` and `second()` to return an integer telling you how many seconds have elapsed since midnight. So, if the function were called at exactly 2:37.19pm, it would return `14*60*60 + 37*60 + 19 = 52639`.

```
void setup() {
  size(200, 200);
  fill(0);
  textSize(24);
  textAlign(CENTER);

  int result = secondsToday();
  text(result, width / 2, height / 2);
}
```

With the above starter code and your function `secondsToday()` the result should look like the following.



QUESTION FOUR

You are given the starter code below.

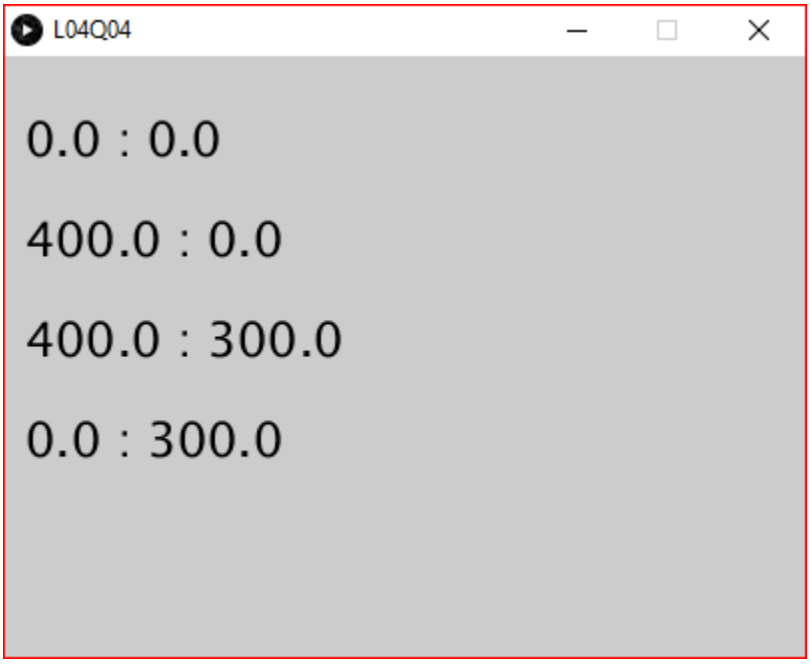
Write a function `fourCorners()`. It takes no arguments, and returns an array of four `PVector` objects, representing the four corners of the sketch window. For example, if `setup()` contains the call `size( 400, 300 );`, then `fourCorners()` might return the array `{new PVector(0,0), new PVector(400,0), new PVector(400,300), new PVector(0,300)}`.

```
void setup() {
  size(400, 300);
  fill(0);
  textSize(24);
  PVector[] cnr = new PVector[4];

  cnr = fourCorners();

  text(cnr[0].x + " : " + cnr[0].y, 10, 50);
  text(cnr[1].x + " : " + cnr[1].y, 10, 100);
  text(cnr[2].x + " : " + cnr[2].y, 10, 150);
  text(cnr[3].x + " : " + cnr[3].y, 10, 200);
}
```

With the above starter code and your function `fourCorners()` the result should look like the following.

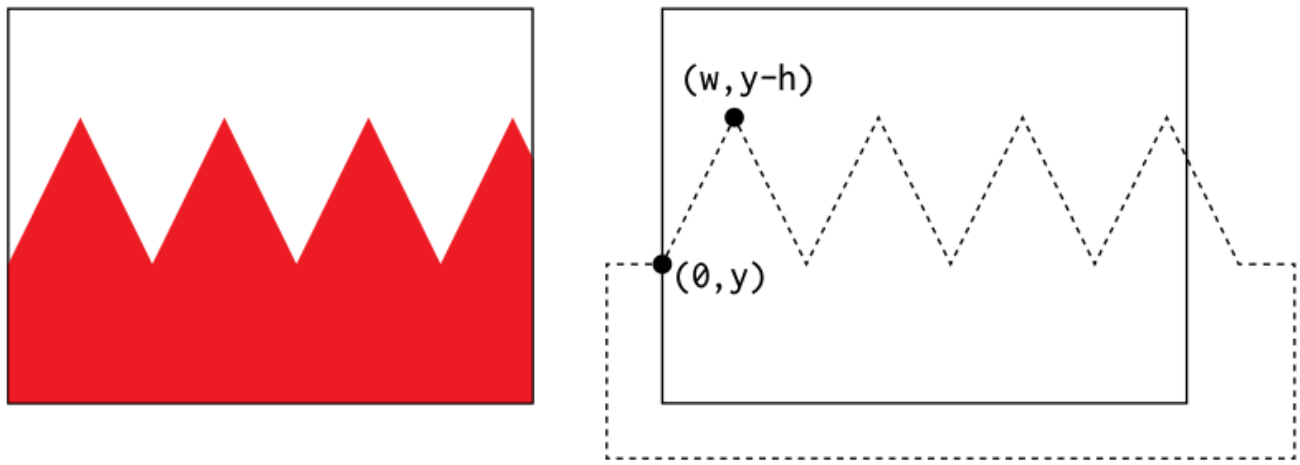


QUESTION FIVE: ZigZags

Write a function called `zigzag()` that can draw zig-zag shapes according to a few different parameters. Use `beginShape()`, `vertex()`, and `endShape()` to specify the path. The `zigzag()` function shouldn't have any other function calls in it (if you want to add fill or stroke colours to the zig-zag, do it in the place where this function gets called). It can have other things in it, like local variables and loops.
In particular, your function should look like this (you can copy this code into your sketch as a starting point):

```
void zigzag( float y, float w, float h )
{
  // TODO: Fill this in
}
```

The following diagram illustrates the shape of the complete path you should draw to construct your zig-zag pattern:



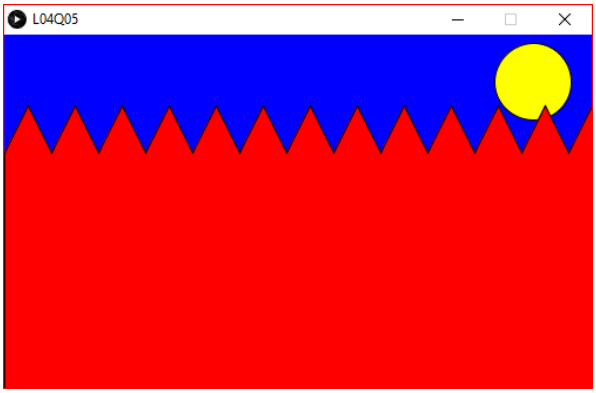
The parameters `y`, `w`, and `h` control the shape and location of the zig-zag. It always starts at the point `(0,y)` on the left edge of the sketch window. Then, it repeatedly steps to the right by distance `w` until it crosses the right edge of the sketch. The steps alternate moving vertically up and down by distance `h`. Thus the value of `w` controls the density of zig-zags across the sketch window, and `h` controls their heights.

In order to build a closed path that can be filled with a solid colour, when you pass the right edge of the sketch window, add vertices that drop down below the bottom of the sketch, overshoot the left edge, and then come back up to join to the first point in the zig-zag. You'll end up with a path that looks like the dotted line above.

Once your `zigzag()` function is written, write a `setup()` function that creates a sketch window of size at least 400×400, and draws a picture using zig-zags as a prominent part of the composition. You can include other graphics if you want, but your picture must have at least three zig-zags in it, all visible, and filled with at least three different colours. Examples are shown below.

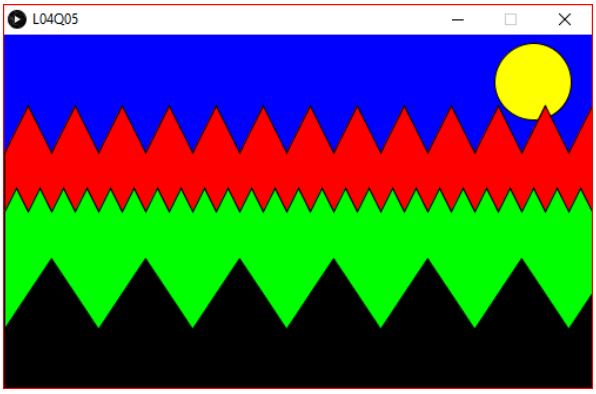
If we have the following starter code, the then the result is as shown below.

```
void setup() {
  size(500, 300);
  background(0, 0, 255); // blue sky
  fill(255, 255, 0);
  ellipse(width-50, 40, 65, 65); // yellow sun
  fill(255, 0, 0); // red zigzag
  zigzag(100, 20, 40);
}
```

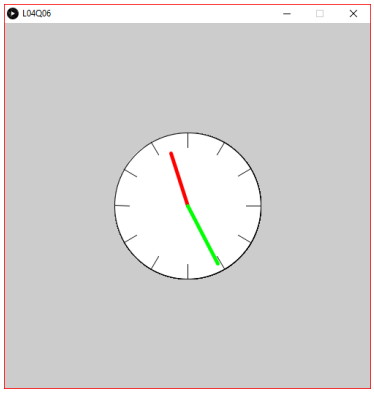


If we have the following starter code, the then the result is as shown below.

```
void setup() {
  size(500, 300);
  background(0, 0, 255); // blue skye
  fill(255, 255, 0);
  ellipse(width-50, 40, 65, 65); // yellow sun
  fill(255, 0, 0); // red
  zigzag(100, 20, 40);
  fill(0, 255, 0); // green
  zigzag(150, 10, 20);
  fill(0); // black
  zigzag(250, 40, 60);
}
```



QUESTION SIX: Analog Clock



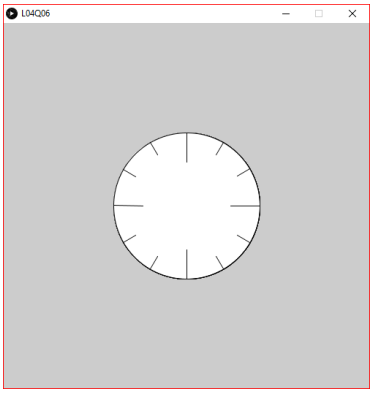
Create a working analogue clock sketch that displays the current time. The sketch window should have size at least 500×500. It should show 12 tick marks around the clock face, with the marks for 12, 3, 6 and 9 o'clock drawn more prominently (e.g., longer). There should be a minute hand and a shorter hour hand that show the current time. The hour hand should be offset by the number of minutes after the hour, so that it sweeps smoothly from one hour to the next.

Use the built-in functions `hour()` and `minute()` to ask Processing for the current time. Look at the documentation for these functions if you need to.

Draw the minute hand as a line from the centre of the clock to the edge. Use the current minutes after the hour to determine what angle to use with the `polar()` function. Remember that the sketch's coordinate system is "upside-down", so you may need to fiddle with angles a bit, and/or turn some numbers into their negatives.

Draw the hour hand as a line from the centre of the clock to a point not quite at the edge. Use the current hour to determine the angle, but also add in a fractional amount of hour based on the current minutes. For example, at 11:26, the hour hand should point halfway between 11 and 12 (as shown above); at 7:23, the hour hand should point to a position 23/60 of the way from 7 to 8.

Starter code is provided on the next page. The starter code produces the clock face as shown here. The polar() function is also provided below.



The polar() function.

```
PVector polar(float r, float theta)
{
  float dt = radians(theta);

  float x = r * cos(dt);
  float y = r * sin(dt);

  return new PVector(x, y);
}
```

Starter code to draw the clock face.

```
//  Draw an analog clock. Starter code.

int count = 0;
int shortLineStart = 80;
int longLineStart = 60;
int ticksOnClock = 12;

void setup() {
  size(500, 500);
  frameRate(1);
}

void draw() {

  // Draw the clock face.
  fill(255); // white
  ellipse(width / 2, height / 2, 200, 200);

  // Draw the 12 ticks on the clock using line()
  //    with longer ticks for 3, 6, 9, and 12 o'clock.
  stroke(0); // black
  for (int i = 0; i < ticksOnClock; i++) {
    int startLine;
    if (i % 3 == 0) {
      startLine = longLineStart;
    } else {
      startLine = shortLineStart;
    }
    // pv1 is the start of the line.
    // pv2 is the end of the line.
    PVector pv1 = polar(startLine, i * 30);
    PVector pv2 = polar(100, i * 30);
    stroke(5);
    line(width / 2 + pv1.x, height / 2 + pv1.y,
      width / 2 + pv2.x, height / 2 + pv2.y);
  }
```

Submission

Submit all sketch directories from this lab as one ZIP file called L04.zip to the lab dropbox on Learn.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0.