Unity C# Cheat Sheet and Quick Reference

MonoBehaviour Event Execution Order

Ordered by first to last method to execute.

```
private void Awake() { /* Called when the script is being loaded */ }
private void OnEnable() { /* Called every time the object is enabled */ }
private void Start() { /* Called on the frame when the script is enabled */ }
private void Update() { /* Called once per frame */ }
private void LateUpdate() { /* Called every frame after Update */ }
private void OnBecameVisible() { /* Called when the renderer is visible by any Camera */ }
private void OnBecameInvisible() { /* Called when the renderer is no longer visible by any Camera */ }
private void OnDrawGizmos() { /* Allows you to draw Gizmos in the Scene View */ }
private void OnGUI() { /* Called multiple times per frame in response to GUI events */ }
private void OnApplicationPause() { /* Called at the end of a frame when a pause is detected */ }
private void OnDisable() { /* Called every time the object is disabled */ }
private void OnDestroy() { /* Only called on previously active GameObjects that have been destroyed */}
```

Physics updates on a Fixed Timestep defined under *Edit* >> *Project Settings* >> *Time* >> *Fixed Timestep* and may execute more or less than once per actual frame.

```
private void FixedUpdate() { /* Called every Fixed Timestep */ }
```

See Physics Events section for a quick reference on associated Physics methods.

Physics Events

```
/* Both objects have to have a Collider and one object has to have a Rigidbody for these Events to work */
private void OnCollisionEnter(Collision hit) { Debug.Log(gameObject.name + " just hit " +
hit.gameObject.name); }
private void OnCollisionStay(Collision hit) { Debug.Log(gameObject.name + " is hitting " +
hit.gameObject.name); }
private void OnCollisionExit(Collision hit) { Debug.Log(gameObject.name + " stopped hitting " +
hit.gameObject.name); }

/* Trigger must be checked on one of the Colliders */
private void OnTriggerEnter(Collider hit) { Debug.Log(gameObject.name + " just hit " + hit.name); }
private void OnTriggerStay(Collider hit) { Debug.Log(gameObject.name + " is hitting " + hit.name); }
private void OnTriggerExit(Collider hit) { Debug.Log(gameObject.name + " stopped hitting " + hit.name); }

/* For 2D Colliders just add 2D to the Method name and the Parameter Type */
private void OnCollisionEnter2D(Collision2D hit) { }
private void OnCollisionStay2D(Collision2D hit) { }
private void OnCollisionExit2D(Collision2D hit) { }
private void OnTriggerEnter2D(Collider2D hit) { }
private void OnTriggerStay2D(Collider2D hit) { }
private void OnTriggerExit2D(Collider2D hit) { }
```

GameObject Manipulation

```
/* Create a GameObject */
Instantiate(GameObject prefab);
Instantiate(GameObject prefab, Transform parent);
Instantiate(GameObject prefab, Vector3 position, Quaternion rotation);
/* In Practice */
Instantiate(bullet);
Instantiate(bullet, bulletSpawn.transform);
Instantiate(bullet, Vector3.zero, Quaternion.identity);
Instantiate(bullet, new Vector3(0, 0, 10), bullet.transform.rotation);

/* Destroy a GameObject */
Destroy(gameObject);

/* Finding GameObjects */
GameObject myObj = GameObject.Find("NAME IN HIERARCHY");
GameObject myObj = GameObject.FindWithTag("TAG");

/* Accessing Components */
Example myComponent = GetComponent<Example>();
AudioSource audioSource = GetComponent<AudioSource>();
Rigidbody rgbd = GetComponent<Rigidbody>();
```

Vector Quick Reference

X = Left/Right Y = Up/Down Z = Forward/Back

```
Vector3.right /* (1, 0, 0) */   Vector2.right /* (1, 0) */
Vector3.left /* (-1, 0, 0) */   Vector2.left /* (-1, 0) */
Vector3.up /* (0, 1, 0) */      Vector2.up /* (0, 1) */
Vector3.down /* (0, -1, 0) */   Vector2.down /* (0, -1) */
Vector3.forward /* (0, 0, 1) */
Vector3.back /* (0, 0, -1) */
Vector3.zero /* (0, 0, 0) */    Vector2.zero /* (0, 0) */
Vector3.one /* (1, 1, 1) */     Vector2.one /* (1, 1) */
float length = myVector.magnitude /* Length of this Vector */
myVector.normalized /* Keeps direction, but reduces length to 1 */
```

Time Variables

```
/* The time in seconds since the start of the game */
float timeSinceStartOfGame = Time.time;

/* The scale at which the time is passing */
float currentTimeScale = Time.timeScale;
/* Pause time */
Time.timeScale = 0;

/* The time in seconds it took to complete the last frame */
/* Use with Update() and LateUpdate() */
float timePassedSinceLastFrame = Time.deltaTime;

/* The interval in seconds at which physics and fixed frame rate updates are performed */
/* Use with FixedUpdate() */
float physicsInterval =  Time.fixedDeltaTime;
```

Coroutines

```
/* Create a Coroutine */
private IEnumerator CountSeconds(int count = 10)
{
  for (int i = 0; i <= count; i++) {
    Debug.Log(i + " second(s) have passed");
    yield return new WaitForSeconds(1.0f);
  }
}

/* Call a Coroutine */
StartCoroutine(CountSeconds());
StartCoroutine(CountSeconds(10));

/* Call a Coroutine that may need to be stopped */
StartCoroutine("CountSeconds");
StartCoroutine("CountSeconds", 10);

/* Stop a Coroutine */
StopCoroutine("CountSeconds");
StopAllCoroutines();

/* Store and call a Coroutine from a variable */
private IEnumerator countSecondsCoroutine;

countSecondsCoroutine = CountSeconds();
StartCoroutine(countSecondsCoroutine);

/* Stop a stored Coroutine */
StopCoroutine(countSecondsCoroutine);
```

```
/* Coroutine Return Types */
yield return null; // Waits until the next Update() call
yield return new WaitForFixedUpdate(); // Waits until the next FixedUpdate() call
yield return new WaitForEndOfFrame(); // Waits until everything this frame has executed
yield return new WaitForSeconds(float seconds); // Waits for game time in seconds
yield return new WaitUntil(() => MY_CONDITION); // Waits until a custom condition is met
yield return new WWW("MY/WEB/REQUEST"); // Waits for a web request
yield return StartCoroutine("MY_COROUTINE"); // Waits until another Coroutine is completed
```

Hotkeys

Pan Tool	Move Tool	Rotate Tool	Scale Tool	Rect Tool
Q	*W*	*E*	*R*	*T*
New empty game object	**New child game object**	**Vertex Snap**	**Toggle Window Size**	**Duplicate**
Ctrl/Cmd+Shift+N	*Alt+Shift+N*	*Hold Ctrl/Cmd+Shift+V*	*Shift+Space*	*Ctrl/Cmd+D*
Play	**Pause**	**Step**	**Track game object in Scene window**	
Ctrl/Cmd+P	*Ctrl/Cmd+Shift+P*	*Ctrl/Cmd+Alt+P*	Select an object and press *Shift+F*	
Undo	**Redo**	**Redo Mac**	**Align Game camera with Scene camera**	
Ctrl/Cmd+Z	*Ctrl+Y*	*Cmd+Shift+Z*	Select a camera and press *Ctrl/Cmd+Shift+F*	

Input Quick Reference

```
if (Input.GetKeyDown(KeyCode.Space)) { Debug.Log("Space key was Pressed"); }
if (Input.GetKeyUp(KeyCode.W)) { Debug.Log("W key was Released"); }
if (Input.GetKey(KeyCode.UpArrow)) { Debug.Log("Up Arrow key is being held down"); }

/* Button Input located under Edit >> Project Settings >> Input */
if (Input.GetButtonDown("ButtonName")) { Debug.Log("Button was pressed"); }
if (Input.GetButtonUp("ButtonName")) { Debug.Log("Button was released"); }
if (Input.GetButton("ButtonName")) { Debug.Log("Button is being held down"); }
```