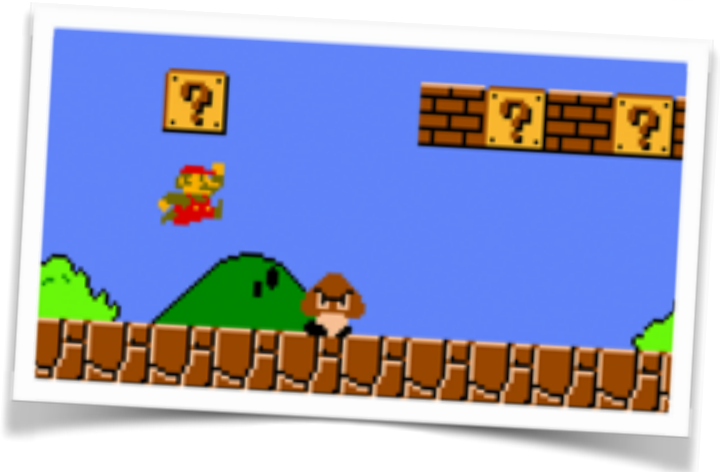

Hopscotch Coding - Week 4 - ‘Platformer’

Overview

In this third week of the Hopscotch course the pupils will learn how to create a simple Platform game, in the vein of classic titles such as Super Mario Bros and Sonic The Hedgehog. The keywords for this week will be **Platformer, Value & Projectile**. In this week’s lesson the class will have the opportunity to be far more independent in their game design, deciding on level structure and challenges. There will however be some mandatory elements including the use of a button controller system and a projectile sprite.

Learning Objectives

- To understand and be able to use the terms **Platformer, Value & Projectile.**
- To begin to develop both game design and coding skills with the use of the Hopscotch App.
- To be able to traverse a 2D plane on both the X and Y axis using a new method of control.
- To understand how to determine events via the values of sprites.

Tools needed

- 15 Ipads

Starter Activity (10mins)

Before any activity commences with the iPads the tutor should post the keywords of the lesson on the board and give a brief explanation of their use in today’s lesson e.g. **Projectile** - “A Sprite that is being thrown by another character”. Be sure to remind all of the pupils that they should use these words when discussing their work with either you or their peers. Do not forget to remind the children of the previous weeks’ keywords and that they should still aim to use them. Introduce today’s brief and that they are trying to emulate the game ‘**Super Mario Bros**’. If you wish, supplementary video material could be included as a prompt for the class!

Activity 1 - Coding the Button Controls (5 - 10 mins)

In their fourth week of coding the class should now be very familiar with the use of the variables **Change X by** and **Change Y by** when moving their character. This week will use the same method of control as last week’s Crossy Road clone. Using the **….. is tapped** applied to 4 appropriate arrow sprites (there are very suitable arrow emojis) encourage the children to independently code each direction of movement. If any of the class have forgotten how to accomplish this task feel free to use hints on the board.

However once they have finished coding the four direction buttons, the pupils must also include a fifth button. This can be any sort of button visually, but will act as the 'power' button.

Activity 2 - Coding the Projectile & Character Values (20 mins)

Firstly before any coding takes place, pupils must decide on the sprite they will use as a projectile (emojis are a good option). Below is the process to code the projectile step by step:

1. Sprite needs to follow the character as if in their pocket - Up until this point, whenever variables have required a value, the pupils have used the calculator. Today however, they will use the X and Y value of the main character to **Set Position** within the **Game Starts** conditional, equalling their starting position.
2. Sprite must move equally with character - There are multiple ways to complete this task; and no one right way. Due to the nature of the app and processing power of the iPad the following method appears as the most elegant. The pupils need to simply repeat the button code (above) for the projectile, using the exact same values. If these values differ then the game will not work as intended, so it should be easy to **Debug** this portion.
3. Coding the power button - Using a similar code to the movement buttons above the pupils will now code the firing of the projectile across the screen. Using a **… is tapped** conditional they require three variables: **Set Speed**, **Change X By** and **Set Position**, all of which they have used in the past. Just as in step 1 the values in the **Set Position** must be the values of the main character. This sends the projectile across the screen when the power button is pressed and once completing its movement across the screen returns to the main character.
4. Making the Sprite Invisible - For the final steps within the Game Starts conditional (above) add a **Set Invisibility** variable (at 100%) before the **Set Position** already coded. Add the opposing **Set Invisibility** (at 0%) before the **Set Speed** in the '**power is tapped'** (above) conditional and a final **Set Invisibility** (at 100%), after the final **Set Position** at the end of this conditional. The reason we add these **Set Invisibility** last is that it makes debugging the rest of the code far easier when the projectile is visible.

Activity 3 - Projectile Collisions (20 mins)

Using almost identical code to our sprite in the previous weeks **(See Week 2)** the class should now code a collision code for when the projectile hits a selection of 'enemy' characters. In this game there should be between 4-6 enemies. In order to best aid their learning, the class should attempt to do this work independently as they completed a very similar task in the previous week. Remind them that they will need to repeat the process for every single moving obstacle that appears on the screen. Encourage the class however to experiment with some other variables such as **Shrink by**, **Spin** or **Flip**. They should also include a negative response if one of the enemy sprites hits the main character. This could be returning to the beginning of the level for example….

Activity 4 - Including a Goal Sprite and Play-testing (15 mins)

The final activity of this lesson will be to finish the game by adding a **Goal Sprite** which triggers an indication that the player has won. This could be for instance a specific noise that triggers; again let the pupils lead in their decisions. Once this is complete encourage the class to engage in play-testing. It is important that the difference between **Debugging** (checking for errors) and **Play-testing** (testing how well the game works/is it playable/is the difficulty appropriate) is made explicit. Explain that if play-testing reveals anything about the game that is not as good as it could be, then this is the time to change it. This is also a good time for the pupils to customise their games with colour or with specific visual designs.

National Curriculum:

Coding: Write a simple algorithm whilst commanding sprites.

Coding: Debugging and checking accuracy of game.

Coding: Create a short game/animation using a simple visual programming language.

Coding: Beginning understand basic computing and mathematical concepts and vocabulary.

General: Numeracy, Reasoning Skills, Creative Design

e: bookings@milittlepad.com www.milittlepad.com telephone: 07860 937 270