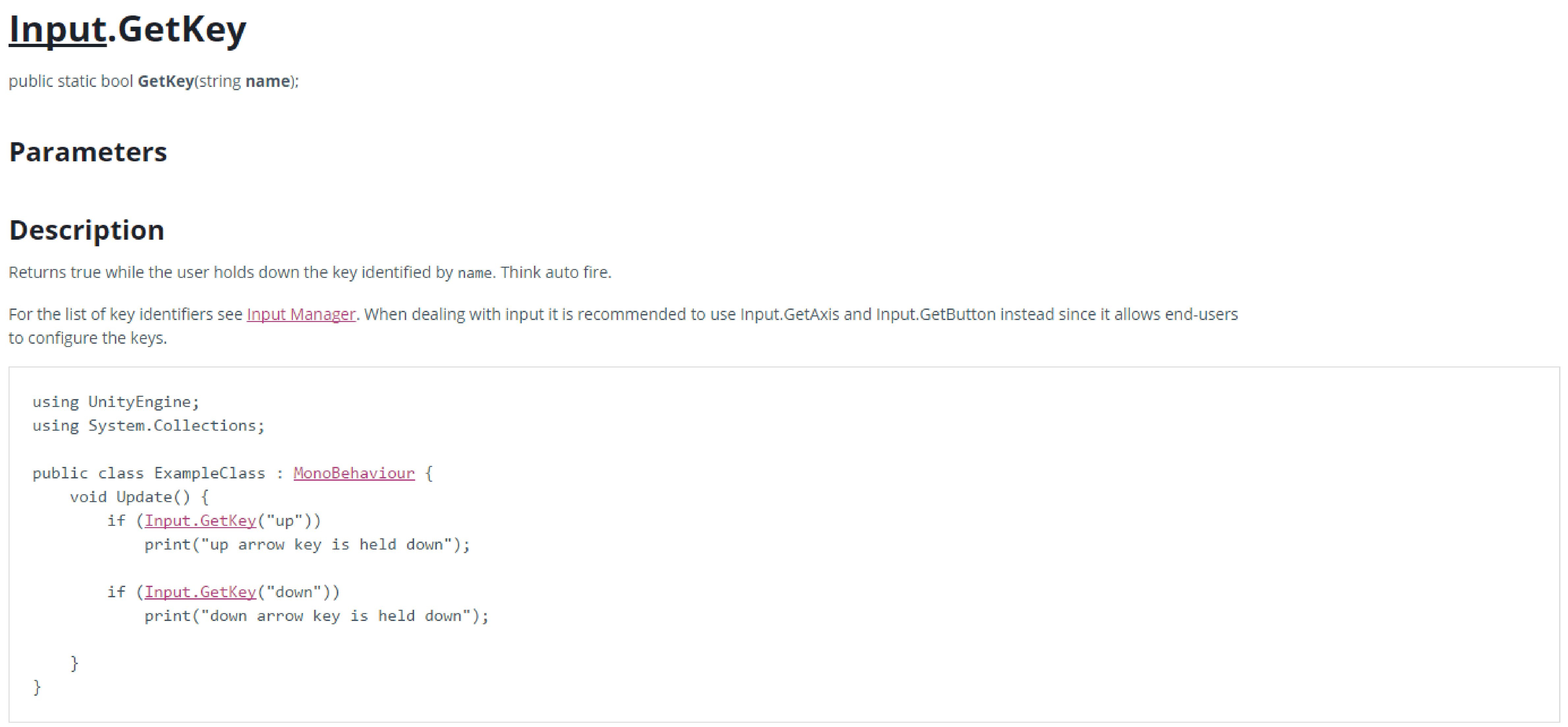
C#

(C Sharp)

Getting Input

(Quick Way)



Input.GetKey

public static bool **GetKey**(string **name**);

Parameters

Description

Returns true while the user holds down the key identified by `name`. Think auto fire.

For the list of key identifiers see Input Manager. When dealing with input it is recommended to use Input.GetAxis and Input.GetButton instead since it allows end-users to configure the keys.

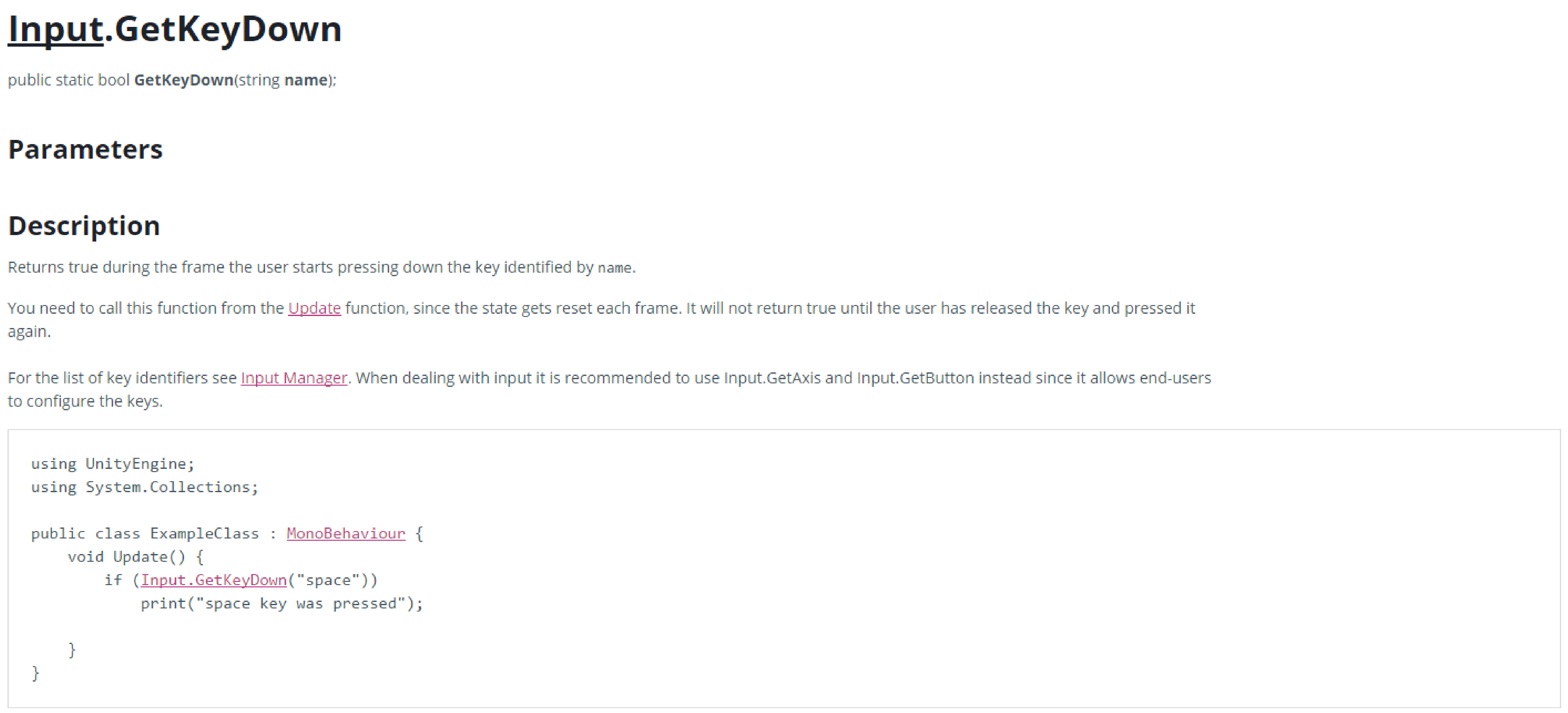
```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Update() {
        if (Input.GetKey("up"))
            print("up arrow key is held down");

        if (Input.GetKey("down"))
            print("down arrow key is held down");

    }
}
```

https://docs.unity3d.com/ScriptReference/Input.GetKey.html



Input.GetKeyDown

public static bool **GetKeyDown**(string **name**);

Parameters

Description

Returns true during the frame the user starts pressing down the key identified by `name`.

You need to call this function from the Update function, since the state gets reset each frame. It will not return true until the user has released the key and pressed it again.

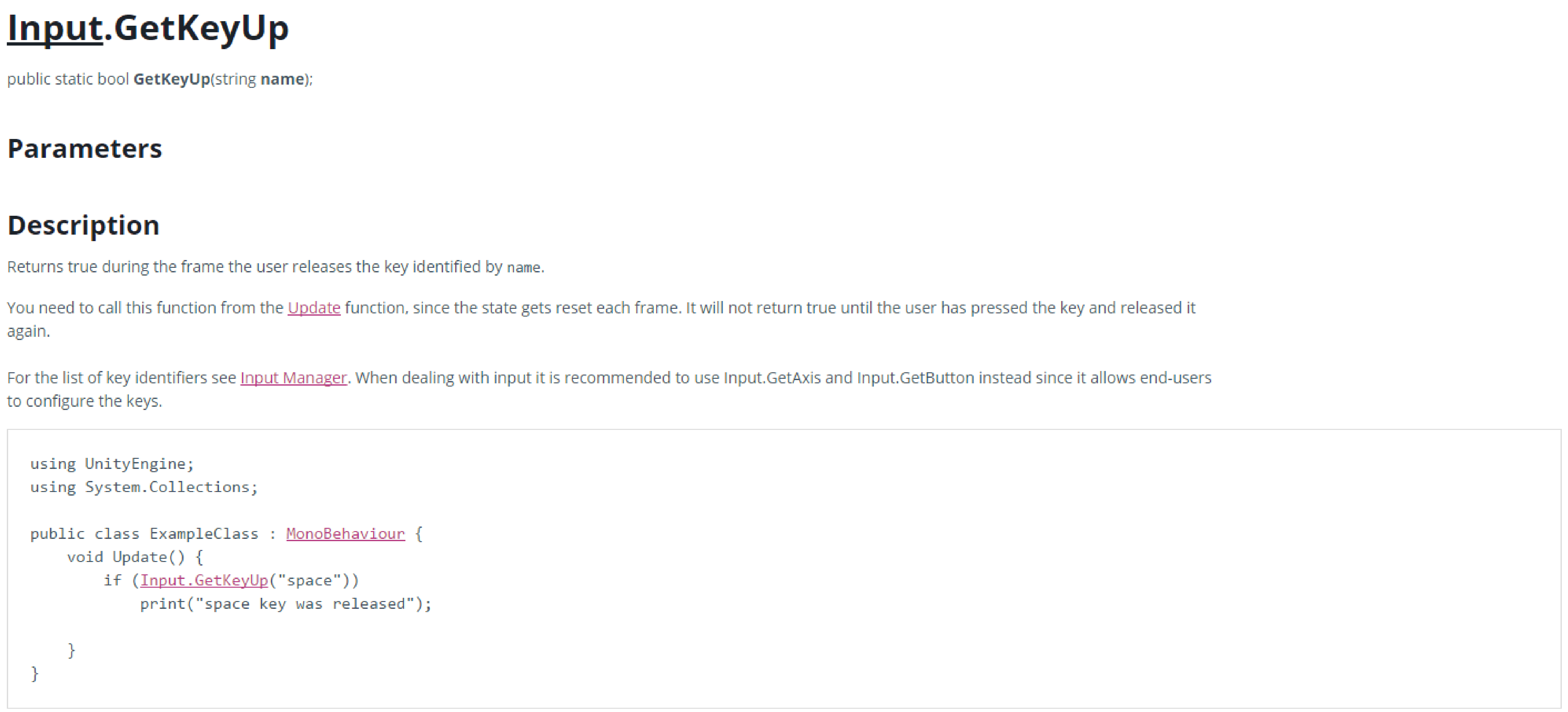
For the list of key identifiers see Input Manager. When dealing with input it is recommended to use Input.GetAxis and Input.GetButton instead since it allows end-users to configure the keys.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Update() {
        if (Input.GetKeyDown("space"))
            print("space key was pressed");

    }
}
```

https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html



Input.GetKeyUp

public static bool **GetKeyUp**(string **name**);

Parameters

Description

Returns true during the frame the user releases the key identified by `name`.

You need to call this function from the Update function, since the state gets reset each frame. It will not return true until the user has pressed the key and released it again.

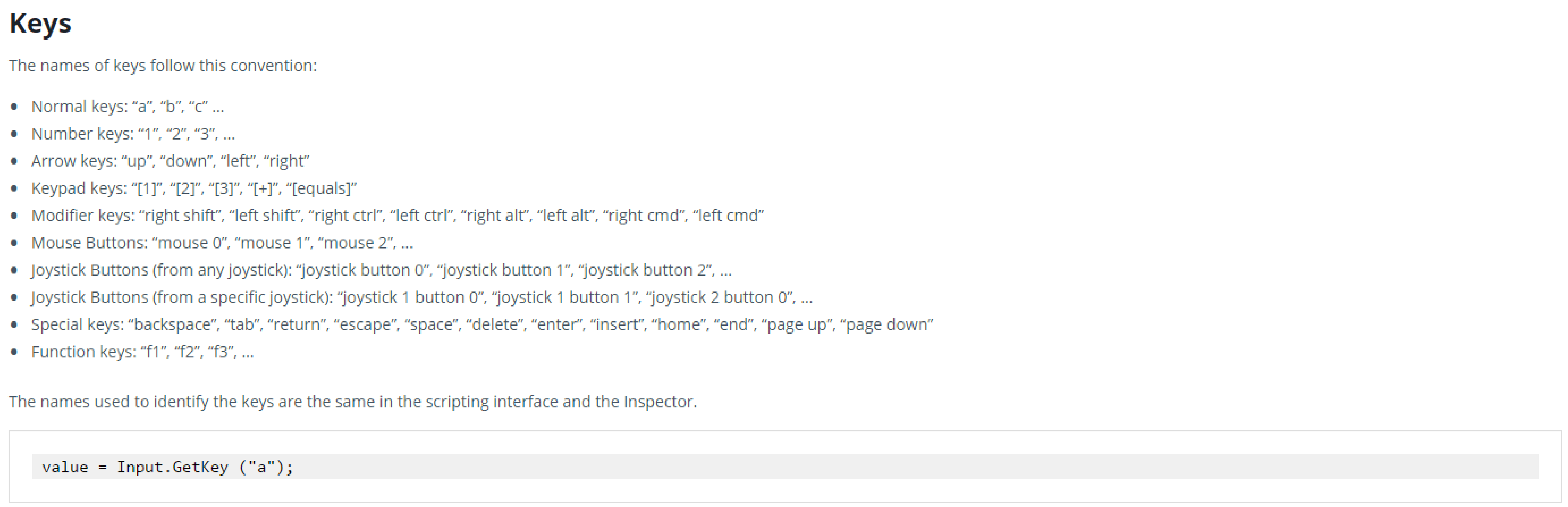
For the list of key identifiers see Input Manager. When dealing with input it is recommended to use Input.GetAxis and Input.GetButton instead since it allows end-users to configure the keys.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    void Update() {
        if (Input.GetKeyUp("space"))
            print("space key was released");

    }
}
```

https://docs.unity3d.com/ScriptReference/Input.GetKeyUp.html



Keys

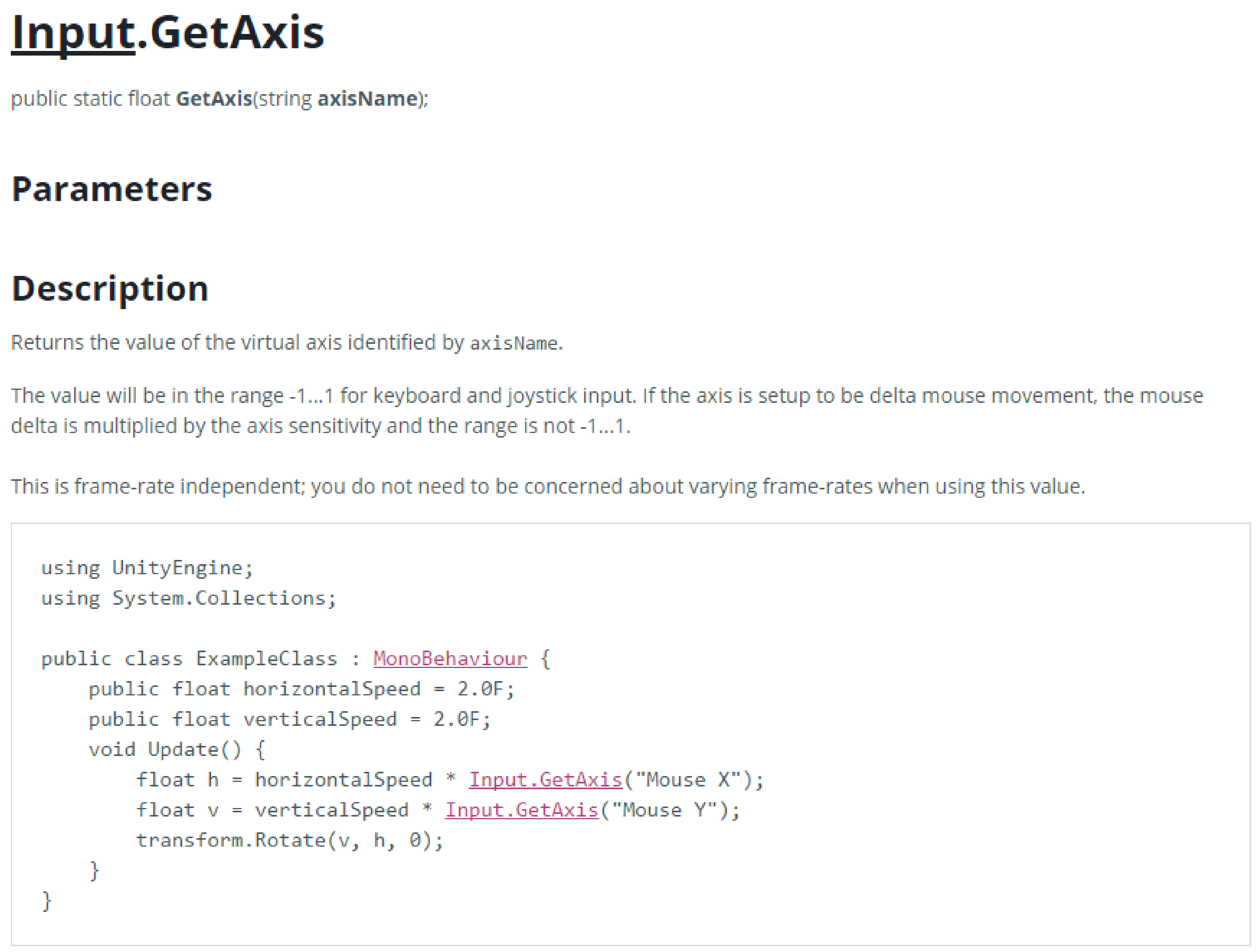
The names of keys follow this convention:

- Normal keys: “a”, “b”, “c” ...
- Number keys: “1”, “2”, “3”, ...
- Arrow keys: “up”, “down”, “left”, “right”
- Keypad keys: “[1]”, “[2]”, “[3]”, “[+]”, “[equals]”
- Modifier keys: “right shift”, “left shift”, “right ctrl”, “left ctrl”, “right alt”, “left alt”, “right cmd”, “left cmd”
- Mouse Buttons: “mouse 0”, “mouse 1”, “mouse 2”, ...
- Joystick Buttons (from any joystick): “joystick button 0”, “joystick button 1”, “joystick button 2”, ...
- Joystick Buttons (from a specific joystick): “joystick 1 button 0”, “joystick 1 button 1”, “joystick 2 button 0”, ...
- Special keys: “backspace”, “tab”, “return”, “escape”, “space”, “delete”, “enter”, “insert”, “home”, “end”, “page up”, “page down”
- Function keys: “f1”, “f2”, “f3”, ...

The names used to identify the keys are the same in the scripting interface and the Inspector.

```
value = Input.GetKey ("a");
```

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



Input.GetAxis

public static float **GetAxis**(string **axisName**);

Parameters

Description

Returns the value of the virtual axis identified by `axisName`.

The value will be in the range -1...1 for keyboard and joystick input. If the axis is setup to be delta mouse movement, the mouse delta is multiplied by the axis sensitivity and the range is not -1...1.

This is frame-rate independent; you do not need to be concerned about varying frame-rates when using this value.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public float horizontalSpeed = 2.0F;
    public float verticalSpeed = 2.0F;
    void Update() {
        float h = horizontalSpeed * Input.GetAxis("Mouse X");
        float v = verticalSpeed * Input.GetAxis("Mouse Y");
        transform.Rotate(v, h, 0);
    }
}
```

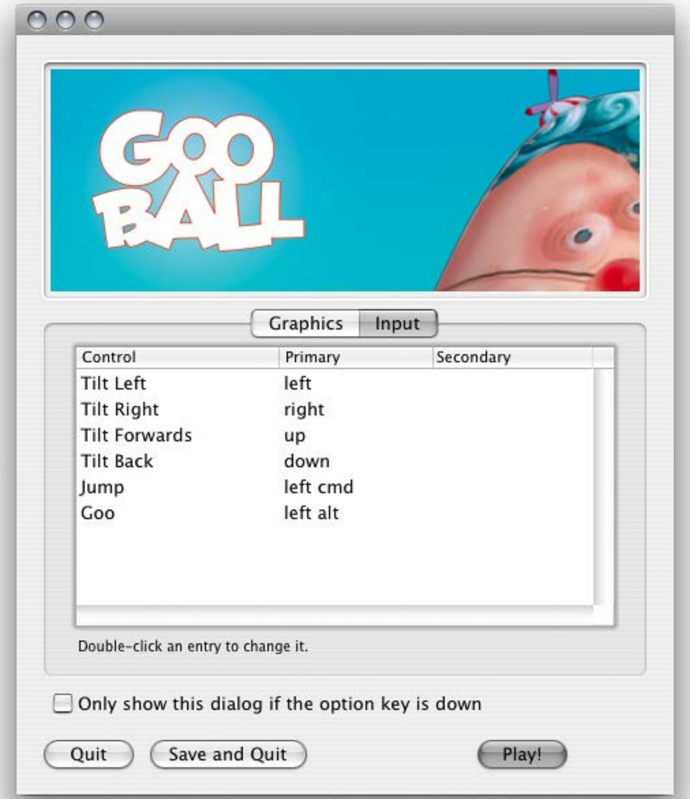
https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

More Mouse Inputs

- Input.GetMouseButton
- Input.GetMouseButtonDown
- Input.GetMouseButtonUp

Customizable Input

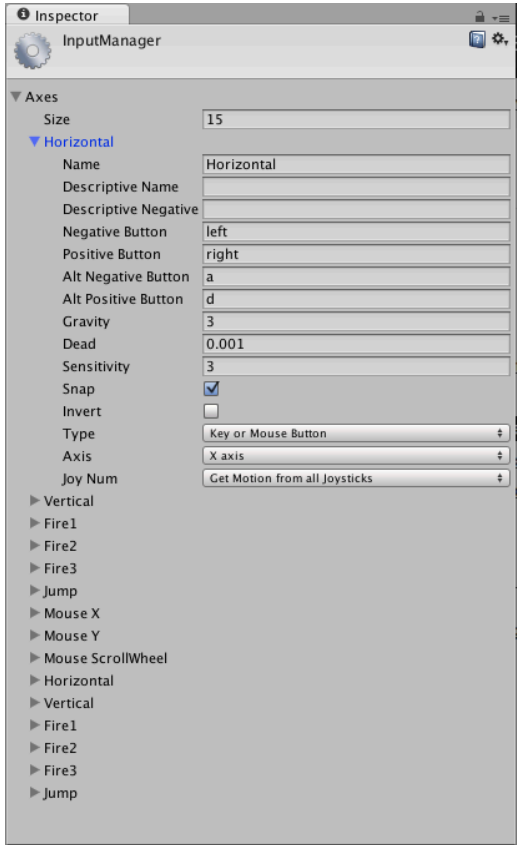
See https://docs.unity3d.com/Manual/Input.html



Control	Primary	Secondary
Tilt Left	left	
Tilt Right	right	
Tilt Forwards	up	
Tilt Back	down	
Jump	left cmd	
Goo	left alt	

Double-click an entry to change it.

☐ Only show this dialog if the option key is down



Axes

Size	15

Horizontal

Name	Horizontal
Descriptive Name	
Descriptive Negative	
Negative Button	left
Positive Button	right
Alt Negative Button	a
Alt Positive Button	d
Gravity	3
Dead	0.001
Sensitivity	3
Snap	✓
Invert	☐
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

- Vertical
- Fire1
- Fire2
- Fire3
- Jump
- Mouse X
- Mouse Y
- Mouse ScrollWheel
- Horizontal
- Vertical
- Fire1
- Fire2
- Fire3
- Jump

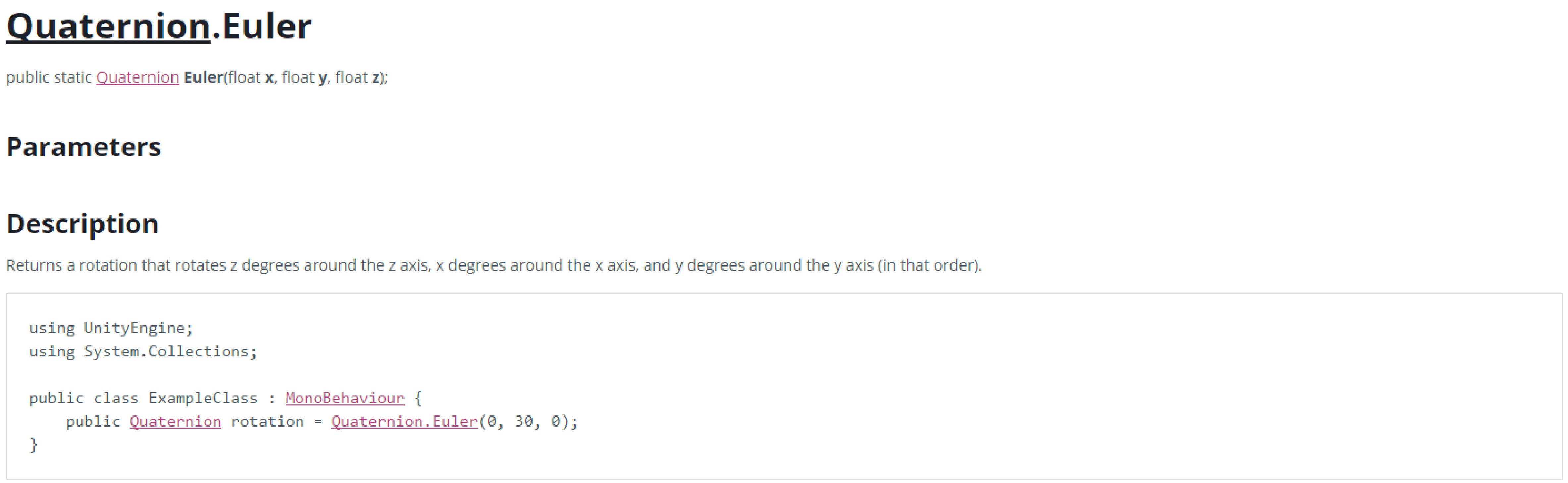
Euler vs Quaternions

Euler Rotation

```
float horizontalMovement = Input.GetAxis("Mouse X");
float verticalMovement = Input.GetAxis("Mouse Y");

// Wrong way to rotate along two axes! Don't do this.
transform.Rotate(0, horizontalMovement, 0);
transform.Rotate(-verticalMovement, 0, 0);
```

Gimbal Lock
in 30 seconds



Quaternion.Euler

public static Quaternion **Euler**(float **x**, float **y**, float **z**);

Parameters

Description

Returns a rotation that rotates z degrees around the z axis, x degrees around the x axis, and y degrees around the y axis (in that order).

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public Quaternion rotation = Quaternion.Euler(0, 30, 0);
}
```

https://docs.unity3d.com/ScriptReference/Quaternion.Euler.html

Quaternion Rotation

```
// Rotating with quaternions - much better!
transform.localRotation = Quaternion.Euler(45f, 20f, 0f);
```

```
// Exercise:
//
//  - Add forward/backward movement with the w and s keys
//  - Add strafing movement (left/right) with the a and d keys
//  - Add vertical movement (up/down) with the q and e keys
//  - Bonus: add the ability to hold shift to move at 2x speed
//
// Note: Use a public field for Speed, so these movement speeds can be
// adjusted in the inspector. Speed should be in meters/second.
```

Classes and Instances

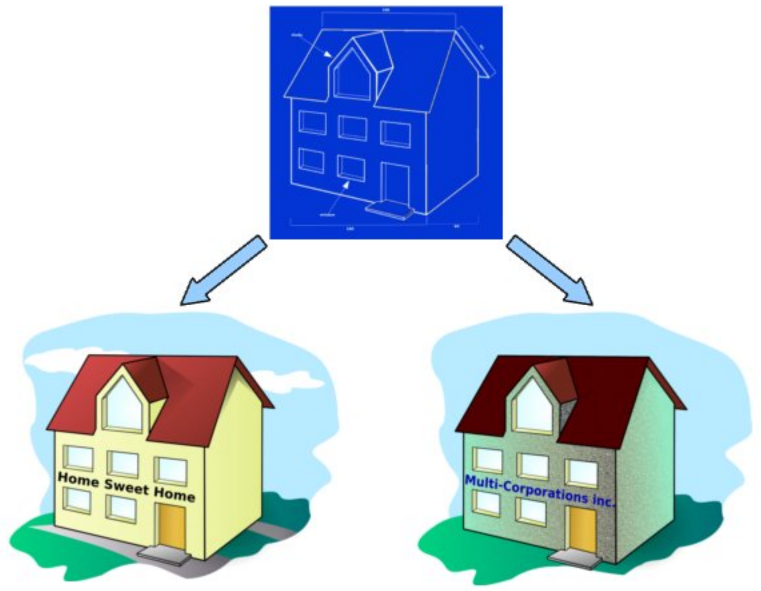
(OOP)

Classes & Instances

- Encapsulation: organize variables and functions together
- Code reuse

Analogy Time

- Blueprint -> House
- Cookie Cutter -> Cookie
- Person -> Bob



http://processing.lyndondaniels.com/53blueprint.php

Class

ACCESS MODIFIER

CLASS NAME

```java
public class Enemy {
    // Fields and methods go inside brackets
}
```

Fields

Fields

```
public class Enemy {
    // Fields
    public string Name;
}
```

$$\underbrace{\texttt{public}}_{\text{ACCESS MODIFIER}}\ \underbrace{\texttt{string}}_{\text{VARIABLE TYPE}}\ \underbrace{\texttt{Name}}_{\text{VARIABLE NAME}}\texttt{;}$$

```
public class ClassDemo : MonoBehaviour {

    void Start () {
        Enemy enemy1 = new Enemy();
    }

}

public class Enemy {
    // Fields
    public string Name;
}
```

INSTANCE

CLASS

$$\underbrace{\texttt{Enemy}}_{\text{VARIABLE TYPE}}\ \overbrace{\texttt{enemy1}}^{\text{VARIABLE NAME}}\ \texttt{=}\ \underbrace{\texttt{new}}_{\text{KEYWORD}}\ \overbrace{\texttt{Enemy()}}^{\text{CONSTRUCTOR}}\texttt{;}$$

INSTANCE

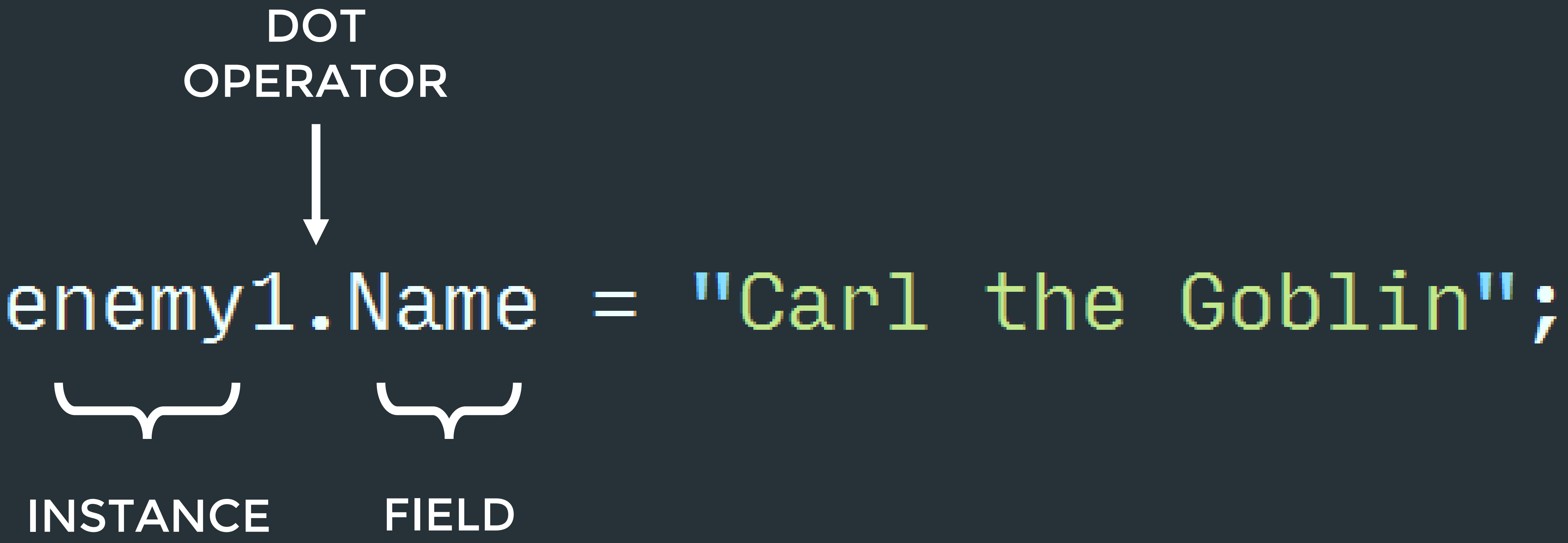
```
public class ClassDemo : MonoBehaviour {

    void Start () {
        Enemy enemy1 = new Enemy();
        enemy1.Name = "Carl The Goblin";
        Debug.Log("Monster 1 is: " + enemy1.Name);
    }

}
```

CLASS

```
public class Enemy {
    // Fields
    public string Name;
}
```



```
enemy1.Name = "Carl the Goblin";
```

```csharp
public class ClassDemo : MonoBehaviour {

    void Start () {
        Enemy enemy1 = new Enemy();
        enemy1.Name = "Carl The Goblin";
        Debug.Log("Monster 1 is: " + enemy1.Name);

        Enemy enemy2 = new Enemy();
        enemy2.Name = "Radcliff";
        Debug.Log("Monster 2 is: " + enemy2.Name);
    }

}

public class Enemy {
    // Fields
    public string Name;
}
```

INSTANCE

INSTANCE

CLASS

Constructors

```
public class Enemy {
    // Fields
    public string Name;

    // Constructor
    public Enemy(string name) {
        Name = name;
    }
}
```

```
public class ClassDemo : MonoBehaviour {
    void Start () {
        Enemy enemy1 = new Enemy("Carl The Goblin");
        Debug.Log("Enemy 1 is: " + enemy1.Name);
    }
}

public class Enemy {
    // Fields
    public string Name;

    // Constructor
    public Enemy(string name) {
        Name = name;
    }
}
```

Methods

```csharp
public class Enemy {
    // Fields
    public string Name;

    // Constructor
    public Enemy(string name) {
        Name = name;
    }

    // Methods
    public void Speak() {
        Debug.Log("Hello, I am " + Name + ".");
    }
}
```

```
Enemy enemy1 = new Enemy("Carl The Goblin");
enemy1.Speak();
```