HOPSCOTCH

Hopscotch Coding - Week 2 - 'Labyrinth'

Overview

In this second week the class will be introduced to the idea of creating a functional game with a distinct set of controls and rules. Inspired by the classic game Labyrinth a.k.a Marble Maze, the game will feature a character that moves with the tilting of the iPad's accelerometer and a sprite (target) that will randomly re-spawn when it is touched by the character. The keywords in this lesson will be **Collision**, **Accelerometer, Sprite.** This will be the first lesson where pupils will be designing a 'game' and as such there will be opportunity for them to be creative with characters and backgrounds e.t.c.

Learning Objectives

- To understand and be able to use the terms **Collision**, **Accelerometer** and **Sprite**.
- 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 method of control.

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. **Collision** - "When something bumps into something else". 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 classic game '**Labyrinth**'. If you wish, supplementary video material could be included as a prompt for the class…

Activity 1 - Using the Accelerometer (15 mins)

First introduce the class to the accelerometer and how it measures the tilt of the iPad and relays data to Hopscotch. Secondly introduce the four appropriate **iPad tilted …** conditionals to the class, explaining that all four will be included to allow the character to move in all directions. The pupils should then, semi-independently, code their character to move in the four directions (using the **change x by** and **change Y** by variables). It is advised that the class may not remember everything from the previous week and therefore doing each direction one by one as a class may be best. This should be taken as an opportunity to quiz the class on what they remember of using the X and Y plane. Once all the conditionals have been coded encourage the class to **Debug** their game and change the value of each movement to find the speed they prefer.

Activity 2 - Coding the Sprite and Collisions (10 mins)

Now that they have a character that moves across the screen the game requires a goal. Explain how this effects the game - that it gives the game an aim (all games require an aim in order to be a game). This goal will be introduced as a **Sprite** - a small character or shape that is not controlled by the player (also known in gaming as an NPC). In this instance a good sprite would be an emoji of a trophy for example. Once the sprite has been chosen the tutor must introduce the collision specific conditional that will be used today - **….. bumps ……** This conditional will allow for any variable to initiate and effect the character/sprite that has been selected once the collision has happened. In this instance the pupils will require four variables. The first two are self explanatory - **Start Sound** and **Set Invisibility.** The second two may need explaining.

Activity 3 - Re-spawning the Sprite (10 mins)

Re-spawning the sprite may take a little explaning. Directly below the other variables, pupils will require the variable **Set Position.** Re-explain the concept of coordinates if the children have forgotten, and explain that **Set Position** requires both an X and Y coordinate number. However instead of using the calculator they will use the random variable. **Random** may need to be explained, particularly that it offers a two number range (the suggested range is 10 to 880). This will cause the **Sprite** to randomly re-spawn somewhere on the screen. However as it is still invisible, they should finish coding the sprite by including the **Set Invisibility** variable and returning the number to 0 (remember that it is measured in percentages!).

Activity 4 - Customisation of Game (15 mins)

The final activity of this lesson will be predominantly led by the students as they are encouraged not only to **Debug** their game but to customise it. They may customise it by adding more collision **Sprites** as well as including obstacles. Obstacles are achieved by coding a **Set Position** variable within a **… bumps…** conditional on their main character, that sends the character back to the beginning of the game. These are not compulsory suggestions, but rather ideas to get the class to independently decide on new rules for their games.

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