Debug Toolkit for Unity (UDT)

> *Thanks for using the UDT. Unity 6+ is supported for this asset.*

This documentation aims to get you ready to use and modify UDT for your awesome projects.

The current version of the toolkit 1.0.

If you are looking for the last patch note please check the Patch Notes. If you are looking for features in development take a look at the What's Next section.

Features

UDT is composed of four elements :

- A modular console, for which can create custom commands but also log anything you want at runtime. Maybe you wish to have a simple way to kill your player, teleport to some place, show all the colliders that are in the scene or maybe you want to go frame by frame to know what is happening. Now its a simple command line.
- A system of optimized runtime Gizmos with an easy to use API. It makes your debugging way easier. Control them with the console, or as a standalone feature.
- A FreeCam to inspect your scene at runtime. With a simple command in the console activate or deactivate the FreeCam and navigate through your scene.
- A metrics system to now your fps, the number of batches, and your tris/vert usage.

Introduction to UDT

This toolkit aims to make your life easier while debugging your game.
In this version we focused on the run time debugging

We are aiming for a load of improvements in the future versions of this toolkit. If you want to now more about the next features please check the What's Next section.

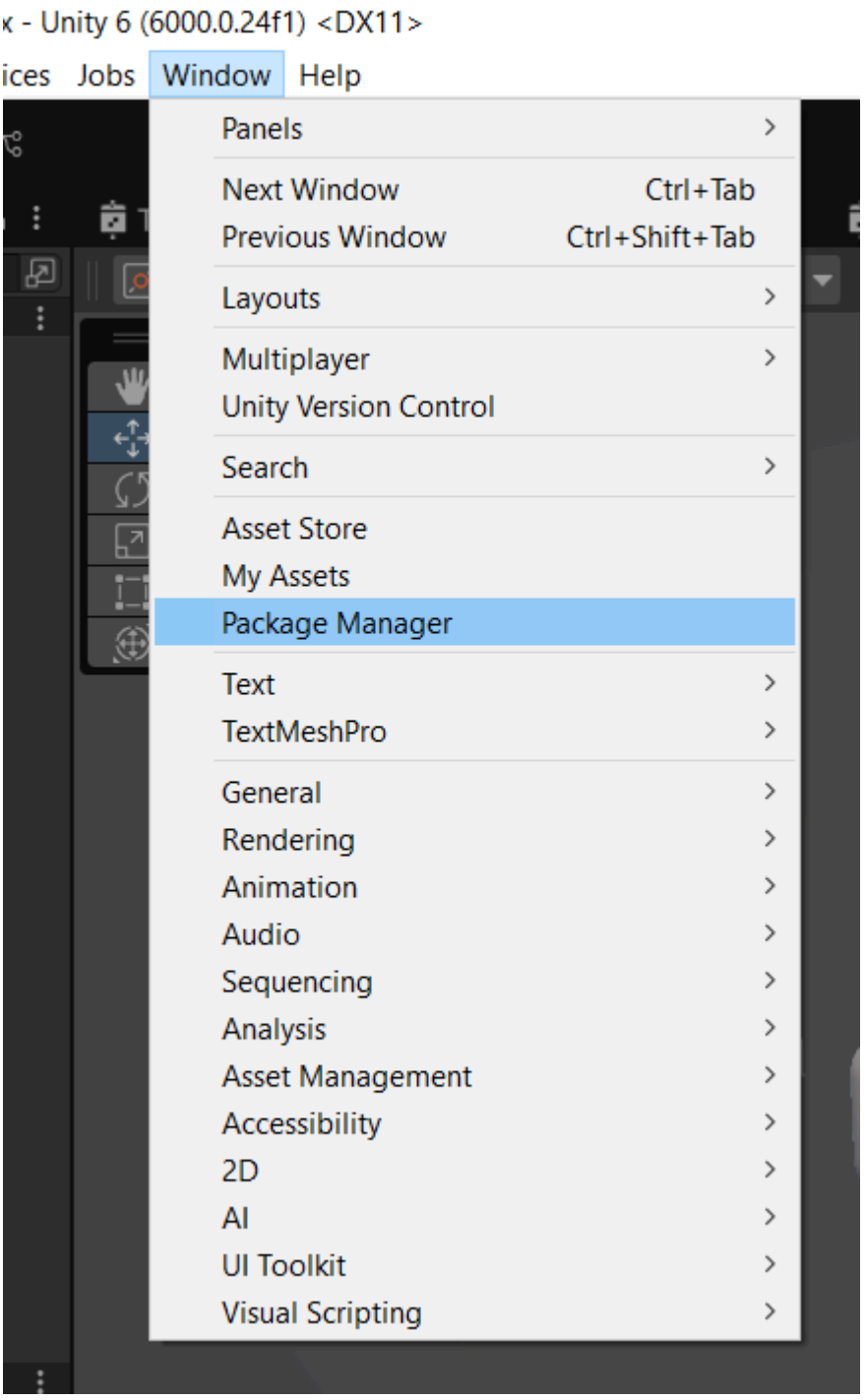
In the next few sections you are going to learn how to import UDT to your project and how to use it.

Install UDT

> *Make sure that you are using Unity 6 or above.*

Package manager

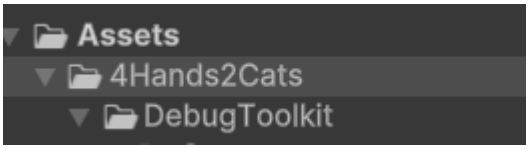
Get into the unity package manager at the top of the Unity Engine window and look for UDT in your assets



Click on the **download** button at the bottom right of the menu.

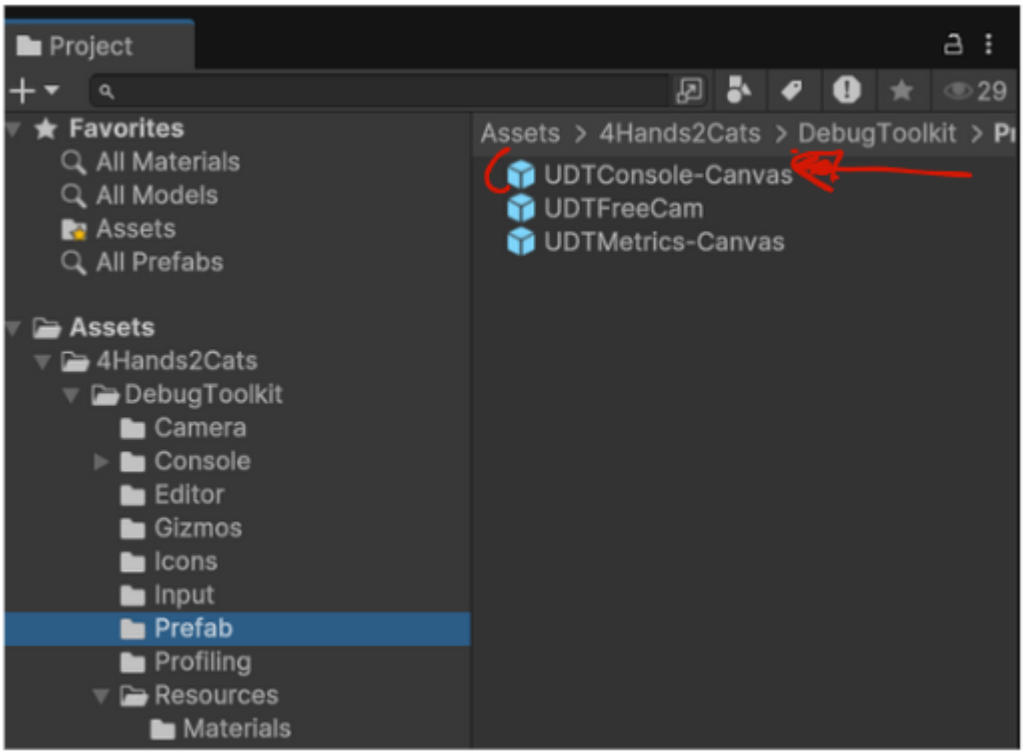
Once the files are downloaded, import them in your project by clicking on the **Import** button.

The toolkit is in the 4Hands2Cats folder. In the future if we do more assets they'll install in this folder as well.



Quick start guide

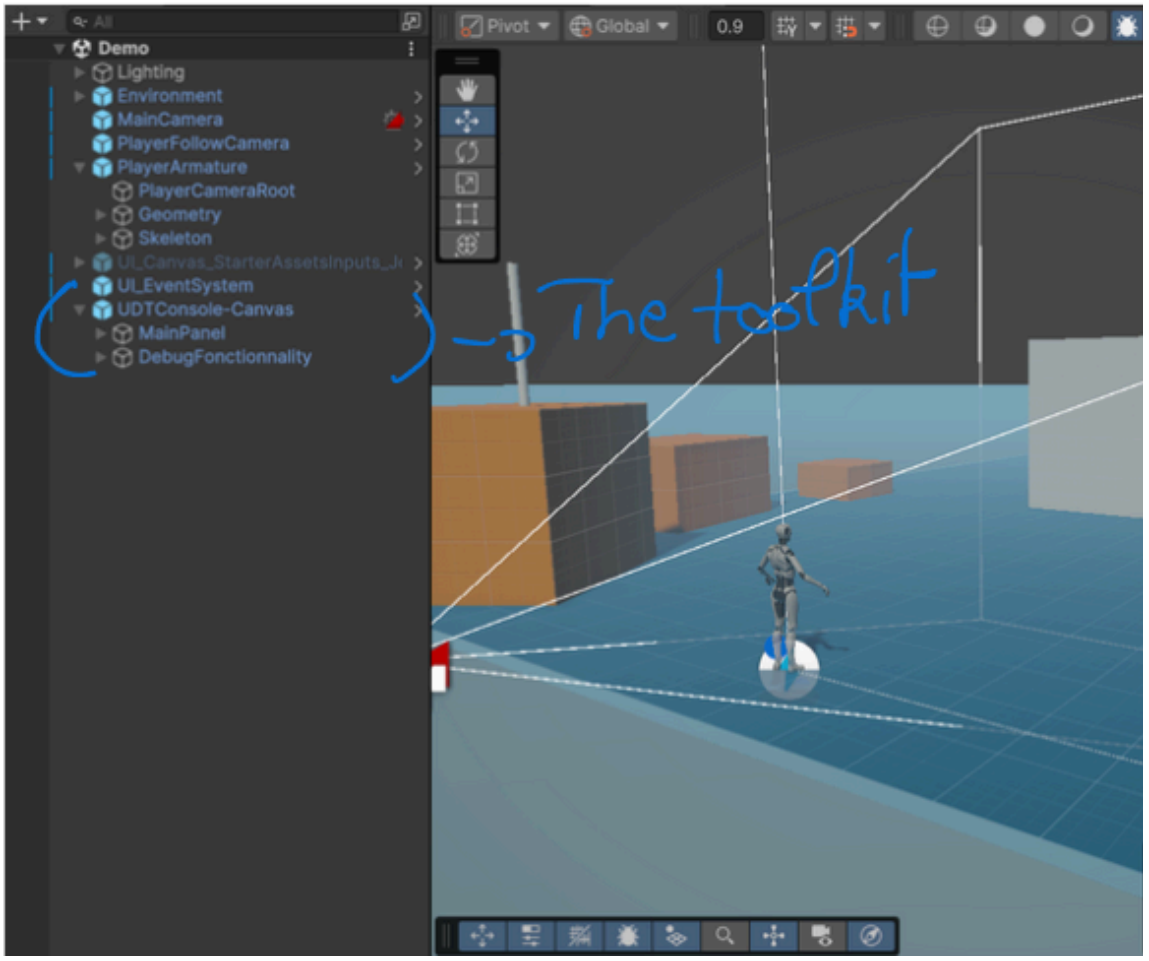
As you just seen in the demo scene, the toolkit is pretty simple to include in your project. You just have to drag and drop the **UDTConsole-Canvas** prefab from the prefab folder inside your scene.



Don't forget to remove it for production builds.

Demo Scene Tour

The demo scene is based on the third person template of Unity. We've added our debug toolkit in this scene.



To use this demo scene just press play. The Console and the all the other features are on the **F12** key.

To navigate in the scene use the **WSDQ** keys and the mouse to control the camera.

Console

The In-game console is the central piece of UDT. It controls every features of the toolkit and more.

> *This section is the documentation of the usage of the console as is. In the customization section you'll find explanations on how to make your own commands for the console.*

Refer to the Quick Start section to learn how to enable the in game console for your project.

Built in commands

There is a set of built in commands that you can use to control the toolkit and enable/disable features.

Like in any console you can use the top and bottom arrow of your keyboard to navigate through the command you've already written.

Simple

- Type *help/Help/-h/-h* in the console to get a list of all the available commands.

Booleans

- Type *metrics/Metrics/-m/-M* followed by *enable/e* or *disable/d* to enable or disable the metrics.
- Type *freecam/Freecam* followed by *enable* or *disable* to enable or disable the freecam.
- Type *light/-l* followed by *enable/e* or *disable/d* to enable or disable all the lights in the scene.
- Type *shadows/-s* followed by *enable/e* or *disable/d* to enable or disable all the shadows in the scene.
- Type *Collider/collider/-c/-C* followed by *enable/e* or *disable/d* to enable or disable the in game gizmos rendering for the collider (beware on large scene there might be a small freeze when enabling). This command also activates all the other in game gizmos [going to change in next version]
-
- Type *Gizmos/gizmos/-g/-G* followed by *enable/e* or *disable/d* to enable or disable the in game gizmos.

Vector

- Type *Time/time/-t/-T* followed by a float value between 0 and 100 to change the time scale of your game.
- Type *Frame/frame/-f/-F* followed by a float value x to navigate x frames in the future. [Note : If the number of x is too high you might wait a long time].

Enum

- Type *graphics/Graphics/-g/-G* followed by *low/l* or *medium/m* or *high/h* or *ultra/u* to change the quality settings. [Note : This scriptable object of this command will have to be changed if your quality settings are not following the standard ones of an URP project (*verylow/low/medium/veryhigh/ultra*)].

Debugs logs

Free Cam

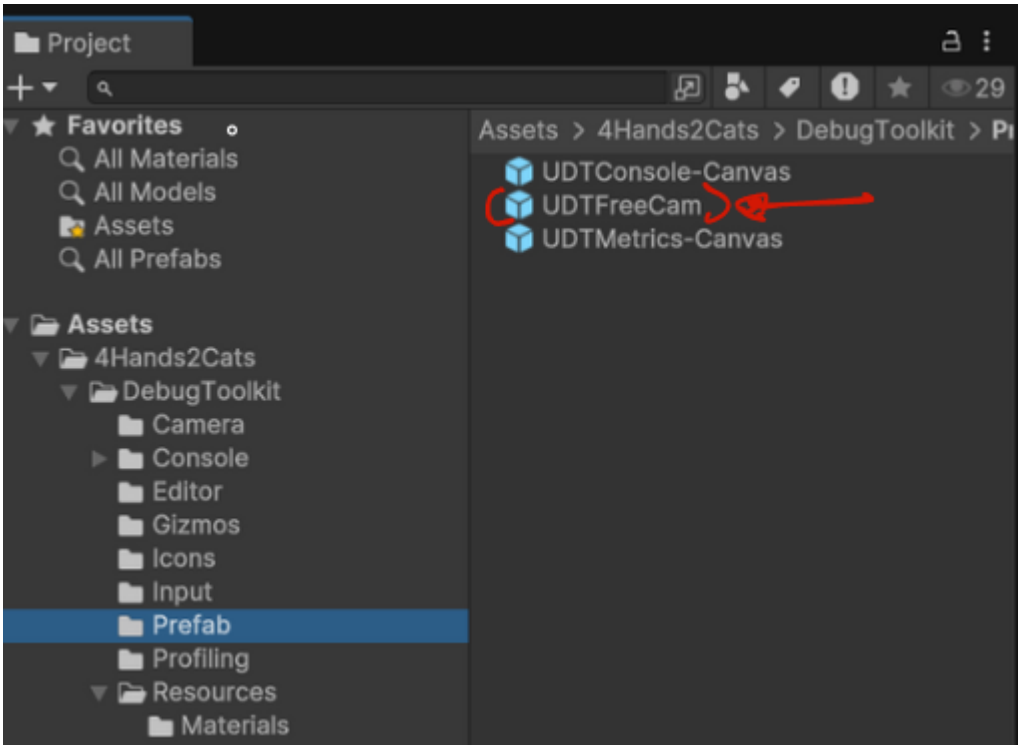
The free cam is here to give you a tool that's similar to the navigation in the scene panel of unity, but at runtime.

To use type *freecam/Freecam* followed by *enable* or *disable* to enable or disable the freecam in the console after opening it using **F12**.

Though this embeded in the package, the freecam comes as stand alone feature. Feel free to use it for your gameplay if you want.

> *Future versions of the cam should be compatible with cinemachine.*

The prefab for this one is in the same folder as the console.



To use as a stand alone feature just drag and drop the prefab in your scene.

> *Note : If you use it as stand alone feature it'll not be control anymore by the console.*

Gizmos

As you now there is already an API in unity to draw gizmos for debugging purposes. But you cannot draw gizmos for run time. Those gizmos are harvesting the power of the **GL** API to show performance friendly gizmos at runtime for quite anything.

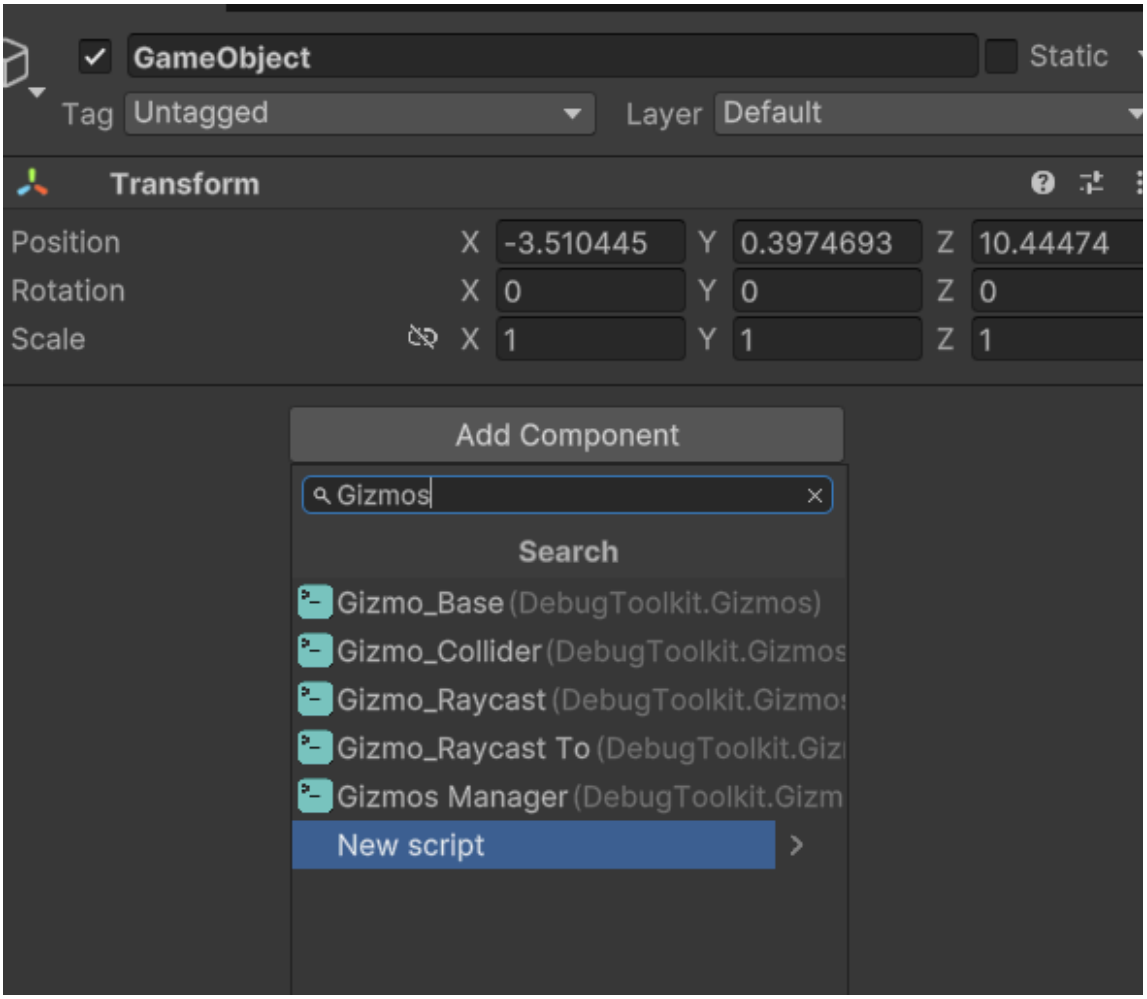
There are two way of using the gizmos. Manual and automatic using the console.

- The manual way consist of adding the gizmos components to your gameobjects
- The automatic way is used to show all colliders in the scene. To show all colliders simple type the command : *Collider/collider/-c/-C* followed by *enable/e* or *disable/d* to enable or disable the in game gizmos rendering for the collider (beware on large scene there might be a small freeze when enabling). This command also activates all the other in game gizmos [going to change in next version] int the ingame console.

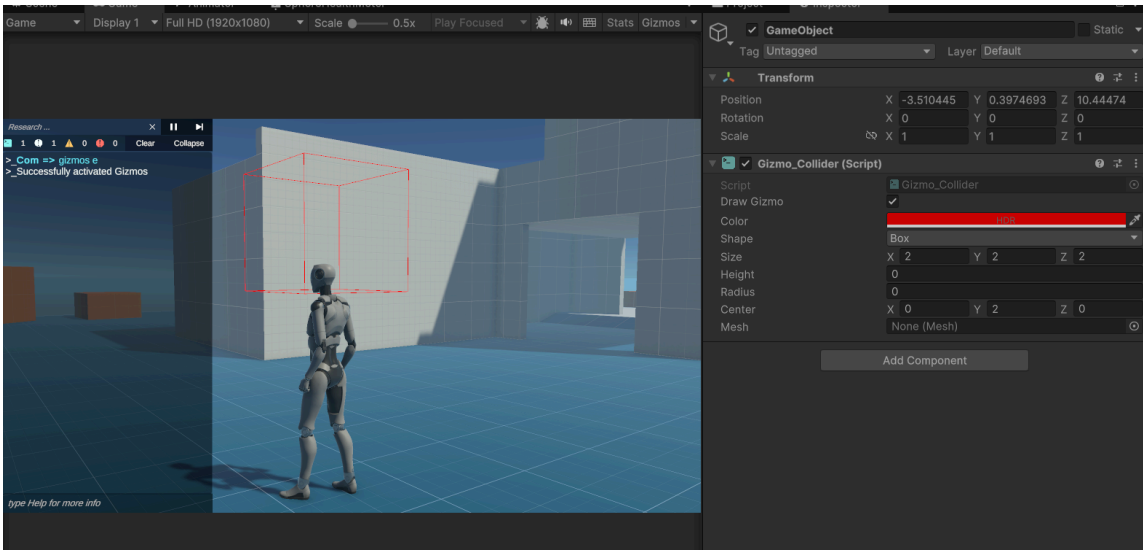
Gizmos that are manually setted up can are still managed by the ingame console. Simple type : *Gizmos/gizmos/-g/-G* followed by *enable/e* or *disable/d* to enable or disable the in game gizmos.

> *Note : this commands would also hide the collider gizmos. We are awaiting for you feedback to improve this feature.*

To set up a Gizmos as component simple add the choosen Gizmos to your gameobject.



Use the gizmos Collider to choose show a collider like shape.

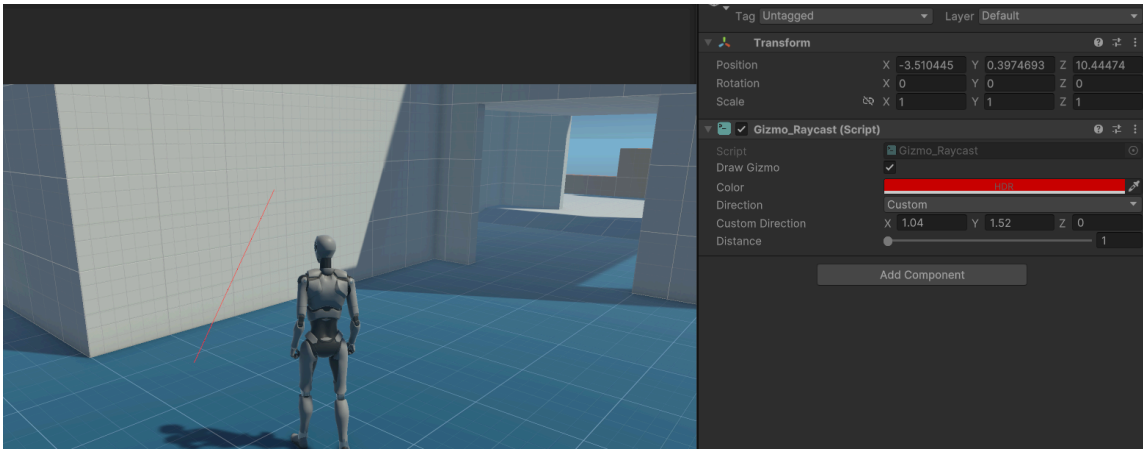


There are four options :

- The **Box**, that you can control using the **Size** and the **Center** parameters
- The **Sphere**, that you can control using the **Radius** and the **Center** parameters
- The **Capsule**, that you can use using the **Radius**, the **Center** and the **Height** parameters
- The **Mesh**, that you can use by drag and dropping a mesh in the mash container of the gameObject. [To enable this feature you need to enable Read/Write on your meshes in the import settings].

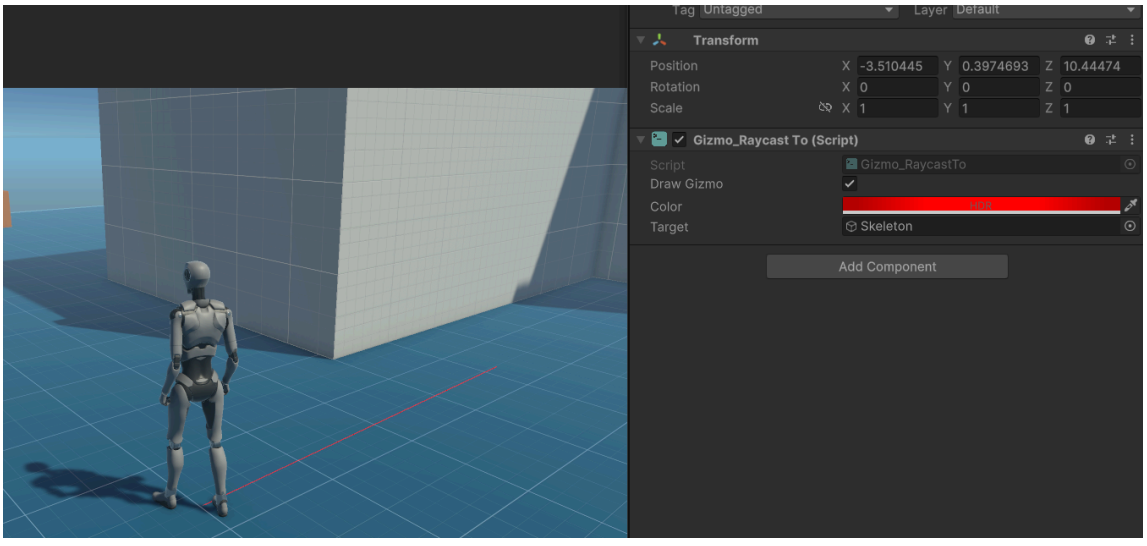
> *Note that those gizmos support bloom and they can be used as part of your project for other purposes than debugging.*

Use the Gizmos Raycast component to draw a Ray cast.



There are some direction preset for all the cartesian direction (up, left, etc...) but you can use custom to choose your own direction.

Use the Gizmos Raycast To component to draw a gizmos between two targets.



You just have to give it a ref to a gameobject and it'll draw a ray to it and update the ray at everyframes.

We are planning on adding custom editors in the future to simplify the usage of those features. Please take a look at the section Gizmos API to learn about the code API for the gizmos.

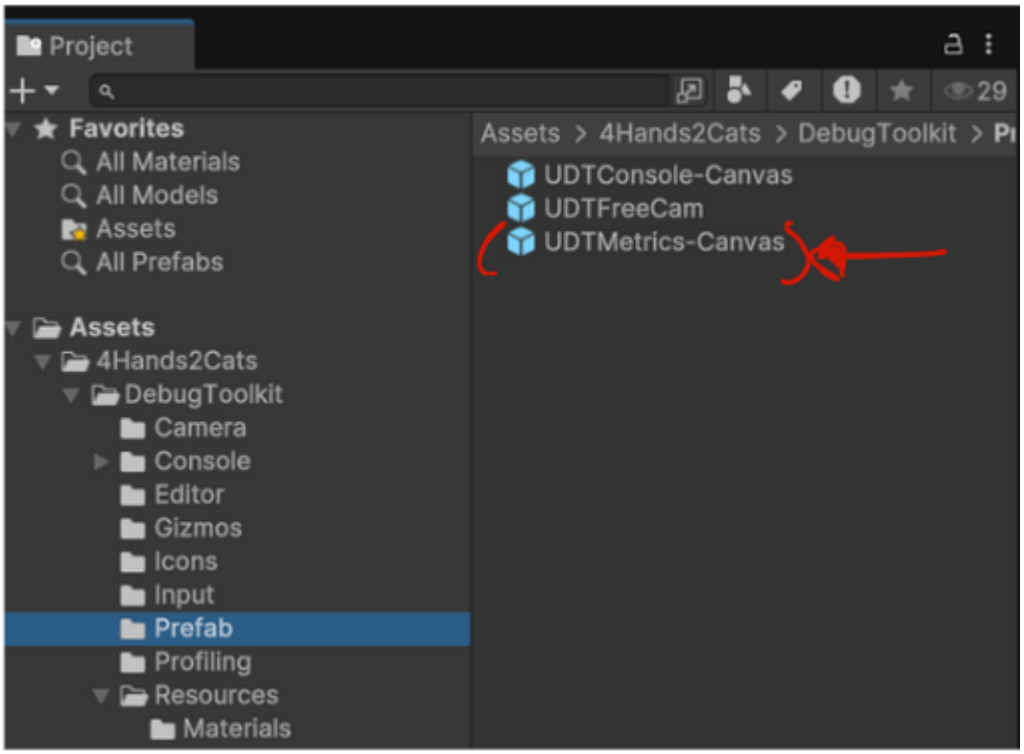
Metrics

The metrics are enabled using the interactive console with the command :

> *metrics/Metrics/-m/-M followed by enable/e or disable/d to enable or disable the metrics.*

For now it show the FPS, the number of batches, the number of tris and the number of vert. In future updates it'll help you to do some profiling.

You can use the metrics as a stand alone feature by drag and dropping it in your scene.



> *Note : If you use the metrics as a standalone feature, it'll note be managed by the console anymore.*

About us

At 4Hands2cats, we create debugging tools to streamline development. As two devs (and two cats), we focus on robust, user-friendly solutions for Unity. Our first asset, a complete debug toolkit, works in build and runtime for full control. We're committed to improving and expanding our tools.

FAQ

What's next ?

Patch Note

This section is the log of all the changes and additions to UDT since release.

V 1.0

UDT Release :

- In game Console
- In game gizmos
- In game FreeCam
- In game basic metrics