**University of London**
**Creative Computing**
**CO1112 Creative computing I: image, sound and motion**
**Coursework assignment 2 2018–19**

## Aims

This coursework assignment aims to provide you with practical experience of designing and implementing code with user interaction and motion. It is also designed to allow you to explore how the various techniques you have learned during the course can be effectively combined to create an engaging computer game. Finally, it provides a chance for you to practise describing your design concepts in writing, and reflecting on the strengths and weaknesses of the work you have developed.

## Citation and referencing

It is important that your submitted assignment is your own individual work and, for the most part, written in your own words. You must provide appropriate in-text citation for both paraphrase and quotation, with a detailed reference section at the end of your assignment (this should not be included in the word count). Copying, plagiarism and unaccredited and wholesale reproduction of material from books or from any online source is unacceptable, and will be penalised (see: How to avoid plagiarism).

## Introduction

As a starting point for this coursework assignment, the following code is provided (it is based upon one of the questions in last year's examination). The code can be downloaded from the CO1112 page on the VLE (the file is named `SticksAndStones.pde`). Study the code to understand what it does. Download and run the code to see it in action.

```
1  int holeX;
2  int ballX, ballY;
3  int ballInc=2;
4  int ballDiameter=30;
5  int barHeight=50;
6  int screenSize=600;
7  int score=0;
8
9  void settings()
10 {
11   size(screenSize,screenSize);
12 }
13
14 void setup()
15 {
16   frameRate(30);
```

```
  noStroke();
  textSize(32);
  resetPositions();
}

void draw()
{
  background(0, 0, 255);

  // write score
  fill(255, 255, 255);
  text(score, screenSize/2, 40);

  // draw bar
  fill(255, 0, 0);
  rectMode(CORNER);
  rect(0, screenSize/2 - barHeight/2,
screenSize, barHeight);

  // draw hole in bar
  fill(0, 0, 255);
  rectMode(CENTER);
  rect(holeX, screenSize/2, barHeight,
barHeight);

  // draw ball
  fill(255, 255, 255);
  ellipse(ballX, ballY, ballDiameter,
ballDiameter);

  // detect collisions between ball and bar
  if (collisionDetected())
  {
    resetPositions();
  }

  // update ball position
  ballY += ballInc;

  // update score and reset if necessary
  if (ballY >= screenSize - ballDiameter/2)
  {
```

```
    score++;
    resetPositions();
  }
}

// reset the ball to a random position at
the top of the screen
void resetPositions()
{
  holeX=screenSize/2;
  ballX=(int)random(screenSize);
  ballY=0;
}

// handle keyboard input
void keyPressed()
{
  if (key == 'S' || key == 's')
  {
    holeX += 5;
  }
  else if (key == 'A' || key == 'a')
  {
    holeX -= 5;
  }
}

boolean collisionDetected()
{
  boolean collision = false;

  // TO DO (PART A): IMPLEMENT THIS METHOD!
  // (set the collision variable to true if
a collision is detected)

  return collision;
}
```

**Part A**

To complete this part of the coursework assignment, create a copy of `SticksAndStones.pde` named `PartA.pde`, and place it in a directory called `PartA`. The zip file you submit for assessment should include the `PartA` directory with the `PartA.pde` file within it.

The provided code forms the basis of a simple game. The idea is that the user moves the hole in the central bar left or right (using the 'A' and 'S' keys) to allow the ball to drop through the hole without hitting the bar. The code to check whether the ball has hit the bar is incomplete.
Complete the `collisionDetected()` function in `PartA.pde` to implement collision detection between the ball and the bar. If a collision is detected, your code should assign the value `true` to the variable named collision within this function.

Think carefully about all of the conditions under which a collision should be detected. For the purposes of this function, you may assume that the ball is a square with side length equal to the diameter of the ball (*i.e*. the collision detection does not need to take into account the curvature of the ball).

Include comments in your code to explain exactly what each part of your collision test is checking for.

In your submitted PDF file, write a short paragraph for Part A stating whether or not you believe the collision detection is working correctly. Describe any problems you encountered when implementing the function, and how you tried to overcome them (whether or not you were successful).

**[15 marks]**

**Part B**

To complete this part, create a copy of your `PartA.pde` file named `PartB.pde`, and place it in a directory called `PartB`. The zip file you submit for assessment should include the `PartB` directory with the `PartB.pde` file within it.

Extend the code so that it draws an additional bar, 75 pixels below the existing one. The new bar should also have a hole in it, but when the game begins the hole should be drawn at a random position along the x axis. Your new code should implement keyboard interaction for the second bar using the 'Z' and 'X' keys to move the hole left and right, respectively. You should also implement collision detection code for the new bar.

Think carefully about how your new code should be designed. Ensure that your code is well commented to explain the purpose of the new code.

This part of the coursework assignment is worth a maximum of 20 marks. A maximum of 10 marks is available for a straightforward extension of the existing code. To obtain more than 10 marks you will need to refactor the code using an object oriented design. For further information about object oriented programming in Processing, see https://processing.org/tutorials/objects/.

In your submitted PDF file, write a short paragraph for Part B describing how you approached this part. Describe any problems you encountered, and how you tried to overcome them (whether or not you were successful).

**[20 marks]**

**Part C**

To complete this part, create a copy of your `PartB.pde` file named `PartC.pde`, and place it in a directory called `PartC`. The zip file you submit for assessment should include the `PartC` directory with the `PartC.pde` file (and any additional required files and subdirectories) within it.

For this part of the coursework assignment, we would like you to extend the code you have developed in Parts A and B to produce a more exciting and engaging game experience. We expect you to use some of the techniques you have learned during the CO1112 course when doing this, but the choice of exactly what you do is yours. For example, you could include texture maps, sound, more complicated movements, or anything else you have learned about. Your extensions might, for example, improve the user's aesthetic experience of the game (*e.g*. by adding sound or graphics) or introduce more engaging gameplay (*e.g*. by adding different levels of difficulty). The choice is yours.

Importantly, you should have a clear idea of *why* you are adding each new feature and how it will improve the user's experience of the overall game.

In your submitted code directory be sure to include all files needed to run your program (*e.g.* including any image or sound files required).

In your submitted PDF file, write a description (around 500 words) of what you have attempted to achieve in Part C, and what techniques you have used. Also discuss any problems you encountered when developing the code, and how you attempted to overcome them (whether successfully or not). You should plan to spend at least 10 hours on this part of the coursework assignment. Your work will be assessed according to a variety of factors, including: the ambition of what you have tried to achieve, the effectiveness of the end product, the quality of your code (including in-code comments), and your written description of what you have done.

**[50 marks]**

**Part D**

In your submitted PDF file, write a self-assessment of what you achieved in Part C. Consider the following questions in your answer:

- Were you successful in achieving what you had originally planned to do?
- How did the end result meet your expectations? Was it as good as you thought it would be? Better? Worse? Are there any aspects that you think work particularly well? Are there aspects that you are not so happy with?
- In what ways do you think the game could be improved in future work?

When answering these questions, be sure to explain and justify your answers. We are looking for realistic and insightful answers about what you achieved, no matter how successful you were in Part C. Your answer should be around 500 words.

**[15 marks]**

**[Total: 100 marks]**

**Submit the following by uploading to the VLE:**

1. A single PDF file containing the written discussion requested for each part of the coursework. The file should be named `FamilyName_SRN_CO1112cw2.pdf` (replacing "FamilyName" with your own name, and "SRN" with your 9-digit student registration number).

2. A single zip file named `FamilyName_SRN_CO1112cw2.zip`, containing the following:
   (a) A directory named `PartA`, containing your file `PartA.pde`.
   (b) A directory named `PartB`, containing your file `PartB.pde`.
   (c) A directory named `PartC`, containing your file `PartC.pde` and any other files required to run your code for Part C.

Before submitting your work, be sure to test your submission by unzipping your zip file into a clean directory and checking that the code for each of Parts A, B and C runs correctly in Processing without any need to rename or move files.

**[END OF COURSEWORK ASSIGNMENT 2]**