2020

Crap Guide to Unity

VERSION 0.1, WORKING IN PROGRESS

ANSON WONG

This is not a guide on how to use Unity, but a simple guide of what Unity can offer. Mainly includes commonly used classes, methods, etc with a summary of what it can do. It is a nice way to introduce common components and methods instead of drilling through Unity's official documentation and getting confused.

Please always refer back to the official documentation for method parameters

https://docs.unity3d.com/Manual/index.html

Contents

Classes

C#

Primitive types

String

Char

Int

Float

Double

Null

Array and List

<Type>[]

Typical fixed length array

Methods	Description
.Length	**Get** size of array

List<Type>

Similar to Java's ArrayList

Methods	Description
.Add()	add parameter to list
.Contains()	Check if parameter in in list
.Find()	Find an element in the list *Personally, I never get this to work correctly, so I normally just write a while loop
.Count	**Get** size of array

Unity

GameObject

Everything in the scene (and Hierarchy) is a GameObject, for example the cube, plane, camera, light, empty GameObjects, etc.

You can create an empty GameObject by right clicking in the Hierarchy and create one. They will appear as nothing other than axis for their transform components

*I used them to either get a point in the world or as a Parent of something with multiple GameObjects attatched

Methods	Description
.activeSelf	Check if the gameObject[1] itself is active
.SetActive()	Set .activeSelf to parameter
.tag	**Get** tag of the gameObject *tag is another way to identify a type/ group of a GameObject, for example, Player, Enemy, Wall, etc
.CompareTag()	Compares the current gameObject's tag to the parameter *mostly used for identifying what GameObject collided and act differently
.transform	**Get** the transform component
.GetComponent <ComponentType>()	**Return** a certain component from gameObject If there is no said component **Return** Null instead

.TryGetComponent (out ComponentType variableName)	**Return** Boolean of if the gameObject has said component
	A Safer way of getting a component from a GameObject
	
	Example:
	If (gameObject.TryGetComponent(out TestComponent x)
	{
	x.DoSomething();
	} else {
	//it cannot find the component
	Debug.LogError("Can not find component");
	}

Static Methods	Description
GameObject .FindGameObjectWithTag()	**Return** a GameObject in the scene with the tag specified in the parameter There is also FindGameObjectsWithTag() which returns an array of GameObjects
FindGameObjectOfType <ComponentType>()	**Return** a GameObject in the scene with the Type specified *best for finding a specific script/ component in the scene that you can't drag and drop in the inspector *best used in Awake() or Start() to set variables that are components in the scene, eg. Main Camera There is also FindGameObjectsOfType() which returns an array of GameObjects
Destroy()	Destroys the GameObject specified in the parameter Eg. Destroy((GameObject) g, (float) 5f); It destroys the GameObject g after 5 seconds *read documentations for more parameters

[1] Lower case gameObject to signify it's variable, Upper case GameObject to signify the class GameObject

Instantiate()	Instantiate the GameObject specified in the parameter Eg. Instantiate((GameObject) g, (Vector3) v, (Quaternion) q) Creates the GameObject g at the location v with a rotation of q *q is often Quaternion.identity for no rotations *read documentations for more parameters

Vector3, Vector2

It is basically a tuple that stores 3 or 2 floats respective. It is normally used for representing a position and/ or direction. The values are stored in this order **(x ,y ,z)** for vector3 and **(x, y)** for vector2.

Vector3 and Vector2 shares a lot of similar methods, best to double check the documentation for more details

Properties	Description
.magnitude	**Get** the magnitude/ length of the Vector Eg. a Vector2 (3,4) has a magnitude of 5.
.normalized	**Get** the normalized Vector, where the Vector is in the pointing the same direction but scaled down to have a magnitude of 1 Eg. A Vector2 (-4,-4) will be normalised to (~-0.7,~-0.7)

Static Properties

.right, .up, .forward	**Get** the world axis x, y, z with magnitude of 1 respectively

Static Methods	Description
.Angle()	**Return** the angle in degrees between the 2 Vectors *it is a float, not a Quaternion *you can also use .SignedAngle()

.MoveTowards()	Moving a Vector to a target Vector in a constant speed
	
	Eg. Moving a GameObject to position targetVector
	
	Vector3 targetVector;
	Float speed;
	
	Void FixedUpdate(){
	transform.position = Vector3.MoveTowards(transform.position, targetVector, speed*Time.fixedDeltaTime);

.Slerp()	“Spherically interpolates between two vectors.” – Unity Basically it allows you to smoothly move a Vector to a new Vector

	Eg. Have a GameObject to face in the direction of the targetVector
	
	Vector3 targetVector;
	Float speed;
	
	Void FixedUpdate(){
	transform.position = Vector3.Slerp(transform.position, targetVector, speed*Time.fixedDeltaTime);

Quaternion

This handles the rotation.

Do not modify the x, y, z, w values directly unless you understand Quaternion

It can also be used as a rotation matrix.

Eg.

Quaternion q;

Vector3 v;

q*v allows the rotation q to v

Properties	Description
.eularAngles	**Get** a Vector3 representation of the Quaternion in degrees

Static Methods	Description
.Angle()	**Return** the angle between the 2 Quaternion
.AngleAxis()	**Return** a new quaternion which rotates parameter angle degrees around parameter axis Eg. Quaternion.AngleAxis(30f, Vector3.Up) Retruns a Quaternion with a rotation of 30 degrees in the y azis
Eular()	**Return** a new quaternion based on the parameters of the angle on the x, y, z axis
.Slerp()	Same as the one for Vector3 but for rotation

MonoBehaviour

All Unity scripts derives from this script

Diagram of the order of methods calls:

https://docs.unity3d.com/560/Documentation/uploads/Main/monobehaviour_flowchart.svg

Methods

There are methods from the MonoBehaviour class that you can define

Methods	Description
Start()	This method gets called once before the first frame's Update() gets called
Awake()	This method gets called once when the script instance gets loaded Aka, first method to be called, earlier than start
Update()	Method gets called on every frame
FixedUpdate()	Method gets called on a fixed time By default, it gets called once every 0.02 of a second You can change that in the settings
LateUpdate()	Method gets called on every frame, but after all the Update methods have been called
OnCollisionEnter() OnCollisionStay() OnCollisionExit()	These method handle collisions and are called at different moments. They require a collider to be attached to them **with** the is trigger box **unticked.** **Enter**: called once when the 2 objects collide **Stay**: called every frame when the 2 objects remain in contact **Exit**: called once when the 2 objects stopped being in contact These methods also have a 2D counter part to them and only works if you use the 2D colliders You can also modify what objects can collide with each other by assigning it a layer and modify the layer's behaviour in the settings
OnTriggerEnter() OnTriggerStay() OnTriggerExit()	Same as the OnCollision counterparts, but they require a collider to be attached to them **with** the is trigger box **ticked.** It allows objects to pass through the trigger collider Only works for the for collision between it and a collider object that is not a trigger
OnEnable()	Method gets called once when the gameObject gets enabled
OnDisable()	Method gets called once when the gameObject gets disabled

Common Components

Transform

This component handles with the position, rotation and scale of the object. All GameObjects in the scene has this component.

Methods	Description