# Worksheet 3: Direction

|  |
| --- |
| **Learning Intentions**: We are learning to be able to … |
| Direct sprites using the scratch direction system. |
| *Why are we learning this?* |
| Turning is a part of animation and movement that we found out was so important last lesson. |
| **Success Criteria**: I will be successful if I can … |
| Be able to angle a sprite in the correct direction for the problem I am solving |

## Instructions:

|  |
| --- |
| Acquire and integrate knowledge: |
| Take note of the directional **blue arrow**, which by default points to 90 degrees (right), *regardless of whichever way the sprite is first drawn*:    Scratch by default aims sprites facing 90 (right), even if you drew the sprite facing upwards / downwards etc. Note the difference between Scratch and normal Cartesian coordinates:  Image result for scratch direction system  The **blue line** indicates which way the sprite is currently facing, **regardless of how the sprite costume looks.** Note the directional options on the **set rotation style** block – you will need them to answer the next question:     1. What is the effect of the previous script on the cats costume when each one of these values is set in the **set rotation style** block?  |  |  |  | | --- | --- | --- | | **Set rotation style block** | **Values** | **What happens to the costume when you run the script above with this value set?** | |  | left-right | Click here to enter text. | | don’t rotate | Click here to enter text. | | all around | Click here to enter text. |   The Scratch direction system will still allow you to turn or point past 180 degrees – and the degrees will be added in a clockwise direction for positive (or anti-clockwise for negative). For example:     1. Which way will a sprite be facing if the following block is run?  |  |  | | --- | --- | | **Script block** | **Direction of sprite after script block is run:** | |  | Click here to enter text. | |  | Click here to enter text. | |  | Click here to enter text. | |
| Extend and refine knowledge: |
| We can calculate the rebound angle of a ball on a paddle via the following algorithm:  **Rebound angle** = 180 – [*direction at collision*]:     1. Delete the cat, and add a **blue paddle** sprite (use a solid blue rectangle) with this script: 2. Implement the above **rebound angle** algorithm on the tennis ball: |
| Use knowledge meaningfully: |
| 1. The game you have started created is called Pong, and is available online via the Atari Arcade (<https://www.atari.com/arcade>). It implements some of the skills you have learnt this far. (We will make Breakout, which is similar, later in the course).    1. Can you create a second player (player two)?    2. Can you create an unbeatable AI paddle?    3. Can you create a beatable AI paddle? (note - this is much harder than an unbeatable AI paddle, as it implies *difficult but not impossible*, meaning the computer has to make a mistake, which it is not programmed to do). |