CMSC 143: Object-Oriented Programming with Robots

Lab 9: Programming with Class
Due April 17, 2014

In this lab, we will improve upon the small *Frogger* game we started in lecture. In the first part of the lab, you are asked to create class diagrams of the existing classes in the game. A class diagram describes the attributes and methods of the class as well as type information. For example, below is a Duck class along with its class diagram.

```
class Duck:

  ducks = 0

  def __init__(self, name):
    self.name = name
    self.wings = 2
    self.body = Scribbler()
    Duck.ducks = Duck.ducks + 1

  def quack(self, voice):
    setVoice(voice)
    speak(self.name + " says quack")

  def waddle(self, times):
    for i in range(times):
       self.body.move(1, -0.5)
       self.body.move(1, 0.5)
       self.body.stop()
```

Name	Duck
Class Attributes	DUCKS: int
Object Attributes	name: string wings: int body: Scribbler
Methods	quack(voice: string) waddle(times: int)

In the second part you should improve the game by designing a new class for a new object(s) in the game. For example, you could create a timer for the game, a treasure object that rewards the player when collected, a speed booster, or create a new type of enemy. Feel free to change the current Player and Car class.

First write the class diagram for the new class before writing any python code.

Learning Objectives

○ Create Class Diagrams ○ Design Classes ○ Implement Classes ○ Create a Game

Deliverables

1. `cmsc143_lab9_LASTNAME_FIRSTNAME.pdf` – Your class diagrams.
2. `cmsc143_lab9_LASTNAME_FIRSTNAME.py` – Your program.

```
# A simple game inspired by Frogger
#
```

```
# Keith O'Hara <kohara@bard.edu>
# April 2014

from Myro import *
from Graphics import *
import random

class Car:
    ''' The enemy class in our frogger game.
        The car moves horizontally with some random speed'''

    MAX_SIZE=60
    MIN_SIZE=20
    SPEED=2

    def __init__(self, win):
        ''' creates a car randomly placed in the GraphWin 'win'.
            The car's color is randomly chosen as blue, orange, or red'''
        self.win = win
        x = win.getWidth()*random.random()
        y = (win.getHeight()-100)*random.random()
        self.width = random.uniform(Car.MIN_SIZE, Car.MAX_SIZE)
        self.height = random.uniform(Car.MIN_SIZE, Car.MAX_SIZE)
        self.appearance = Rectangle(Point(x, y),
                                    Point(x + self.width, y + self.height))
        self.appearance.setFill(Color(random.choice(["blue", "orange", "red"])))
        self.appearance.setOutline(Color("gray"))
        self.vx = random.uniform(-Car.SPEED, Car.SPEED)
        self.vy = 0
        self.appearance.draw(win)

    def step(self):
        ''' move the horizontally by car by its speed
            the car wraps around if it reaches the left or
            right of the screen.'''
        self.appearance.move(self.vx, self.vy)
        if  self.appearance.getX() < 0:
            self.appearance.move(self.win.getWidth(), 0)
        elif self.appearance.getX() > self.win.getWidth():
            self.appearance.move(-self.win.getWidth(), 0)

    def checkCollision(self, other):
        ''' detect if this object is colliding with another rectangle
            returns True if the objects are colliding'''
        myx1 = self.appearance.x - self.width/2
        myy1 = self.appearance.y - self.height/2
        myx2 = self.appearance.x + self.width/2
        myy2 = self.appearance.y + self.height/2
        otherx1 = other.x - other.getWidth()/2
        othery1 = other.y - other.getHeight()/2
        otherx2 = other.x + other.getWidth()/2
        othery2 = other.y + other.getHeight()/2

        return myx1 < otherx2 and myx2 > otherx1 and myy1 < othery2 and myy2 > othery1
```

```
class Player:
    ''' The main player class in our frogger game.
        The player can move around the screen in both dimensions'''

    DEAD = 0
    RUNNING = 1
    FINISHED = 2

    def __init__(self, p, win):
        ''' creates a player in the middle of the GraphWin 'win'.
            Uses the Picture 'p' as the player's appearance'''
        self.speed = .5
        self.win = win
        self.startTime = currentTime()
        self.state = Player.RUNNING
        self.appearance = p
        p.setX(win.getWidth()/2)
        p.setY(win.getHeight())
        self.appearance.draw(win)

    def move(self, dx, dy, cars):
        ''' move the player relatively by dx and dy
            the player wraps around if it reaches the left or
            right of the screen, it dies if it reaches the top
            or bottom or if it collides with a car. Also updates
            the time display'''

        self.appearance.move(self.speed*dx, self.speed*dy)

        if self.appearance.getX() < 0:
            self.appearance.move(self.win.getWidth(), 0)
        elif self.appearance.getX() > self.win.getWidth():
            self.appearance.move(-self.win.getWidth(), 0)
        if self.appearance.getY() < 0:
            self.state = Player.FINISHED

        elif self.appearance.getY() > self.win.getHeight():
            self.state = Player.DEAD

        for car in cars:
            if car.checkCollision(self.appearance):
                self.win.setBackground(Color('red'))
                self.state = Player.DEAD

def playGame():
    win = Window("game", 500, 500)
    win.setBackground(Color("black"))

    frog = Player(makePicture("frog.png"), win)     #http://www.openclipart.org/detail/2112
    car = Car(win)

    while frog.state == Player.RUNNING:
        movement = getGamepadNow("axis")
        frog.move(movement[0], movement[1], [car])
        car.step()
```