Project 2: Graphics Simulation

- Project starts on: **Saturday, February 20th**
- Project is due on: **Monday, March 7th at 23:59**
- This is an individual project

**IMPORTANT:** When you submit your project, PLEASE CHECK to ensure that your submission has been accepted. If you do not do this, and the submission fails, a 5% penalty will be applied.

You are allowed to discuss about the project in a high level manner with your classmates. However, you cannot show your code to other students. We will use MOSS before grading this project. If a plagiarism or cheating is detected, you will be reported to Dean of Students.

**Project Objectives**

1. Utilize the Graphics library to create animation.
2. Practice decision structures, loops, functions, and lists.
3. Learn how to use Unit Tests to verify the correctness of your project subparts.

**Demo Video**

In Project 2, you will be using the Graphics Library to create a simple game with three moving objects. The game starts with three objects (a square, a circle, and a rectangle) placed on the bottom of the game canvas. On the first mouse click, the objects move vertically to the top and bounce back off the window's boundary. Here's a short video demonstration of the game that you are going to implement in this Project.

https://www.youtube.com/watch?v=A50kJMZ2iAw&feature=youtu.be

**What is a Unit Test?**

Unit Testing is the practice of independently testing parts of your programs such as functions. It gives
programmers the ability to verify that functions work as expected. That is to say, for any given function and a given set of inputs, a Unit Test validates whether or not the function is returning the proper values.

**In this Project, you'll be given Unit Tests to help you validate the functionality of your program as your progress.** You may add the test function's code to your project and invoke the test independently from the IDLE prompt. Note that the Unit Tests are not added to the project skeleton. You would need to add the test to your project, verify the functionality and correctness of your code, and then remove the test.

Setting up the environment

Go to your working directory in “data.cs.purdue.edu” and create the directory “cs177/project2”. Refer to the lab01 if you need to remember the steps to do so. Make sure that `graphics.py` is saved in your `project2` directory.

Your project skeleton is [here](http://courses.cs.purdue.edu/). Download this file and start working on it!

**Task 1: Creating the Game Canvas**

In this task, you will implement a `createGameCanvas` function that creates a window of size 500×500 and sets the background to brown. *This function receives no parameters, and returns a window object.*

**Task 2: Initializing Score and Clicks labels**

In this task you will implement a `initializeScoreLabels` function that creates four Text Object labels and places the objects on the game canvas:

- `scoreLabel`: displays the string “Score” at location: (30,10).
- `scoreValueLabel`: displays the game score, initialized with “0”, at location: (60,10).
- `clicksLabel`: displays the string “Clicks” at location (30,30).
- `clicksValueLabel`: displays the number of clicks, initialized with “0”, at location: (60,30).

Finally, set the color of all text labels to yellow.

*The function receives a window object as a parameter and returns two text objects: scoreValueLabel and clicksValueLabel.*

**Unit Test**

test1 invokes `createGameCanvas` and `initializeScoreLabels` to create the game canvas and labels.

```python
def test1():
```

http://courses.cs.purdue.edu/
Task 3: Adding Shape Objects to Game Canvas

In this task, you will implement a `drawShapes` function that creates three objects: a square, a circle and a rectangle. The function places the objects on the game canvas as follows:

- Square: corner points at (100, 460) and (140, 500). Colored red.
- Circle: center point at (250, 475) with radius 25. Colored blue.
- Rectangle: corner points at (400, 425) and (440, 500). Colored green.

_The function receives a window object as a parameter and returns a list of three shape objects: a square, a circle and a rectangle._

**Unit Test**

test2 builds on test1 and invokes `drawShapes` in order to place shapes objects on the game canvas.

```python
def test2():
    win = createGameCanvas()
    scoreLabelValue, clicksValueLabel = initializeScoreLabels(win)
    shapesList = drawShapes(win)

>>> test2()
```

Expected output:
Task 4: Check for a Click in the Game Canvas

In this task, you will implement a `checkForClick` function that checks whether a click has occurred on the game canvas. In the case of a click, the function will then determine whether or not the click occurred inside any of the moving objects.

*The function receives a window object, a square object, a circle object and a rectangle object as parameters and returns an integer value of:*

- 0: if the click is on the canvas but outside all the objects.
- 1: if the click is inside the square.
- 2: if the click is inside the circle.
- 3: if the click is inside the rectangle.
- -1: if no click is detected.

Consider a moving rectangle (or square) on the game canvas, what are the pieces of information you can obtain about the rectangle from the Graphics library? Now, given the coordinates of the upper corner point, lower corner point, center point, and mouse-click. How can you determine whether the mouse click occurred inside the shape? Likewise, how can you utilize the information about the center of the circle and its radius to answer the same question (in the case where the click is inside the circle)?
Unit Test

test3 builds on test2, and adds an indefinite loop. The loop keeps on running until a click takes place. Notice that the print statement prints the value returned from checkForClick. Use the printed value to verify the validity of the type of click (either inside a square, circle or rectangle, or outside all shapes).

```python
def test3():
    win = createGameCanvas()
    scoreLabelValue, clicksValueLabel = initializeScoreLabels(win)
    shapesList = drawShapes(win)
    while True:
        val = checkForClick(win, shapesList)
        if val >= 0 and val < 3:
            print(val, "a click detected!")
        break

>>> test3()
```

Task 5: Generate random color

In this task, you will implement a generateRandomColor function that produces an RGB color by generating random values for Red, Green and Blue. Hint: check the color_rgb function from graphics.py. You will use the generateRandomColor function in Task 8.

This function receives no parameters and returns a color.

Unit Test

```python
>>> print(generateRandomColor())
#46896e
>>> print(generateRandomColor())
#ba88a7
>>> print(generateRandomColor())
#684414
```
Task 6: Check whether any of the moving objects has reached the boundaries of the Game Canvas

All of the shapes move vertically in discrete quantities. The objects move upwards when the sign of the speed value is negative. Similarly, the objects move downwards when the sign of the speed value is positive. The direction of a shape's movement changes when it hits the boundary of the canvas window; this will cause it to bounce back. The following gif illustrates this movement:

In this task you will implement a `checkBoundary` function that tests whether any of the moving objects reached the boundaries of the game canvas. The function receives a shape object and its respective speed. Firstly, the function needs to determine the type of the object: a rectangle, a square, or a circle. There is a built-in function termed `isinstance(shape, class_name)` that returns True if the shape is of the class type. Here is an example:

```python
def test4():
    win = GraphWin("MyWindow", 500, 500)
    ball = Circle(Point(250,250), 20)
    if isinstance(ball, Rectangle):
        print("Rectangle object detected")
    elif isinstance(ball, Circle):
        print("Circle object detected")

>>> test4()
```

Once the function detects the passed type of the object, it utilizes the information about the shape (i.e., corners and center in the case of a rectangle, and center and radius in the case of a circle) to reason whether the object is inside the canvas or outside of it. Furthermore, in order to bounce the object, you will need to flip the sign of its speed value (make sure that you understand how function `move` operates). Finally, the function returns the speed.

Unit Test

test4 creates the animation illustrated in the gif above. The while loop runs indefinitely. Lines 6-8
move the objects. time.sleep(0.01) is added to prevent the animation from going too fast (this may happen if you're using Windows; you'll most likely not notice a difference if you're using a Mac). The last 3 lines invoke checkBoundary for each shape object. You will have to visually verify the correctness of your function. Make sure the objects bounce without dipping outside the canvas.

```python
def test5():
    win = createGameCanvas()
    shapesList = drawShapes(win)
    speeds = [-5, -7, -9]
    while True:
        shapesList[0].move(0, speeds[0])
        shapesList[1].move(0, speeds[1])
        shapesList[2].move(0, speeds[2])
        time.sleep(0.01)
        speeds[0] = checkBoundary(shapesList[0], speeds[0])
        speeds[1] = checkBoundary(shapesList[1], speeds[1])
        speeds[2] = checkBoundary(shapesList[2], speeds[2])

>>> test5()
```

**Task 7: Generating a list of three speed values**

In this task, you will implement a `generateSpeedsList` function that generates a list of three random, integer speed values in the range of -10 to -3. The returned list will represent the speed of the objects.

*This function receives no parameters and returns a list of three integer speed values.*

**Unit Test**

```python
>>> print(generateSpeedsList())
[-9, -3, -3]
>>> print(generateSpeedsList())
[-5, -5, -10]
```

**Task 8: The Game Driver (main function)**

In the main function, you will be finishing off your program and completing the game. There are three phases to complete:

- **Initialization phase**: consists of creating the game canvas, adding score and click labels, drawing the list of shapes on the window, and generating a random list of speeds, and initialize counters for number of clicks and score.
- **Looping phase**: as long as the loop invariant is True (i.e. number of clicks are less than 10) the objects will keep on moving. You need to keep track of the number of clicks and the score, update
the labels, and, most importantly, make sure that all the objects remain inside the game canvas.

- **Calculating the game score**: a click inside a square is worth 5 points, a click inside a circle is worth 10 points, and a click inside a rectangle is worth 2 points.

You may need to place a small amount of sleep time (e.g., time.sleep(0.01)) after invoking move to prevent the animation from going too fast.

```python
def main():
    #initialize game canvas, labels, shapes, speeds, counters, etc.
    #Wait for the user to click on the Window.
    #Loop until the number of clicks reaches 10
    #-----> move objects, check for click
    #-----> compute score: 5 for square, 10 for circle and 2 for rectangle
    #-----> update labels, check for touching boundaries

    #uncomment the last two lines
    #win.getMouse()
    #win.close()
```

**Turnin Instructions**

Run putty and login to data.cs.purdue.edu. Turn in your project by typing:

```bash
data 51 $ cd cs177
data 51 $ turnin -v -c cs177=COMMON -p project2 project2
```

**Grading Rubric**

This project is worth 100 points.

<table>
<thead>
<tr>
<th>TODO</th>
<th>MAX POINTS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1: drawShapes</td>
<td>10</td>
<td>5 for correct return. 5 for the remaining.</td>
</tr>
<tr>
<td>Task 2: initializeScoreLabels</td>
<td>10</td>
<td>5 for correct return. 5 for the remaining.</td>
</tr>
<tr>
<td>Task 3: drawShapes</td>
<td>15</td>
<td>5 for correct return. 10 for the remaining.</td>
</tr>
<tr>
<td>Task 4: checkForClick</td>
<td>15</td>
<td>5 for correct rectangle, 5 for test, 5 for correct circle test. 5 for the remaining.</td>
</tr>
<tr>
<td>Task 5: generateRandomColor</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Task 6: checkBoundary</td>
<td>15</td>
<td>5 for handling top boundary, 5 for handling bottom boundary. 5 for the remaining.</td>
</tr>
<tr>
<td>Task 7: generateSpeedsList</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TODO</td>
<td>MAX POINTS</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Task 8: main</td>
<td>15</td>
<td>5 for initialization part, 5 for the looping part, 5 for the correctness of score and clicks.</td>
</tr>
</tbody>
</table>

From: [http://courses.cs.purdue.edu/ - Computer Science Courses](http://courses.cs.purdue.edu/)


Last update: **2016/02/21 15:32**