C#

(C Sharp)

Classes and Instances

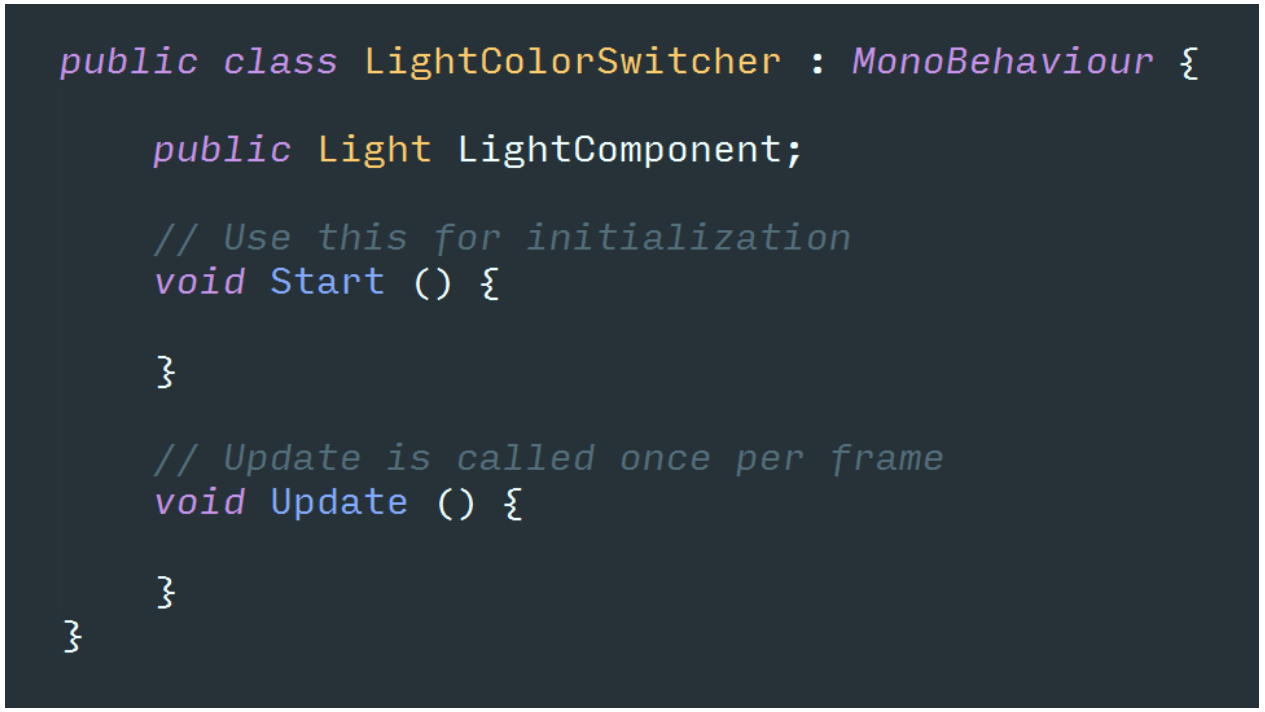
(OOP)

Classes

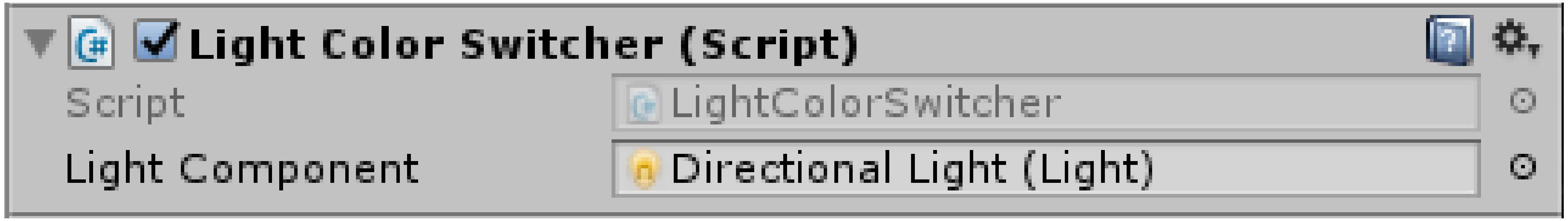
- Encapsulation: organize variables and functions together
- Nearly everything in C#/Unity is a class!

Accessing Components

Via Inspector



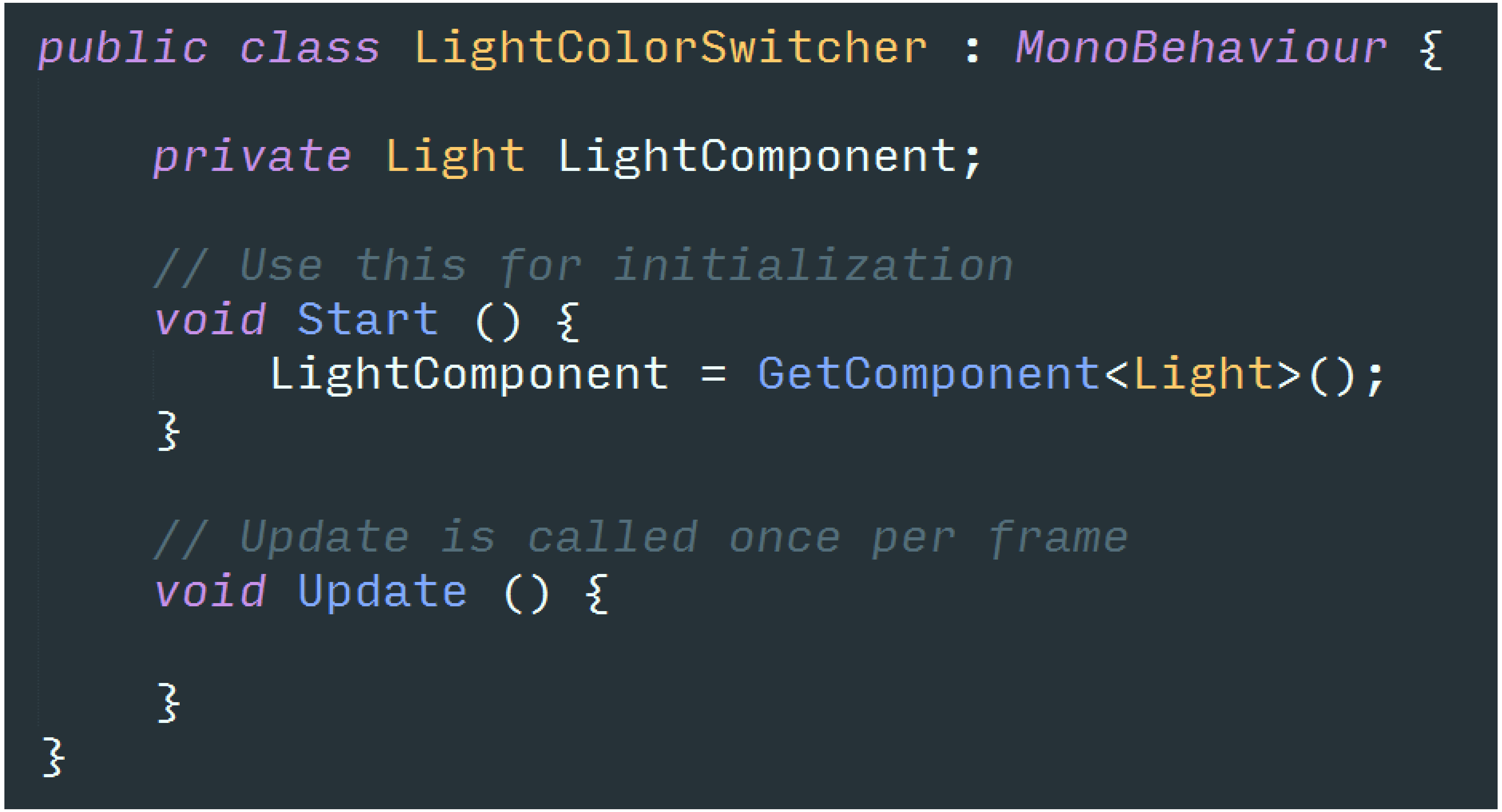
```
public class LightColorSwitcher : MonoBehaviour {

    public Light LightComponent;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```



Light Color Switcher (Script)

Script	LightColorSwitcher
Light Component	Directional Light (Light)

Via Scripting



```
public class LightColorSwitcher : MonoBehaviour {

    private Light LightComponent;

    // Use this for initialization
    void Start () {
        LightComponent = GetComponent<Light>();
    }

    // Update is called once per frame
    void Update () {

    }
}
```

Mathf

http://docs.unity3d.com/ScriptReference/Mathf.html

Mathf.Repeat

public static float **Repeat**(float **t**, float **length**);

Parameters

Description

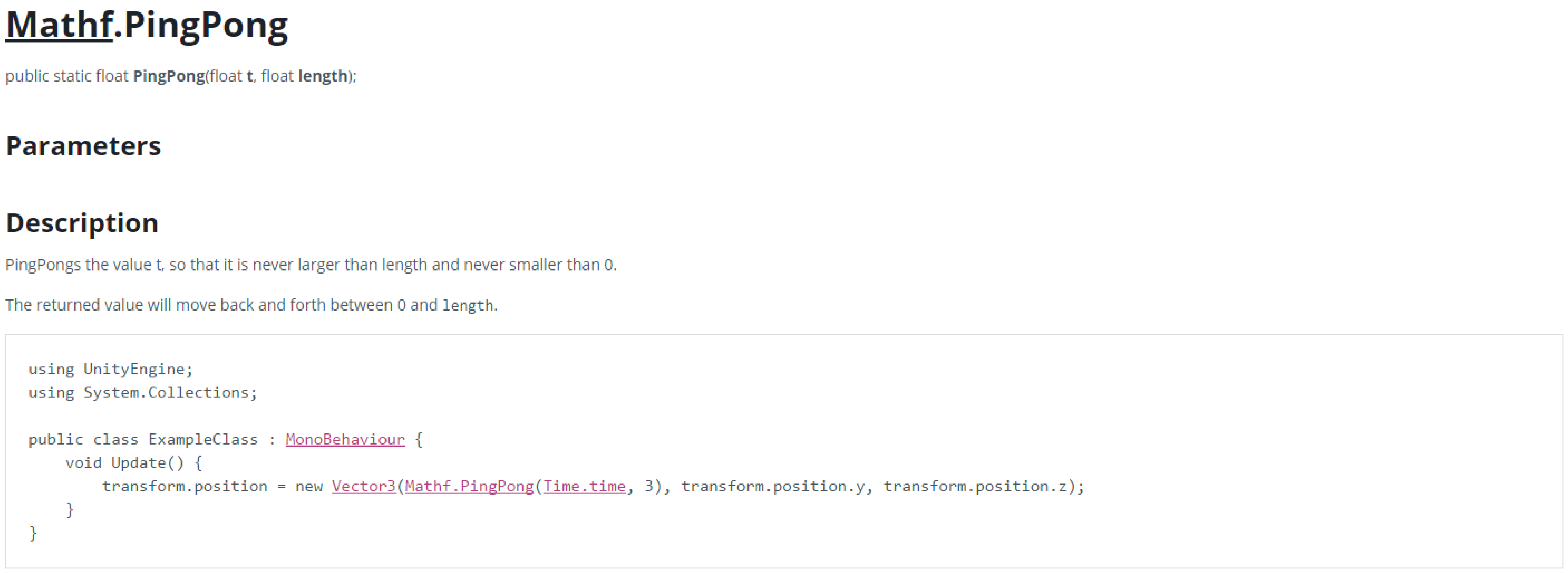
Loops the value t, so that it is never larger than length and never smaller than 0.

This is similar to the modulo operator but it works with floating point numbers. For example, using 3.0 for `t` and 2.5 for `length`, the result would be 0.5. With `t` = 5 and `length` = 2.5, the result would be 0.0. Note, however, that the behaviour is not defined for negative numbers as it is for the modulo operator.

In the example below the value of time is restricted between 0.0 and just under 3.0. This is then used to keep the x position in this range.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Update() {
        transform.position = new Vector3(Mathf.Repeat(Time.time, 3), transform.position.y, transform.position.z);
    }
}
```

http://docs.unity3d.com/ScriptReference/Mathf.Repeat.html



Mathf.PingPong

public static float **PingPong**(float **t**, float **length**);

Parameters

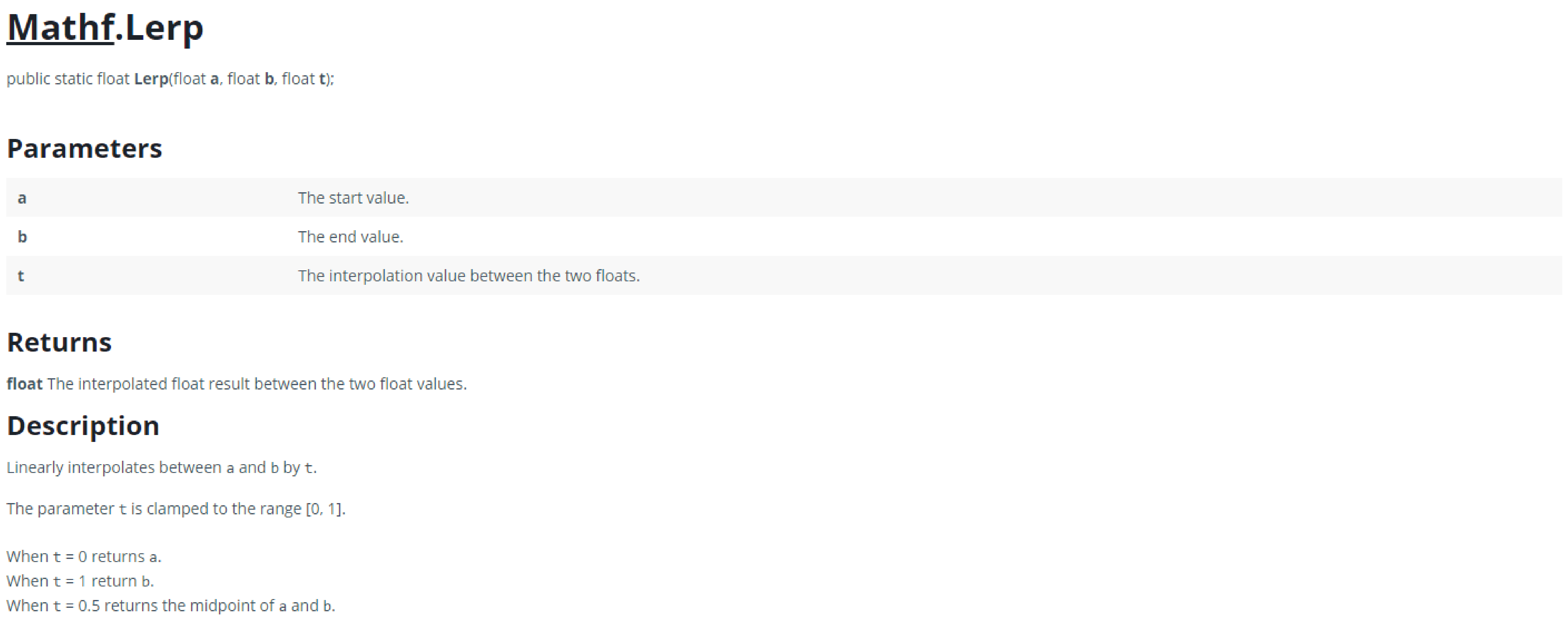
Description

PingPongs the value t, so that it is never larger than length and never smaller than 0.

The returned value will move back and forth between 0 and `length`.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Update() {
        transform.position = new Vector3(Mathf.PingPong(Time.time, 3), transform.position.y, transform.position.z);
    }
}
```

http://docs.unity3d.com/ScriptReference/Mathf.PingPong.html



Mathf.Lerp

public static float **Lerp**(float **a**, float **b**, float **t**);

Parameters

a	The start value.
b	The end value.
t	The interpolation value between the two floats.

Returns

float The interpolated float result between the two float values.

Description

Linearly interpolates between `a` and `b` by `t`.

The parameter `t` is clamped to the range [0, 1].

When `t` = 0 returns `a`.
When `t` = 1 return `b`.
When `t` = 0.5 returns the midpoint of `a` and `b`.

https://docs.unity3d.com/ScriptReference/Mathf.Lerp.html

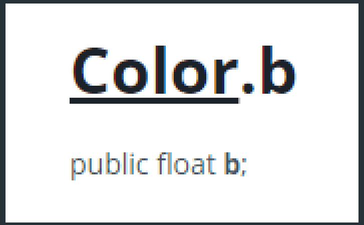
```csharp
// Using Mathf.PingPong to get intensities between 1 and 5
float duration = 2f; // Duration (in seconds) for the fade
float pongedTime = Mathf.PingPong(Time.time, duration); // Between 0 and duration
float lerpAmount = pongedTime / duration; // Between 0 and 1
float intensity = Mathf.Lerp(1, 5, lerpAmount); // Between 1 and 5
LightComponent.intensity = intensity;
```

```
// Using Mathf.PingPong to get lerped colors
float duration = 0.5f; // Duration (in seconds) for the fade
float pongedTime = Mathf.PingPong(Time.time, duration); // Between 0 and duration
float lerpAmount = pongedTime / duration; // Between 0 and 1
Color color = Color.Lerp(StartColor, EndColor, lerpAmount); // Between color 1 and color 2
LightComponent.color = color;
```

Static Classes & Methods

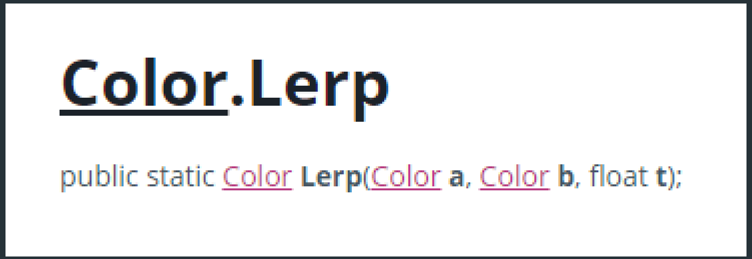
```
public static class AnimationUtilities {

    public static float MappedPingPong(float duration, float min, float max) {
        // Some code goes here!
    }

}
```

$$\texttt{float intensity = }\underbrace{\texttt{AnimationUtilities}}_{\text{CLASS NAME}}\texttt{.}\underbrace{\texttt{MappedPingPong}}_{\text{STATIC METHOD}}\texttt{(2f, 1f, 5f);}$$



Color.b

public float **b**;



Color.Lerp

public static Color **Lerp**(Color **a**, Color **b**, float **t**);

```
Color c = new Color(1f, 0f, 0f);
c.b = 1f;
c.g = 0.1f;
```

INSTANCE
FIELD

```
Color c1 = new Color(1f, 0f, 0f);
Color c2 = new Color(0f, 0f, 1f);
Color.Lerp(c1, c2, 0.5f);
```

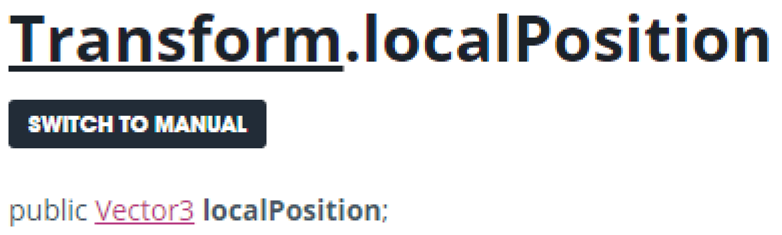
STATIC
METHOD

```
public static class AnimationUtilities {

    public static float MappedPingPong(float duration, float min, float max) {
        // Use time to find a ping pong value (between 0 and duration)
        float pingPongTime = Mathf.PingPong(Time.time, duration);
        // Now, we want a value between 0 and 1 - so divide by duration
        float lerpAmount = pingPongTime / duration;
        // Use the value between 0 and 1 to find a value between min and max
        float mappedValue = Mathf.Lerp(min, max, lerpAmount);
        return mappedValue;
    }

}
```

Vector3

http://docs.unity3d.com/ScriptReference/Vector3.html

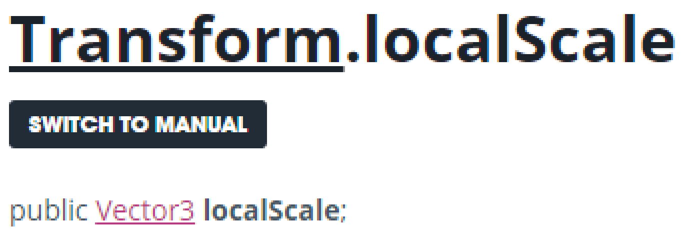
Transform



Transform.localPosition

SWITCH TO MANUAL

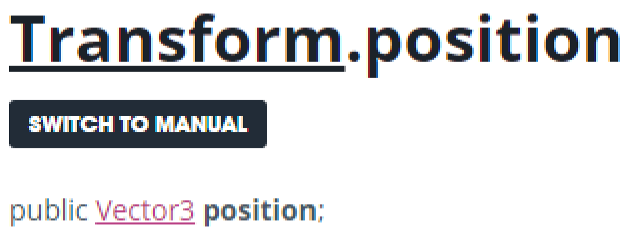
public Vector3 **localPosition**;



Transform.localScale

SWITCH TO MANUAL

public Vector3 **localScale**;



Transform.position

SWITCH TO MANUAL

public Vector3 **position**;

https://docs.unity3d.com/ScriptReference/Transform.html

Vector3

struct in UnityEngine

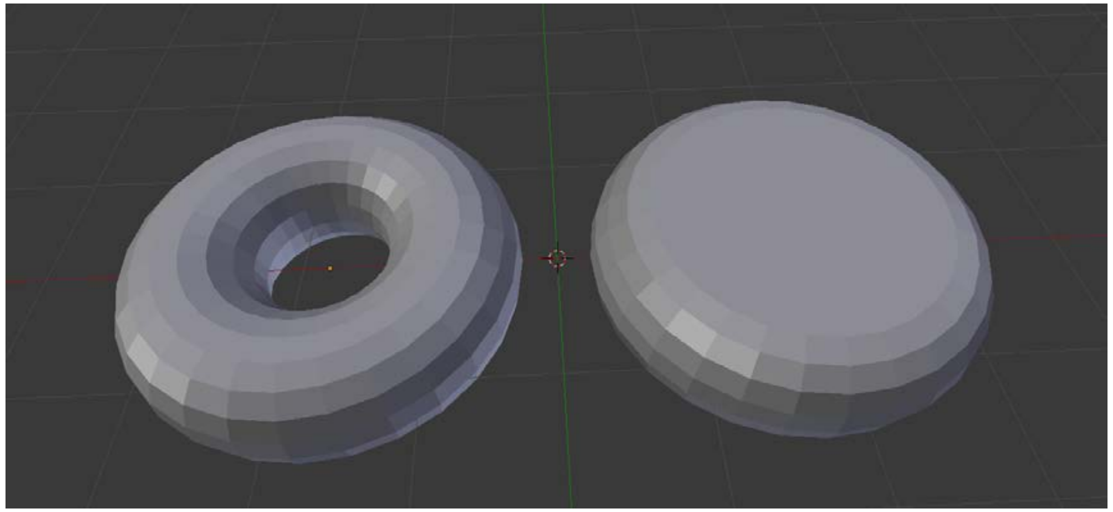
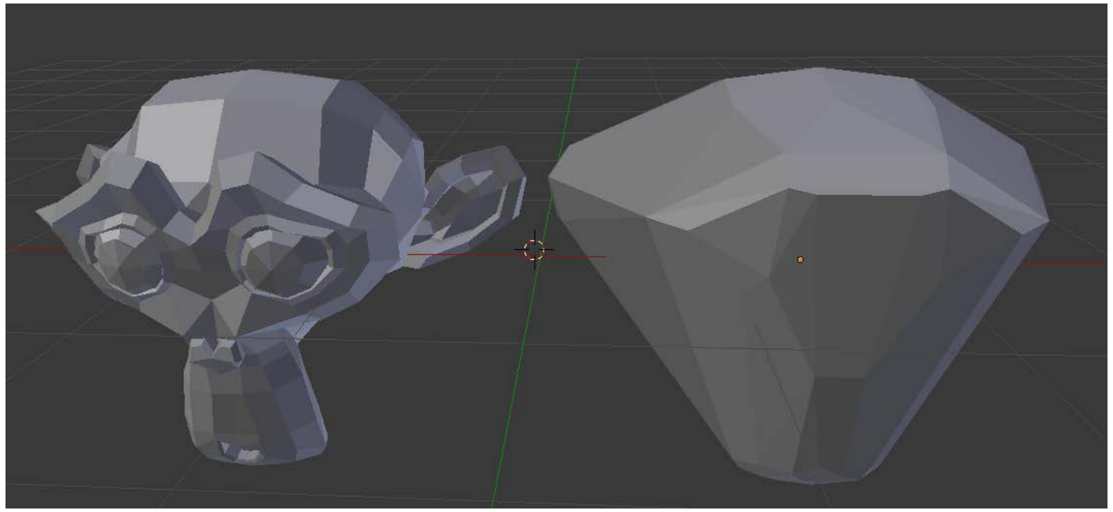
Description

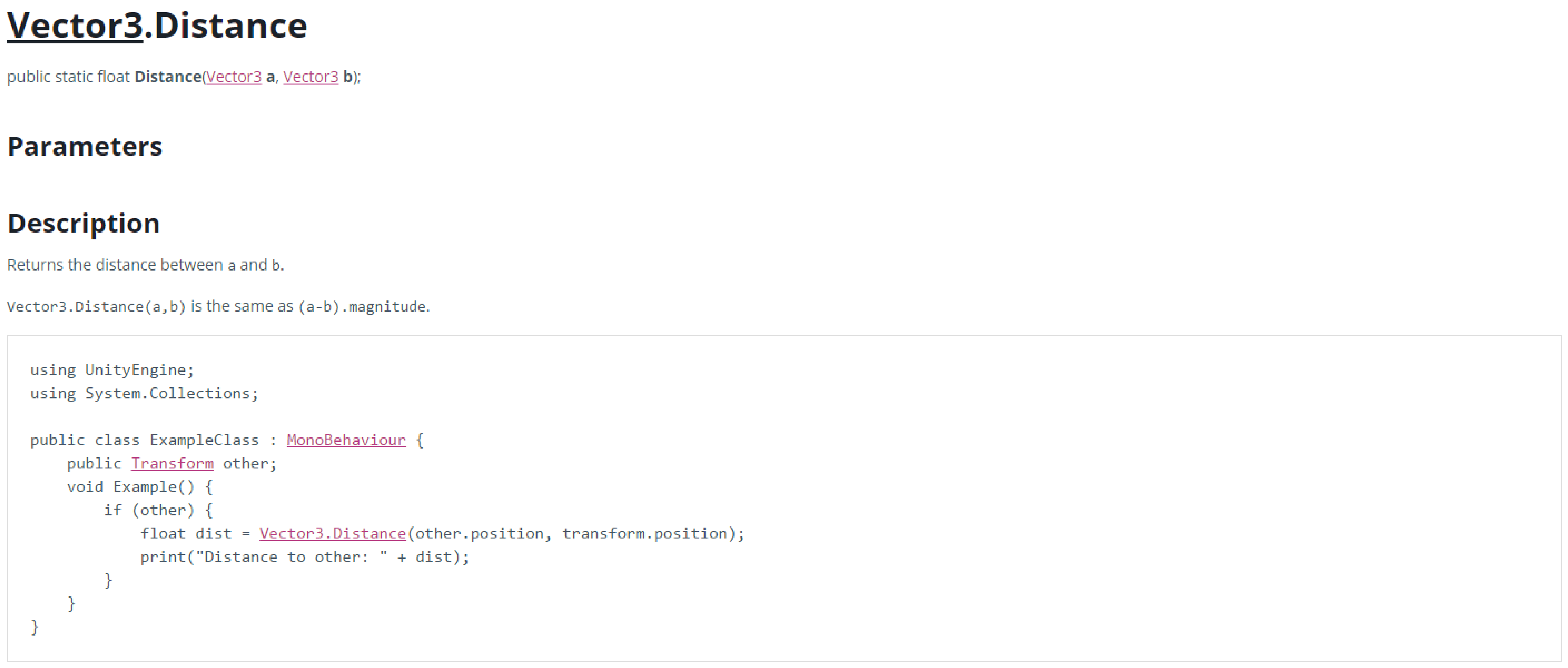
Representation of 3D vectors and points.

This structure is used throughout Unity to pass 3D positions and directions around. It also contains functions for doing common vector operations.

```
Vector3 position = new Vector3(0f, 0f, 1f);
```

Convex Hull



Vector3.Distance

public static float **Distance**(Vector3 **a**, Vector3 **b**);

Parameters

Description

Returns the distance between `a` and `b`.

`Vector3.Distance(a,b)` is the same as `(a-b).magnitude`.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public Transform other;
    void Example() {
        if (other) {
            float dist = Vector3.Distance(other.position, transform.position);
            print("Distance to other: " + dist);
        }
    }
}
```

http://docs.unity3d.com/ScriptReference/Vector3.Distance.html

```
public class DistanceDemo : MonoBehaviour {

    public Transform PlayerTransform;

    void Update () {

        // Find the distance
        float distance = Vector3.Distance(PlayerTransform.position, transform.position);

        // Check how this object is to the player
        if (distance <= 3f) {
            Debug.Log("Player is close!");
        } else {
            Debug.Log("Player is far!");
        }
    }
}
```

Random

https://docs.unity3d.com/ScriptReference/Random.html

Random.Range

public static float **Range**(float **min**, float **max**);

Parameters

Description

Returns a random float number between and `min` [inclusive] and `max` [inclusive] (Read Only).

Note that `max` is inclusive, so using Random.Range(0.0f, 1.0f) could return 1.0 as a value.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    public GameObject prefab;

    // Instantiate the prefab somewhere between -10.0 and 10.0 on the x-z plane
    void Start()
    {
        Vector3 position = new Vector3(Random.Range(-10.0f, 10.0f), 0, Random.Range(-10.0f, 10.0f));
        Instantiate(prefab, position, Quaternion.identity);
    }
}
```

Random.rotationUniform

public static Quaternion **rotationUniform**;

Description

Returns a random rotation with uniform distribution (Read Only).

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Example() {
        transform.rotation = Random.rotationUniform;
    }
}
```

Random.ColorHSV

```
public static Color ColorHSV();
public static Color ColorHSV(float hueMin, float hueMax);
public static Color ColorHSV(float hueMin, float hueMax, float saturationMin, float saturationMax);
public static Color ColorHSV(float hueMin, float hueMax, float saturationMin, float saturationMax, float valueMin, float valueMax);
public static Color ColorHSV(float hueMin, float hueMax, float saturationMin, float saturationMax, float valueMin, float valueMax, float alphaMin, float alphaMax);
```

http://alloyui.com/examples/color-picker/hsv/

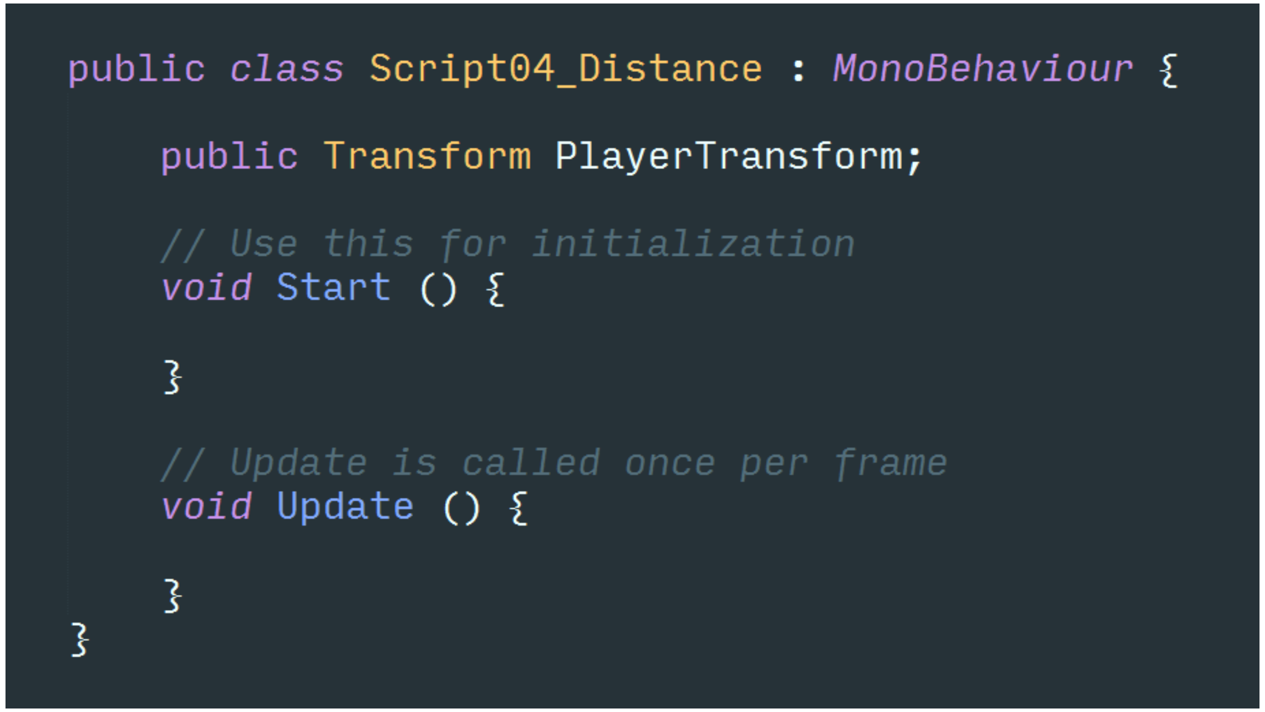
MeshRenderer & Material

http://docs.unity3d.com/ScriptReference/MeshRenderer.html
http://docs.unity3d.com/ScriptReference/Material.html

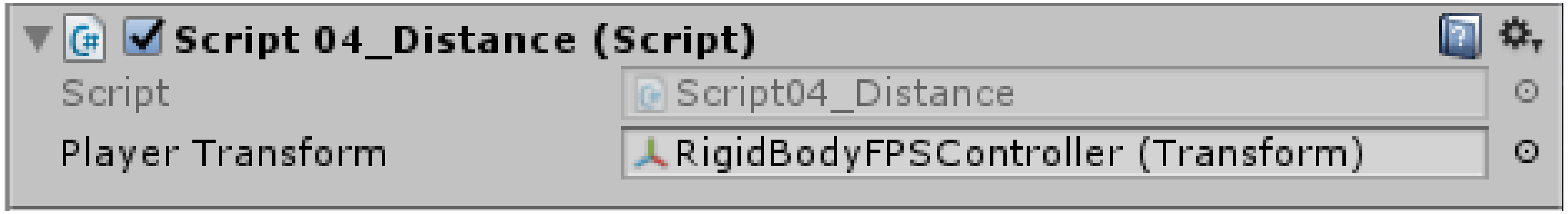
```
public class GettingMaterial : MonoBehaviour {

    private MeshRenderer renderer;
    private Material mat;

    void Start () {
        // Get the MeshRenderer on this game object
        renderer = GetComponent<MeshRenderer>();
        // Get the first material from the renderer
        mat = renderer.material;
        // Change the material's color to red
        mat.color = new Color(1f, 0f, 0f);
    }
}
```

Accessing Other Game Objects

Via Inspector



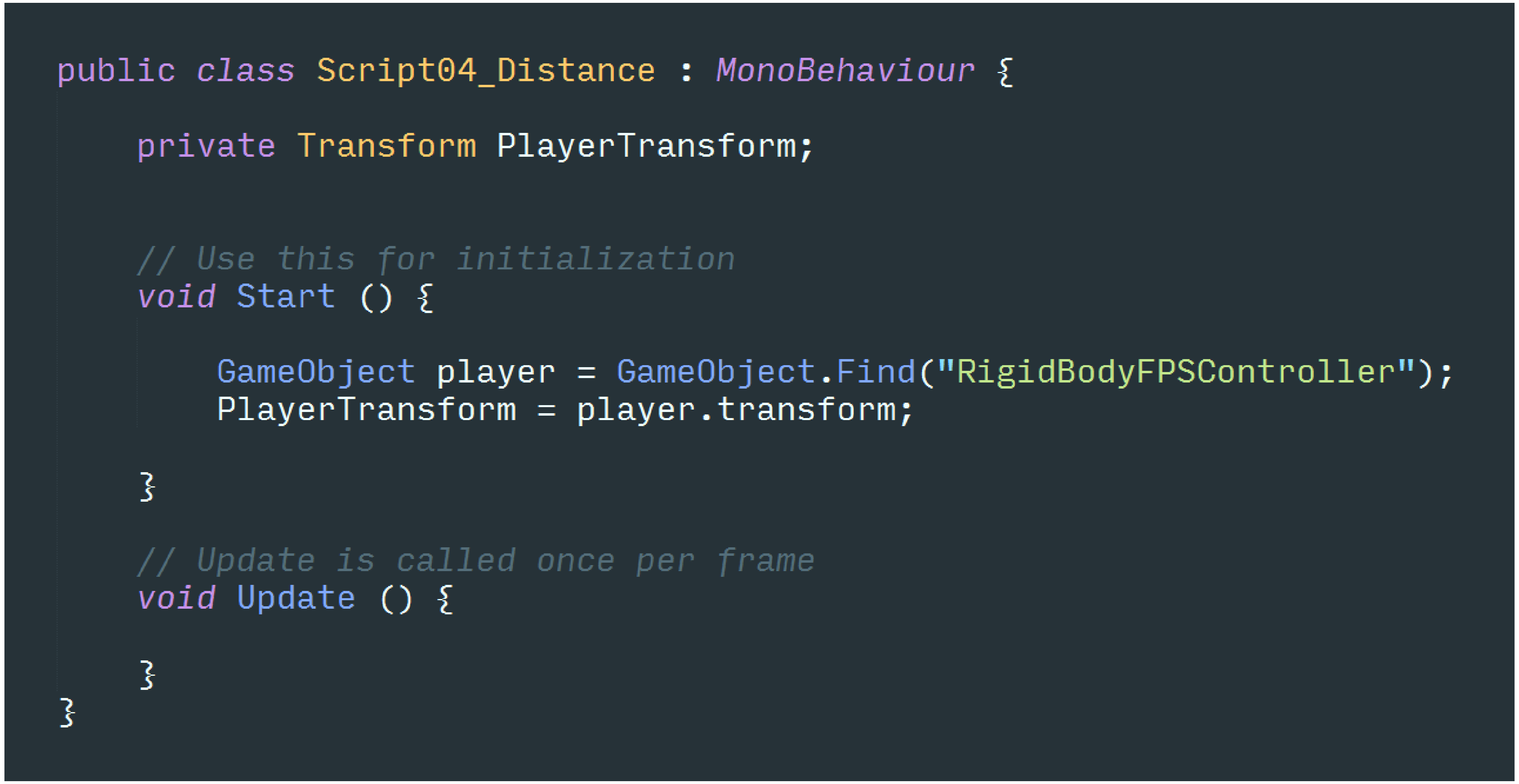
```
public class Script04_Distance : MonoBehaviour {

    public Transform PlayerTransform;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```



Script 04_Distance (Script)

Script	Script04_Distance
Player Transform	RigidBodyFPSController (Transform)

Via Scripting



```csharp
public class Script04_Distance : MonoBehaviour {

    private Transform PlayerTransform;


    // Use this for initialization
    void Start () {

        GameObject player = GameObject.Find("RigidBodyFPSController");
        PlayerTransform = player.transform;

    }

    // Update is called once per frame
    void Update () {

    }
}
```

https://docs.unity3d.com/ScriptReference/GameObject.Find.html