Visual raycast Download

Raycast utilities and visualizer for Unity. Not required any additional components or interfaces on your objects. No conflicts with the standard unity raycast.

Author: Igor-Valerii Chebotar
Email: igor.valerii.chebotar@gmail.com

Requirements

- Simple Man - AsyncOperations
- Simple Man - Utilities

How to install plugin?

Open installer by the click on Tools -> Simple Man -> Main Installer -> [Plugins' name] -> Click 'Install' button. If you don't have one or more of the plugins this plugin depends on, you must install it first.

Quick start

1. Add 'VisualRaycastDrawer' game object on your scene by right click inside the 'Hierarchy' window and select 'Visual Raycast Drawer' option.
2. Add the namespace `SimpleMan.VisualRaycast` .
3. Copy the code below into `Update` method in your class.

```
this.
Raycast().
FromGameObjectInWorld(gameObject).
ToDirection(transform.forward).
ContinueWithDefaultParams();
```

4. Your C# class must be similar to:

```
using SimpleMan.AsyncOperations;
using SimpleMan.VisualRaycast;
using UnityEngine;

public class RaycastExample : MonoBehaviour
{
```

```
    private void Update()
    {
            //The entry point - your class. You can aslo use 'VisualRaycastAPI' instead of 'this'
            this.

            //The type of the operation
            Raycast().

            //The origin of the raycast - this game object
            FromGameObjectInWorld(gameObject).

            //The direction of the raycast - forward direction of this game object
            ToDirection(transform.forward).

            //The parameters of the operation (could be extended)
            ContinueWithDefaultParams();
    }
}
```

5. Done! All examples you also can find in the 'Demo' folder.

Raycasts from camera

The next code demonstrates how you can make raycast from the main camera.

```
//Declarate raycast operation and request to get result of it
var result = this.

//Type of operation
Raycast().

//Origin of ray is main camera
FromMainCamera().

//Direction is mouse position in world
ToMousePositionInWorld().

//The parameters of the operation (could be extended)
ContinueWithDefaultParams();
```

How to ger result of operation?

Each cast operation returns `PhysicsCastResult`. Example:

```
//Getting result
var result = this.
    Raycast().
    FromGameObjectInWorld(gameObject).
    ToDirection(transform.forward).
    ContinueWithDefaultParams();

//Hit check: returns 'true' if at least one hit was detected
if(result)
{
    //Get all hits:
    foreach(var hit in result.hits)
        Debug.Log("Detected object: " + hit.collider.name);
}
```

Using custom parameters

This was a simple exable above. Here is an extended:

```
//The entry point - your class. You can aslo use 'VisualRaycastAPI' instead of 'this'
this.

//The type of the operation
Raycast().

//The origin of the raycast
FromGameObjectInWorld(gameObject).

//The direction of the raycast
ToDirection(transform.forward).

//Single is analogue of simple 'Physics.Raycast' and Multi is analogue of 'Physics.RaycastAll'.
SingleHit().

//Ignore all layers except 9
UseCustomLayerMask(1 << 9).

//Ignore this game object
IgnoreObjects(gameObject);
```

Spherecast and boxcast

You can use this operations the same way as 'Raycast'. Example:

```
//Spherecast
this.
    Spherecast().
    FromGameObjectInWorld(gameObject).
    ToDirection(transform.forward).
    SingleHit().
    WithRadius(_radius).
    WithDistance(_distance).
    ContinueWithDefaultParams();

//Boxcast
this.
    Boxcast().
    FromGameObjectInWorld(gameObject).
    ToDirection(transform.forward).
    SingleHit().
    WithSize(_size).
    WithRotationOf(gameObject).
    WithDistance(_distance).
    ContinueWithDefaultParams();
```

Sphere and box overlap

Plugin supports this operations. You can call them by similar way as previous. Example:

```
//Sphere overlap
this.
    SphereOverlap().
    FromGameObjectInWorld(gameObject).
    WithRadius(_radius).
    ContinueWithDefaultParams();

//Box overlap
this.
    BoxOverlap().
    FromGameObjectInWorld(gameObject).
    WithSize(_size).
    WithRotationOf(gameObject).
    ContinueWithDefaultParams();
```

Can I use layer masks and other standard raycast arguments?

Sure! Visual raycast has all arguments of classic raycast. Also it has `IgnoreGameObjects` parameter. Use it if you need to ignore concrete game objects.

Extension functions for the 'MonoBehaviour'

You can use this functions from each of your MonoBehavior classes. **Don't forget to write '*this.*' keyword to use exension functions.**

Methods

Function name	Description
Raycast	Make ray cast
Boxcast	Make box cast (sphere on end of line)
Spherecast	Make sphere cast (sphere on end of line)
BoxOverlap	Check the area by box
SphereOverlap	Check the area by sphere

You can also use `VisualRaycastAPI` to use the same functions in non-monobehaviour classes. They could be visualized too, of course.

Visual raycast drawer component

This is mono class that draws the gizmos. You can delete it from scene, if you don't need visualization of physics processes. right click inside the 'Hierarchy' window and select 'Visual Raycast Drawer' option to add this object again.

Config and global settings

You can change color of the gizmos, hit poins size and other settings in project settings tab, that can be found by path: Project settings -> Visual raycast