

Practical Session 4: Running and jumping in Pygame

Overview

In this session we will be looking at some simple animation of a running character, and the mechanics behind making him jump. The tasks covered today will help if you intend on making some sort of platform game. The homework tasks for this week are to expand the platformer example and make a start on implementing your coursework game.

You will need to download the “Scratch/Pygame Resources” and “Pygame Scripts” from the class materials wiki page. It is acknowledged that the “Jet Set Willy” (JSW) character is copyright and only used for academic purposes; it should not be used in any commercial project without the copyright owner’s permission.

Task 4.1

Your first task is to set the scene, by creating the appropriate sprites and animating the JSW sprite according to user input. Firstly, create three sprites using the provided resources:

1. JSW
 - 1.1. Use the new AnimatedSprite class, loading the “jsw.png” animation strip
 - 1.2. Initialising
 - 1.2.1. Start at (40, 344)
 - 1.2.2. Frame size is 32
 - 1.3. Rendering
 - 1.3.1. Render after the platform sprites so he appears in front of them
2. Platform1
 - 2.1. Use the updated Sprite class, loading the provided image
 - 2.2. Initialising
 - 2.2.1. Start at (90, 340)
3. Platform2
 - 3.1. Use the updated Sprite class, loading the provided image
 - 3.2. Initialising
 - 3.2.1. Start at (140, 320)

Now you should add controls so that JSW moves left or right using the arrow keys, and that he is facing the appropriate direction (use `Sprite.setFlip()`). To add animation, switch to the next costume when moving right, and the previous costume when moving left. You may want to put a delay in so that the animation runs

smoothly (tip: create a global frameCounter, increased once per render, and change anim frame on every 5th frame [frameCount % 5 == 0])

Task 4.2

Now let's add in some jumping code. The basics of this are really quite simple. Ignoring the platforms for the moment, the algorithm is as follows:

1. When space is pressed
2. Repeat 30 times (once per frame, not a local for loop)
 - 2.1. Move up 1 pixel
3. Wait for a short time (not necessary)
4. Repeat 30 times (once per frame, not a local for loop)
 - 4.1. Move down 1 pixel

If you add in this code and run the program, you will see a very simple jump action being performed.

Task 4.3

Now for the tricky(ish) part: making JSW jump onto a platform, and drop to the ground if he walks or jumps off the edge. There are many ways to achieve this, so it is up to you to find an appropriate method. To get you started, here are some tips:

- Keep the code that makes JSW jump up 30 pixels when the space bar is pressed.
- When he reaches the top of his jump, you could set a variable to say he is falling, remembering to set this variable so he is not falling when the program starts.
- Make him continuously fall until something stops him, rather than repeating a 1 pixel fall 30 times
- When he is falling, check for collision against each of the platforms, and stop him from falling further if he lands on one.
- Similarly, if he reaches the bottom of the screen, stop him from falling.
- If he is moving left or right and stops colliding with the platform he is on, make him start falling again.

Homework Task 4.1

Depending on how you have completed the above tasks, there are probably a few problems with the project at this point. Below are some typical problems that you could have a go at fixing, but as usual, use your imagination and make changes that you think should be made:

- JSW may be able to land inside a platform rather than on top of it, depending on how the collisions are detected. You could sort this by making sure he is above the platform.
- You could make a change so that JSW bangs his head if he hits a platform when jumping, rather than jumping through it.
- Should he be able to change direction mid-jump? In the original JSW games, when you started your jump you were committed to travel in the direction you were moving i.e. diagonally left, diagonally right, or straight up and down.
- The jump is quite unrealistic when scripted. A better way would be to use gravity so that he starts jumping at one speed, but his speed decreases until he reaches the top of his jump, and increases as he falls. Tip: His vertical speed would be changing by a constant acceleration value.
- You've seen how pickups and scrolling backgrounds can be used, so you could try adding these to this program.
- JSW could be given a number of lives, which are lost when he falls from too great a height, or collides with certain obstacles / enemies.

Homework Task 4.2

By now you should have given some thought to the game you will be making for your coursework (as per homework task 3.1). Your second homework task for this week is to make a start on the development of your game so you can gain a better insight into how achievable your game design is.