COMP2007 - Game Development

Week 1 - Code session

Please download the Unity project called “Random” on the DLE in the Week 1 section

Random Walk

Step Mover

We get a random integer (int) between zero and three

```
randomValue = Random.Range(0, 4);
```

NOTE: when Random.Range is dealing with int numbers, the max is never returned
This is known as “exclusive”, the max number is excluded from the possible values

Apply rules based on the random number

- If the random number is zero, move right
- If the random number is one, move left
- If the random number is 2, move forward
- Otherwise, move backwards

Code for the rules of the algorithm

```
if (randomValue == 0)
{
    currentX = moveDistance;
}
else if (randomValue == 1)
{
    currentX = -moveDistance;
}
else if (randomValue == 2)
{
    currentZ = moveDistance;
}
else
{
    currentZ = -moveDistance;
}
```

Step Array Mover

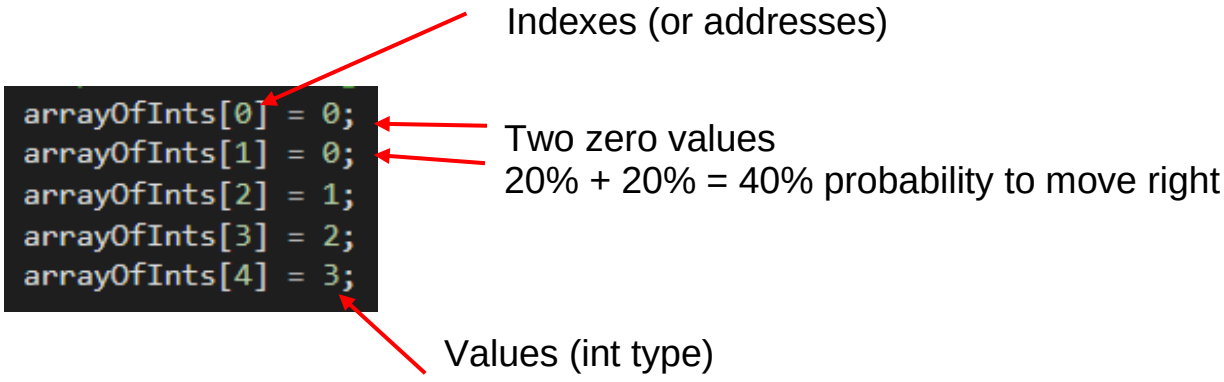
This algorithm generates a random number for an index in an array
The array values contain a number for each rule:
0 = move right, 1 = move left, 2 = move forward 3+ = move backward

With five array values, the probability of selecting one value is 20% (5 x 20 = 100)

In the code have an array of five int values

```
int[] arrayOfInts = new int[5];
```

The array values



We have two zero values, which will increase our probability to move right up to 40%

Step Percentage Mover

The algorithm generates a float (decimal) value between 0.0 and 1.0.
Consider the values as percentages, 0.5 is 50% of 1.0 and 0.25 is 25% of 1.0

```
randomValue = Random.Range(0.0f, 1.0f);
```

Apply rules based on the random number

- If the random number less than 0.25 (25%), move right
- If the random number less than 0.5 (50%), move left
- If the random number is less than 0.75 (75%), move forward
- Otherwise, move backwards

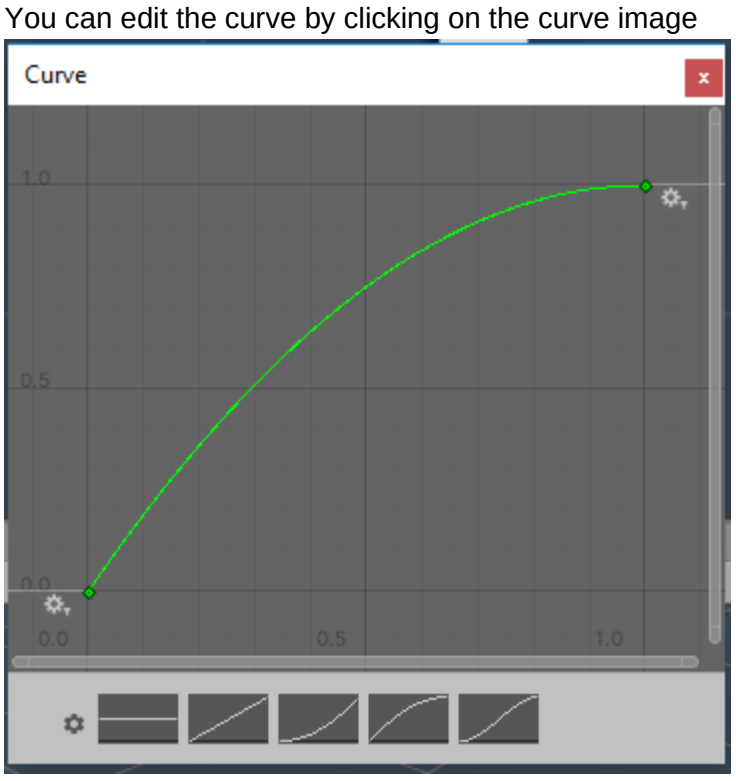
```
if (randomValue < 0.25f)
{
    currentX = moveDistance;
}
else if (randomValue < 0.5f)
{
    currentX = -moveDistance;
}
else if (randomValue < 0.75f)
{
    currentZ = moveDistance;
}
else
{
    currentZ = -moveDistance;
}
```

NOTE: float numbers in unity require a letter “f” after declaring the number if there is a decimal point

Step Curve Mover

This algorithm uses a Unity AnimationCurve to plot a random number in the curve over time
These create a curve on an X/Y axis where we can plot values along the curve over time.
X is the value, Y is the time

You can edit the curve by clicking on the curve image



Editing curves
https://docs.unity3d.com/2020.2/Documentation/Manual/EditingCurves.html

Scripting reference
https://docs.unity3d.com/ScriptReference/AnimationCurve.html

We create a random curve number between zero and one

```
float randomCurve = Random.Range(0.0f, 1.0f);
```

Use the Evaluate method on the AnimationCurve to plot where our random number is in the the graph

```
randomValue = curve.Evaluate(randomCurve);
```

Apply rules based on the random number

- If the random number less than 0.25 (25%), move right
- If the random number less than 0.5 (50%), move left
- If the random number is less than 0.75 (75%), move forward
- Otherwise, move backwards

```
if (randomValue < 0.25f)
{
    currentX = moveDistance;
}
else if (randomValue < 0.5f)
{
    currentX = -moveDistance;
}
else if (randomValue < 0.75f)
{
    currentZ = moveDistance;
}
else
{
    currentZ = -moveDistance;
}
```

Step Vector Probability

This algorithm creates two random numbers, for the X and Z directions
The gameobject will move in two directions at once

```
randomX = Random.Range(-moveDistance, moveDistance);
randomZ = Random.Range(-moveDistance, moveDistance);
```

The movement positions will be very random!

Unity Documentation

MonoBehaviour

MonoBehaviour is the base class from which every Unity script derives
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/MonoBehaviour.html

MonoBehaviour.Start

Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/MonoBehaviour.Start.html

MonoBehaviour.Update

Update is called every frame, if the MonoBehaviour is enabled.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/MonoBehaviour.Update.html

Time.deltaTime

The completion time in seconds since the last frame (Read Only).
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Time-deltaTime.html

InvokeRepeating

Invokes a method after set seconds, then repeatedly every few seconds.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/MonoBehaviour.InvokeRepeating.html

Vector3

Representation of 3D vectors and points.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Vector3.html

Transform

Every object in a Scene has a Transform. It's used to store and manipulate the position, rotation and scale of the object.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Transform.html

Transform.position

The world space position of the Transform.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Transform-position.html

Transform.Translate

Moves the transform in the direction and distance of translation.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Transform.Translate.html

Random.Range

Returns a random float or int within a min/max range.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Random.Range.html

Random.InitState

Initializes the random number generator state with a seed.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Random.InitState.html

Random.state

Gets or sets the full internal state of the random number generator.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Random-state.html

AnimationCurve

Store a collection of Keyframes that can be evaluated over time.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/AnimationCurve.html

GameObject

Base class for all entities in Unity Scenes.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.html

Instantiate

Spawn a GameObject or prefab into the scene
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Object.Instantiate.html

Attributes

SerializeField

Force Unity to serialize a private field (make the field visible in the Inspector)
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/SerializeField.html

Header

Add a header above some fields in the Inspector.
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/HeaderAttribute.html

C# Documentation

Arrays
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/

List
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-5.0

List Add
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.add?view=net-5.0

List Clear
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.clear?view=net-5.0

Region
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/preprocessor-directives/preprocessor-region

Protected
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected

Virtual
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual

Override
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/override

Abstract
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract

Using directive
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-directive

Contexts for the word using in C#
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using

HashSet
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.hashset-1?view=net-5.0

HashSet Add
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.hashset-1.add?view=net-5.0

Foreach
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/foreach-in

Do While
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/do

Differences between C# Namespaces and Java packages

Namespaces link
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/namespaces/

A Namespace operates the same way a Java Package does.
You organise your classes within a Namespace as you would files in a folder.
Just like Java Packages, you use a dot (.) to create the hierarchy, like this : mygame.code

Declaring a Namespace in C#

This code declares the namespace “game.code”. Note we use the word namespace at the start
Also, the namespace has curly braces, code must be within the braces to be part of the Namespace

```
namespace game.code
{

}
```

ONLY the code WITHIN the braces is part of the namespace!

CORRECT	WRONG

`namespace game.code`
`{`
`  public class MyClass`
`  {`
`  }`
`}`
	`namespace game.code`
	`{`
	`}`
	`public class MyClass`
	`{`
	`}`

Inheritance in C# compared to Java

In Java, to inherit from another class, you must use a keyword, either “extends” or “implements”.
In C#, you only need a colon (:) after the class name!

The class “MyClass” is inheriting from “BaseClass” using a colon after the the “MyClass” declaration

```
public class BaseClass
{
}

public class MyClass : BaseClass
{
}
```

NOTE: like Java, you cannot inherit from multiple classes, but you can inherit from multiple interfaces!

Interface inheritance in C#

After any base class, interfaces are added after a comma
NOTE: you must declare the class BEFORE any interfaces!

```
public interface IInterface
{
}

public class BaseClass
{
}

public class MyClass : BaseClass, IInterface
{
}
```