Vectors

In Unity vectors are used to represent the position of objects as well as direction, and movement. A 3D vector has three values: x, y, and z, corresponding to each of the axis in 3D space. For example (in 2D), the vector (3, 4) represents a position 3 units along the x-axis, and 4 units along / up the y-axis.

**Draw a diagram.**

When adding a vector to a position vector, the vector being adding can be said to represent movement. For example, if we add (2, 1) to the above vector, we move 2 units along the x-axis, and 1 unit up the y-axis; giving a new position of (5, 5). When adding vectors, each component from one vector is added to the other.

**Draw a diagram.**

A vector can be used to represent a direction. For example, the vector (0, 1), when drawn as a line originating from (0, 0) points upwards; whereas, the vector (-1, 0) points to the left. The two vectors from previously are both pointing upwards and to the right, but at different angles.

**Draw a diagram.**

Another useful thing about vectors is scalar multiplication. This is when you multiply a vector by a ‘float’ or ‘int’ etc. When doing so, the vector maintains its direction but its magnitude increases. For example, when multiplying the vector (0, 1) by 2, each component (x: 0, and y: 1) is multiplied by 2, giving a vector of (0, 2). When we draw these vectors as lines, they both point in the same direction, the multiplied one is just double the length.

**Draw a diagram.**

### **Player movement calculation**

Here we will construct a vector that represents the movement the player sphere will make each frame.

Input.GetAxis translates a key press into a scalar value between -1 and 1 based on which of the axis keys is pressed, and for how long. Input keys for the horizontal axis are 'a' and 'd', and 'left' and 'right'. This value will be used to determine ‘How far’ the player moves. The sign (+/-) of the horizontalInput will also be used to determine ‘Which way’ the player moves

float horizontalInput = Input.GetAxis("Horizontal");

Vector3.right is the Vector (1, 0, 0) which is being used as a direction vector; it points to the right (hence the name). This value will be used to determine ‘Which way’ the player moves.

For example, when the player presses the left key, the horizontalInput value will return a value between -1 and 0. Lets pretend that -1 is returned. If we multiply the rightward vector by -1 we get the Vector (-1, 0, 0) because we apply the multiplication to each component of the vector. The vector (-1, 0, 0) as a movement vector means to move 1 unit to the ***LEFT*** on the x-axis.

The MaximumSpeed property describes the maximum amount of movement that the player can make each frame. If we set the MaximumSpeed property to say 10, and multiply the result of the above calculation by it, we get the vector (-10, 0, 0) which means to move 10 units to the ***LEFT*** on the x-axis.

However, we do not want to move out cube 10 units each frame.

**DEMONSTRATE**

As you can see the cube is barely visible as it zips off screen. The reason for this is that the cubes movement is ***per frame***. To give an idea of how fast that is, Unity is currently running at [~60fps], 10 units each frame means the cube is moving at 600 units per second if the game runs at 60fps.

To fix this, we want our cube to move at the calculated rate ***per second.*** There is a convenient way to do this in Unity. If we factor the frame rate into our calculation then the net result will be the per second movement.

To do this, we can use Time.deltaTime which is the amount of time in seconds that the last frame took.

Vector3 movement = Vector3.right \* Speed \* horizontalInput \* Time.deltaTime;

**transform.Translate(movement);**