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);
    }
}
```

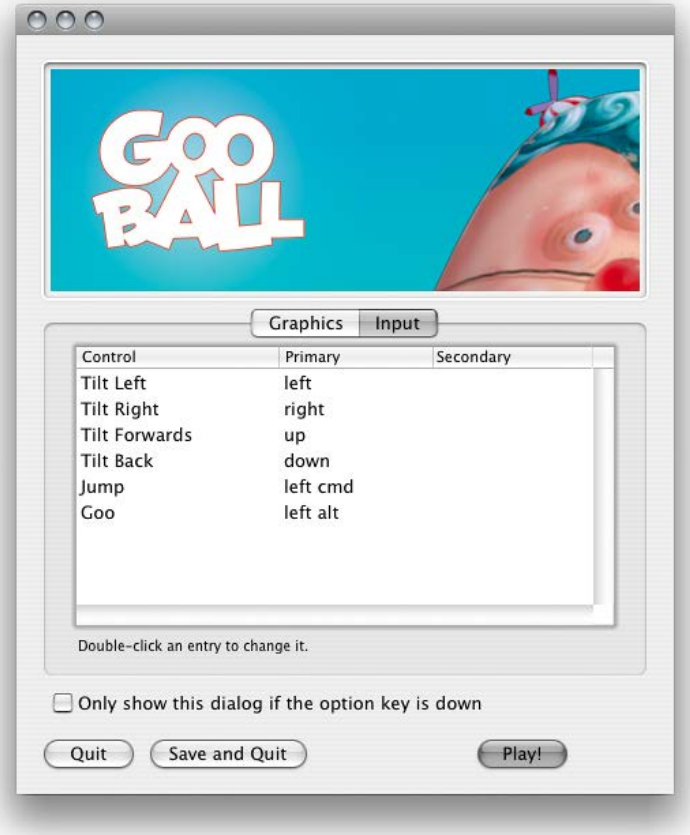
https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

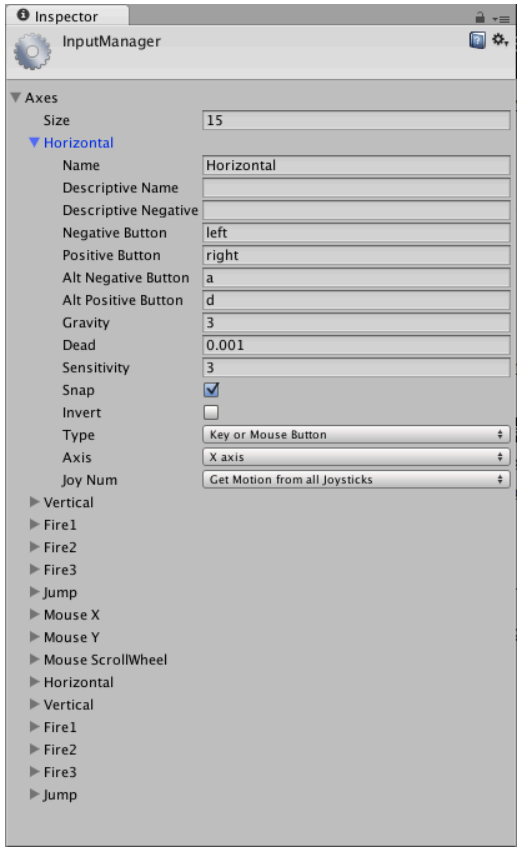
More Mouse Inputs

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

Customizable Input

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





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

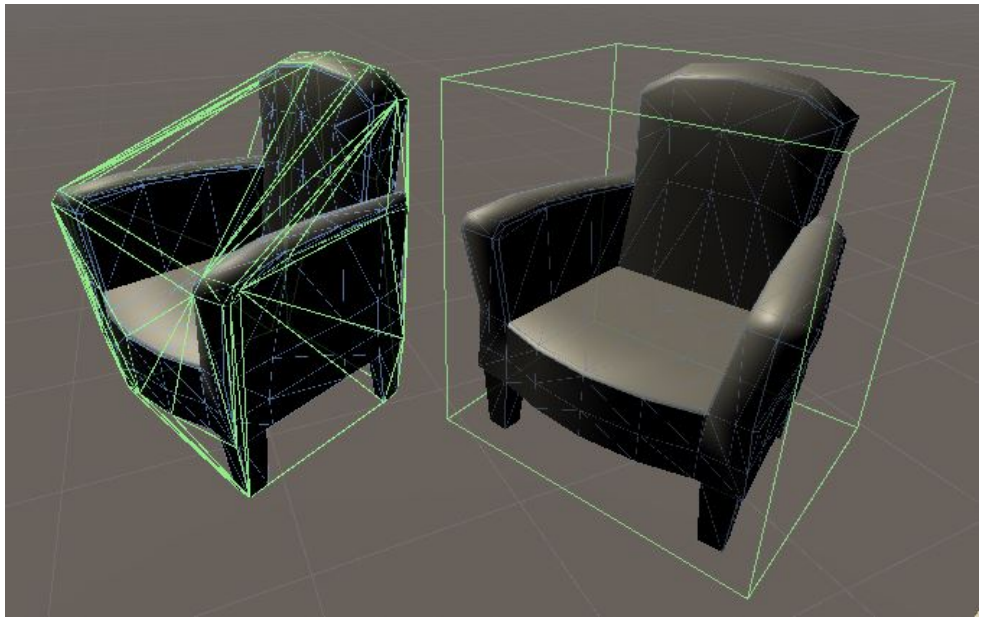
Quaternion Rotation

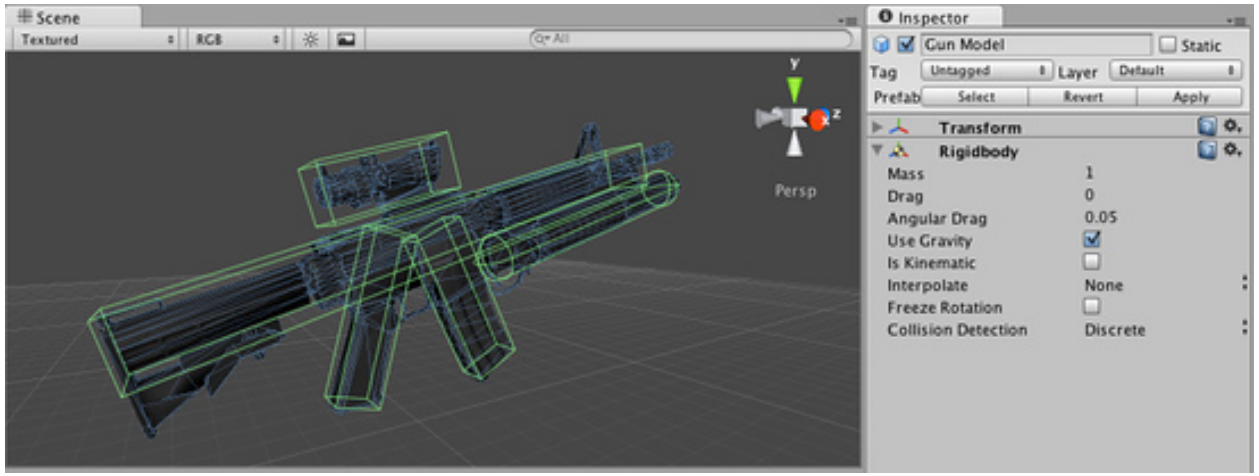
```
// Rotating with quaternions - much better!
transform.localRotation = Quaternion.Euler(45f, 20f, 0f);
```

Physics & 3D Models

Collider

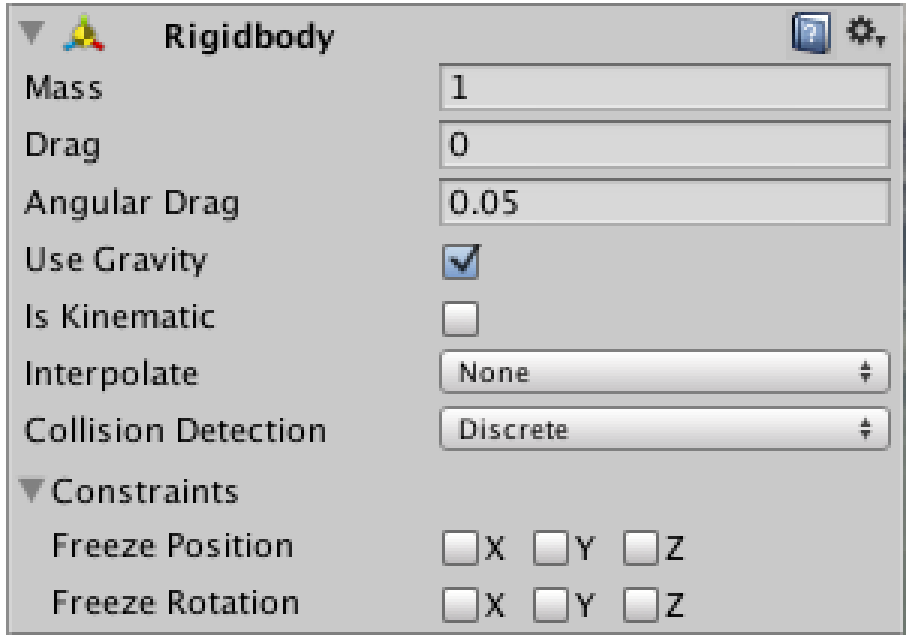
- Invisible shape that defines the physical shape for collisions
- Different shapes: box, capsule, sphere, mesh, etc.
- Default: collider is stationary (never moves)



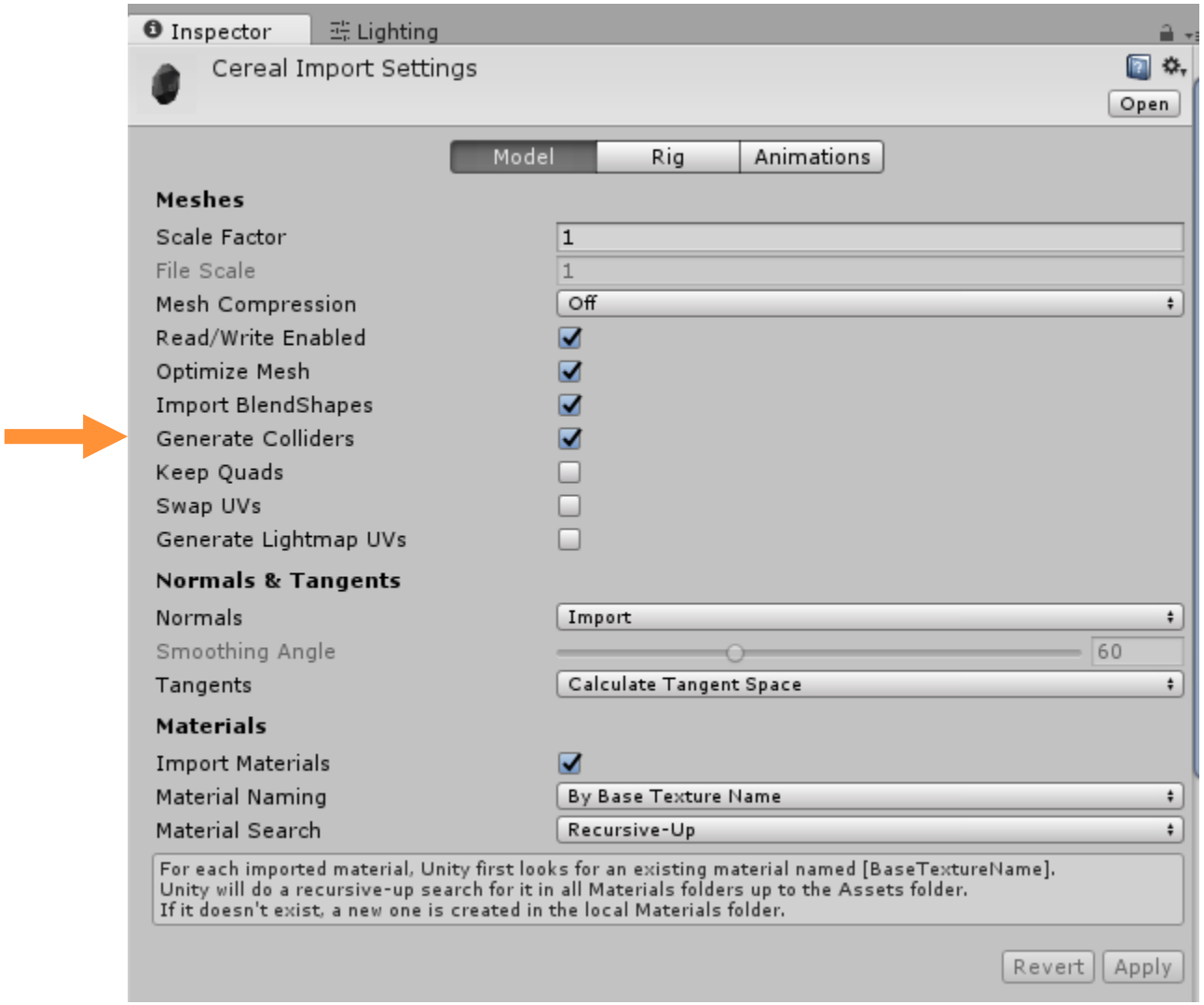


Rigidbody

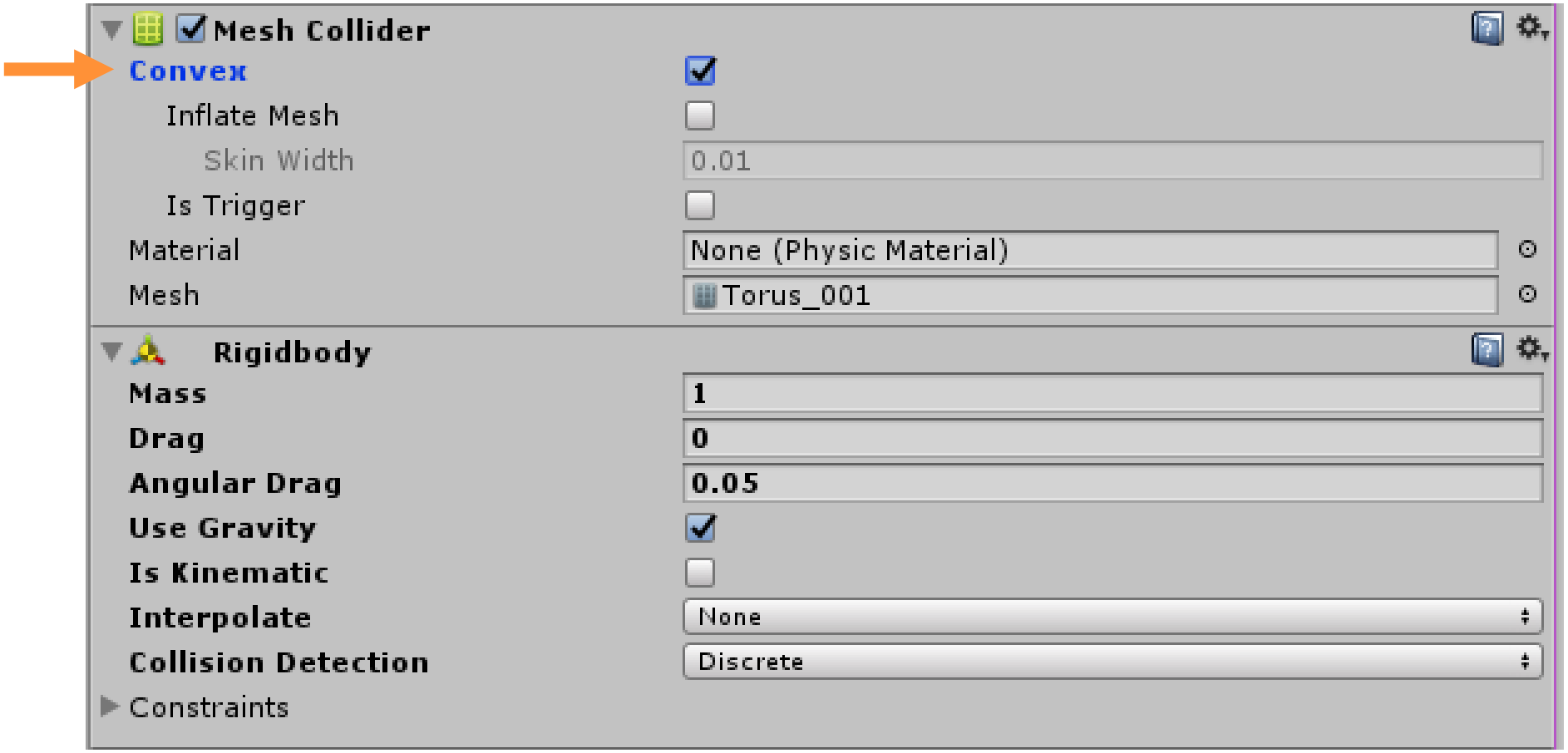
- Physics simulation component
- Adds mass, drag, gravity, etc.
- Requires a collider



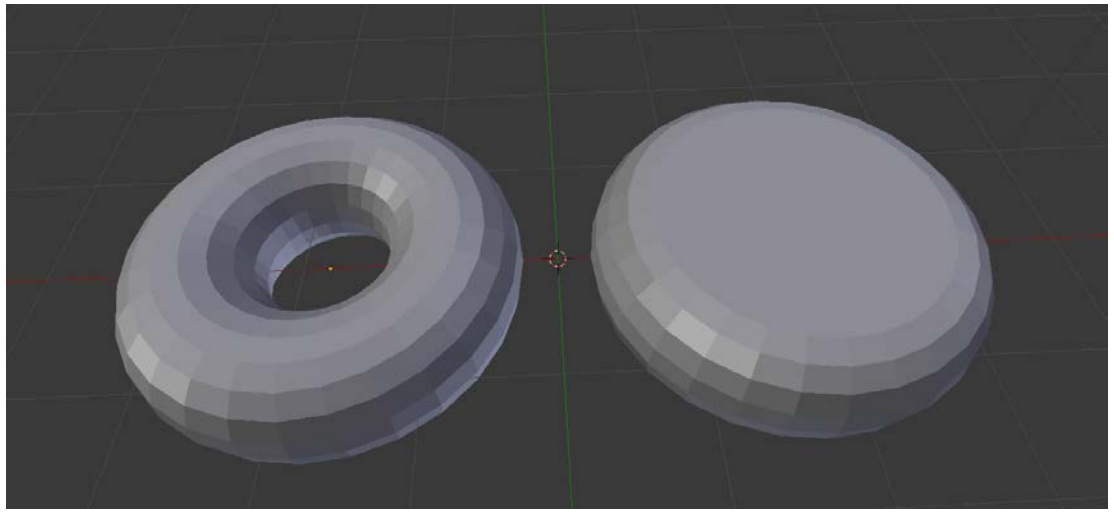
Importing 3D Model for Physics

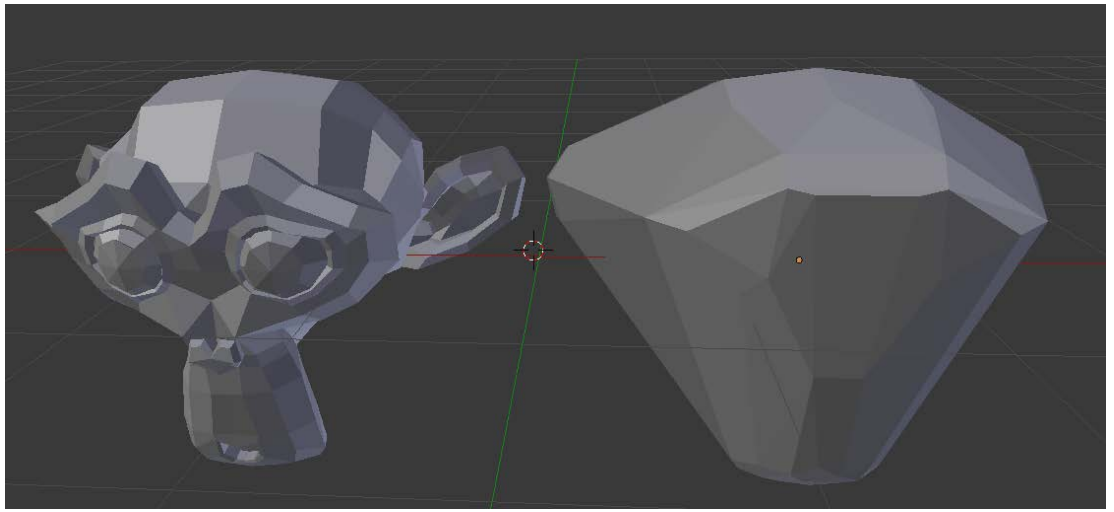


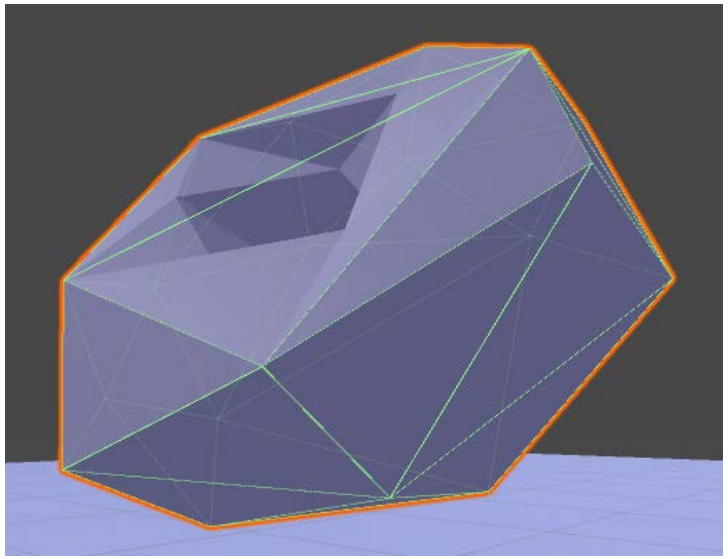
Importing 3D Model for Physics



Convex Hull

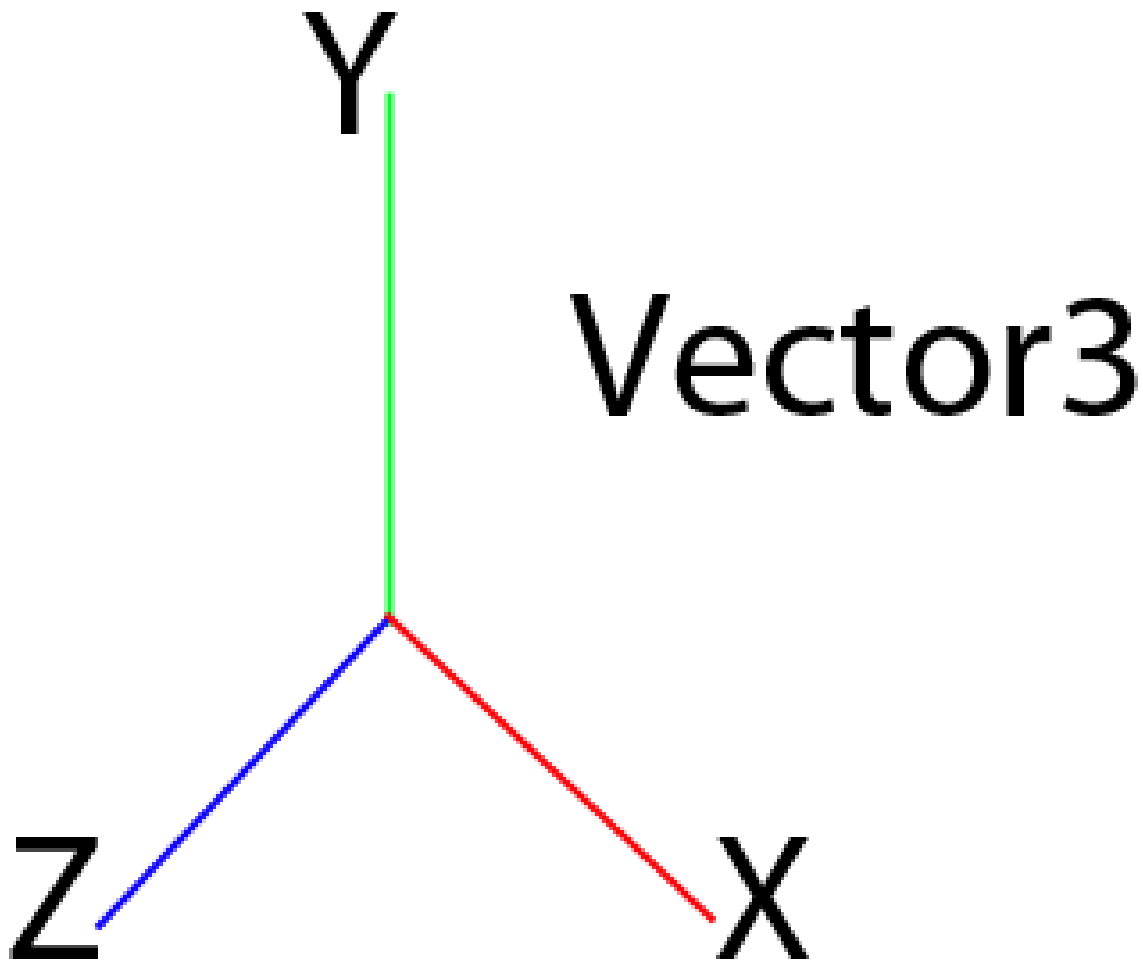






Vector3

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


Y
Vector3
Z
X

Vector3

struct in UnityEngine

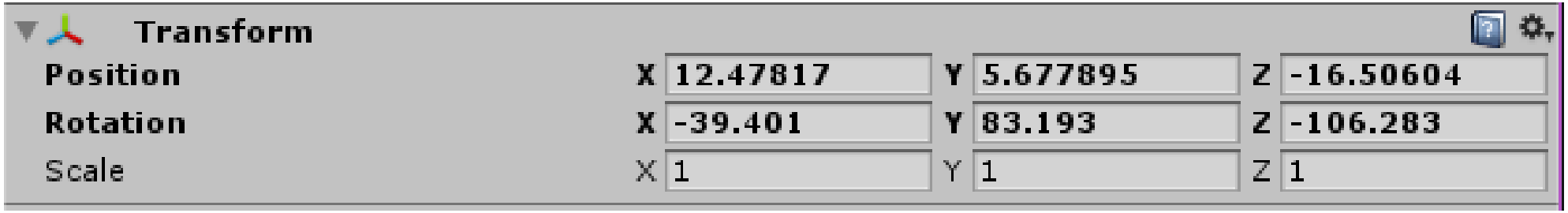
Description

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);
```

Transform



Transform.localPosition

SWITCH TO MANUAL

public Vector3 **localPosition**;

Transform.position

SWITCH TO MANUAL

public Vector3 **position**;

Transform.localScale

SWITCH TO MANUAL

public Vector3 **localScale**;

Transform.eulerAngles

SWITCH TO MANUAL

public Vector3 **eulerAngles**;

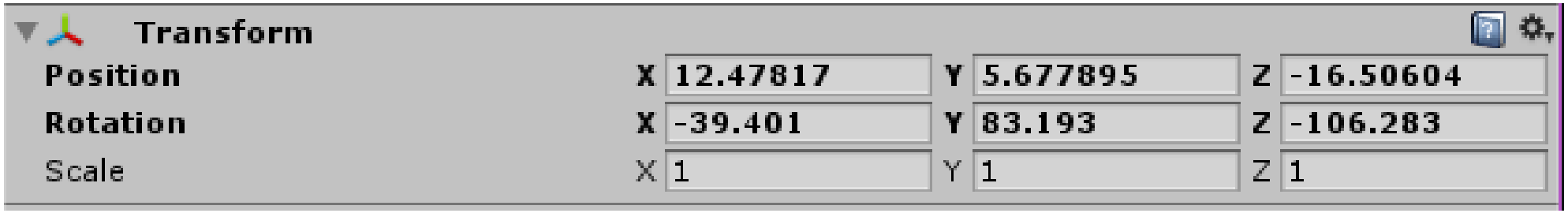
Transform.localEulerAngles

SWITCH TO MANUAL

public Vector3 **localEulerAngles**;

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

Transform



- Access “Transform” component via: “transform”
- Ref: docs.unity3d.com/ScriptReference/Transform.html

```
Vector3 myVector = new Vector3(1, -2, 10);
Vector3 zeroVector = Vector3.zero; // (0, 0, 0)
Vector3 oneVector = Vector3.one; // (1, 1, 1)
Vector3 bigVector = Vector3.one * 10; // (10, 10, 10)
Vector3 sumVector = bigVector + myVector; // (11, 8, 20)
```

back	Shorthand for writing Vector3(0, 0, -1).
down	Shorthand for writing Vector3(0, -1, 0).
forward	Shorthand for writing Vector3(0, 0, 1).
left	Shorthand for writing Vector3(-1, 0, 0).
one	Shorthand for writing Vector3(1, 1, 1).
right	Shorthand for writing Vector3(1, 0, 0).
up	Shorthand for writing Vector3(0, 1, 0).
zero	Shorthand for writing Vector3(0, 0, 0).

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

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/

Distance

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

Accessing Components

Components On The Same Object

```
public class LightColorSwitcher : MonoBehaviour {

    private Light LightComponent;

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

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

    }
}
```

Generic Method

```
LightComponent = GetComponent<Light>();
```

TYPE OF
COMPONENT

Components On Other Objects

(Inspector Method)

```
public class Script04_Distance : MonoBehaviour {

    public Transform PlayerTransform;

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

Components On Other Objects

(Scripting Method)

```
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

```
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!");
        }
    }
}
```

Color

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

Color

struct in UnityEngine

Description

Representation of RGBA colors.

This structure is used throughout Unity to pass colors around. Each color component is a floating point value with a range from 0 to 1.

Components (r,g,b) define a color in RGB color space. Alpha component (a) defines transparency - alpha of one is completely opaque, alpha of zero is completely transparent.

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

Color Constructor

public **Color**(float **r**, float **g**, float **b**, float **a**);

Parameters

r	Red component.
g	Green component.
b	Blue component.
a	Alpha component.

Description

Constructs a new Color with given r,g,b,a components.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public Color color = new Color(0.2F, 0.3F, 0.4F, 0.5F);
}
```

https://docs.unity3d.com/ScriptReference/Color-ctor.html

Color.Lerp

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

Parameters

a	Color a
b	Color b
t	Float for combining a and b

Description

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

`t` is clamped between 0 and 1. When `t` is 0 returns `a`. When `t` is 1 returns `b`.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public Color lerpedColor = Color.white;
    void Update() {
        lerpedColor = Color.Lerp(Color.white, Color.black, Mathf.PingPong(Time.time, 1));
    }
}
```

http://docs.unity3d.com/ScriptReference/Color.Lerp.html