**CSCI 220L**

# Lab 11

## Learning Objectives

- Create simulations using graphics.
- Write conditional logic with while loops.

## Part A: While Loops

The main focus of this lab is Part B, however I've added this short section to make sure you get some practice writing while loops before the upcoming test. Complete the following function in a new file called `lab11.py`:

0. Write a void function `good_input()` that asks the user to enter a number that is either between 1 and 10 (inclusive), is a multiple of 5, or is -1. If the input does not meet these criteria, the function should ask again indefinitely until the user provides a good input. The function should print a message when the good input is entered.

Although this example seems very specific, you can generalize your solution to nearly any problem that involves getting "good" input. Here's what this type of problem looks like in pseudocode:

1. Ask for initial input
2. While the input is not valid:
    3. Display a message to the user
    4. Ask for input again

## Part B: Designing a Game

In your next homework assignment, you will create a graphical game called *Apple Catcher* in which the player tries to catch falling apples in a moving cart. You will have the opportunity to customize different aspects of the game and add new features of your own design. In this lab, you will implement some of the building blocks of the game, which you can then use on your assignment.

Complete the following functions in `lab11.py`. Include a short docstring for each function.

1. Collision Detection

    There are two main objects in the game: a moving cart and a falling apple. In order to determine whether the falling apple lands in the cart or on the ground, you need to perform *collision detection*. To simplify this process, we will represent both the cart and the apple as circles; in the final version of your game, you may choose to overlay another shape or image (or replace the objects entirely and write new collision detection logic for the new shapes).

    Implement the following functions. You may copy over any relevant functions from the last lab.

    (a) `overlap(cir1, cir2)`, which accepts two `Circle` objects and returns a Boolean indicating whether or not they overlap. You can use this to check if an apple lands in the cart.

    (b) `hit_floor(cir, win)`, which accepts a `Circle` and `GraphWin` and returns a Boolean indicating whether or not the circle has hit or passed the bottom of the window. You can use this to check if an apple lands on the ground.

(c) `hit_wall(cir, win)`, which accepts a `Circle` and `GraphWin` and returns a Boolean indicating whether or not the circle has hit or passed the left or right side of the window. You can use this to check if the cart has hit the edge of the window and needs to bounce back.

Tip: The `GraphWin` class has two undocumented methods that may be handy: `getWidth()` and `getHeight()`, which return the width and height of the window, respectively. Their omission from the graphics module reference is likely an oversight.

2. Designing the Cart

   The cart is represented by a circle that moves horizontally across the bottom of the screen. When it hits the edge of the window, it switches directions and continues moving. At the start of the game, the cart appears at a certain height, but with a random position on the *x*-axis.

   Write a function `create_cart(win, radius, height)` that returns a newly-created `Circle` object with the given radius, a random *x* position, a *y* position equal to the given height (from the bottom of the window), and a color of your choice. The function should also draw the `Circle` before returning it.

   Tip: You can generate pseudo-random numbers with the built-in `random` library.

3. Designing the Apples

   An apple is also represented by a circle. Apples appear at random positions near the top of the window and begin to fall when the player clicks anywhere on screen.

   Write a function `create_apple(win, radius, min_height)` that returns a newly-created `Circle` object similar to `create_cart()`. However, the apple's *x* and *y* positions should be random, though its *y* position should not be lower than `min_height`. That is, its *y* position should be between 0 (top of the screen) and `min_height` (some distance from the bottom).

4. Beginning the Game

   In this problem, you will create an early model of your game. Write a function `main()` that displays a graphical window containing the cart and an apple with the following features:

   - The cart can be displayed as a circle, and should appear near the bottom of the screen at a random *x* position.
   - The apple should appear somewhere above the cart at a random *x* and *y* position.
   - The cart should move left and right across the bottom of the screen. When it hits an edge, it should continue in the opposite direction. You may choose the cart's speed and starting direction (or make them random). One way to implement changing directions is by making a velocity variable and moving the cart by that amount each iteration. If the cart hits a wall, simply negate the velocity to make it move in the other direction. You might also wish to include a call to `time.sleep()` in your loop to slow it down.
   - The cart should continue moving until the user clicks anywhere, upon which the window closes. You can use `win.checkMouse()` to poll the window for mouse clicks.

5. Conclusion

   By this point, you have a good chunk of the base game written already. The next steps (not part of this lab) are to make the apple fall when the user clicks, check whether it lands in the cart or on the ground, and then to spawn a new apple in another position. You'll get more specifics when the homework instructions are released.

   In the mean time, you might consider adding some of your own features to the game, like proper artwork, different types of apples or other falling objects (Fruit Ninja anyone?), more interesting player controls (what if you controlled the cart instead of the apples?), time constraints (what if the game sped up over time or had timed rounds?), or anything else that comes to mind. The homework assignment may offer some extra credit opportunities for interesting game design, so feel free to use your lab group to demo any ideas you have.

   If you're looking for something to add but you're not sure what, here's an idea:

   **Optional:** What if the user has to click on each apple to make it fall? This would also enable you to spawn multiple apples at a time without overwhelming the player. To implement this, write a function `apple_clicked(apple, pt)` that returns a Boolean indicating whether or not point `pt` is inside circle `apple`. Then, replace your `win.checkMouse()` with `apple_clicked(apple, win.checkMouse())`. This will let you loop until the user clicks specifically on the apple, and not anywhere else on the screen.

   Note: If the user hasn't click anywhere yet, `win.checkMouse()` returns `None`, so you should make sure your function returns `False` if `pt` is `None`.

Upload `lab11.py` to the OAKS dropbox before the deadline. Make sure you have most of the exercises completed before your lab meeting.