GDAPS2 – Practice Exercise

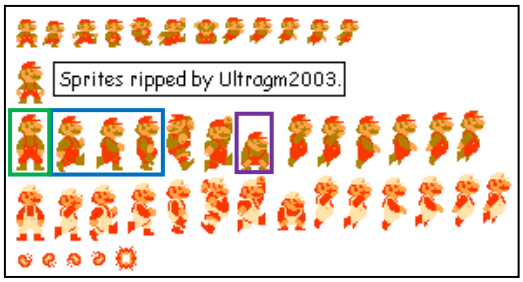
Mario Walking

Objective:

Familiarization with a Finite State Machine and its transitions, and some practice using a Sprite Sheet.

Details:

Open up the "Mario Walking" starter code, which is provided for you on MyCourses. The code has some pre-defined methods in the **Game1** and **Mario** classes for animating Mario, using the image to the right.



We'll be using the four sprites in the boxes in the middle left + the crouching sprite in the middle of the same row.

- The farthest left frame is Mario standing
- The three are his walk cycle.
- The last is Mario crouching

These sprites will cover left and right facing version of all positions, since there is a SpriteEffect that can flip a sprite.

Finite State Machine:

First, **draw the finite state machine** that shows Mario's potential states based on player input. On your arrows, write the kind of input – or lack thereof – that would cause a transition from one state to another. Base your FSM model on the following requirements for this exercise:

- The possible states are *FaceLeft, WalkLeft, FaceRight, WalkRight, CrouchLeft and CrouchRight.*
- There are three possible events that trigger transitions: a *Left* or *Right* arrow key press (or lack of key press) or an *Up* arrow key press (or release).
- Mario must be facing left before he starts walking left and must be facing right before walking right.
- Mario cannot go from walking in one direction directly to facing the other.
- Mario must be standing in order to crouch and he cannot move (or change the direction he's facing) while crouching.

You can draw this on scratch paper or in the tool of your choice. We will not be collecting it. However, ***you must show us this drawing when you demo the exercise in order to receive credit!***

There are any number of free tools to help with modeling. I don't care what you use. Some that I have tried include:

- https://www.draw.io/
- https://www.gliffy.com/
- I make most of my diagrams using http://plantuml.com/ - but this involves learning a new markup language that is then rendered via an online tool or IDE, etc. integrations such as:
  - https://gsuite.google.com/marketplace/app/plantuml_gizmo/1009735747823
  - https://www.planttext.com/
  - And many more - http://plantuml.com/running

Implementation – Standing and Walking

To start, focus ONLY on the standing and walking states in your state machine (helper methods already exist for you to draw these).

1. Alter the **Game1.Update()** method so that:
    a. It changes the MarioState field appropriately (using the Mario object's State property), based on the player's keyboard input and Mario's current state.
        Note: Changing the state here in Update lets you separate the underlying state of Mario from how his state is shown to the player – a useful habit since you don't want to mix the specifics of your art with the logic of your game.
    b. If Mario is walking, update his X location based on left or right movement.

2. Alter the Mario class' Draw() method so that Mario's visuals match his state, based on the current value in state. You'll need a switch statement here to check the state (but not change it). To help draw Mario, two methods are defined for you: `DrawStanding()`and `DrawWalking()`. These will either draw Mario standing or walking. They also take a SpriteEffects enum parameter, which controls whether Draw() should flip the image horizontally and/or vertically.

Test and make sure this much works before you proceed!

Implementation – Crouching

To start, focus ONLY on the standing and walking states in your state machine (helper methods already exist for you to draw these).

1. Add states to support crouching to the `MarioState` enum in Mario.cs
2. Alter the **`Game1.Update()`** method so that your FSM switch supports:
    a. Transitioning to crouching from either *FaceLeft* or *FaceRight* IF only the Down arrow is pressed (i.e. no trying to walk while crouching)
    b. Transition back to *FaceLeft* or *FaceLeft* when the Down arrow is no longer pressed.
3. Add a new **`DrawCrouching()`** helper method to Mario.cs to draw only the crouching frame from the sprite sheet.
4. Alter the Mario class' **`Draw()`** method so that Mario's visuals support the new states.

Finished?

Make sure you followed the class's coding standards for all code you created!

Demonstrate your FSM draw AND working solution for the instructor or TA when done and commit & push your changes to your GDAPS2 repo on the KGCOE server.

For this PE, you will make ONE myCourses submission: A **link** to the final commit in your git repo that contains the project. *(See the PE: MonoGame Basics write up for instructions.)*