I strongly recommend that you use the online documentation:
https://tim-entertainment.gitbook.io/tim.console/

Of course, if you can't, offline documentation has been created for this purpose.

Discord if you need help:
https://discord.gg/PRwCwdPppp

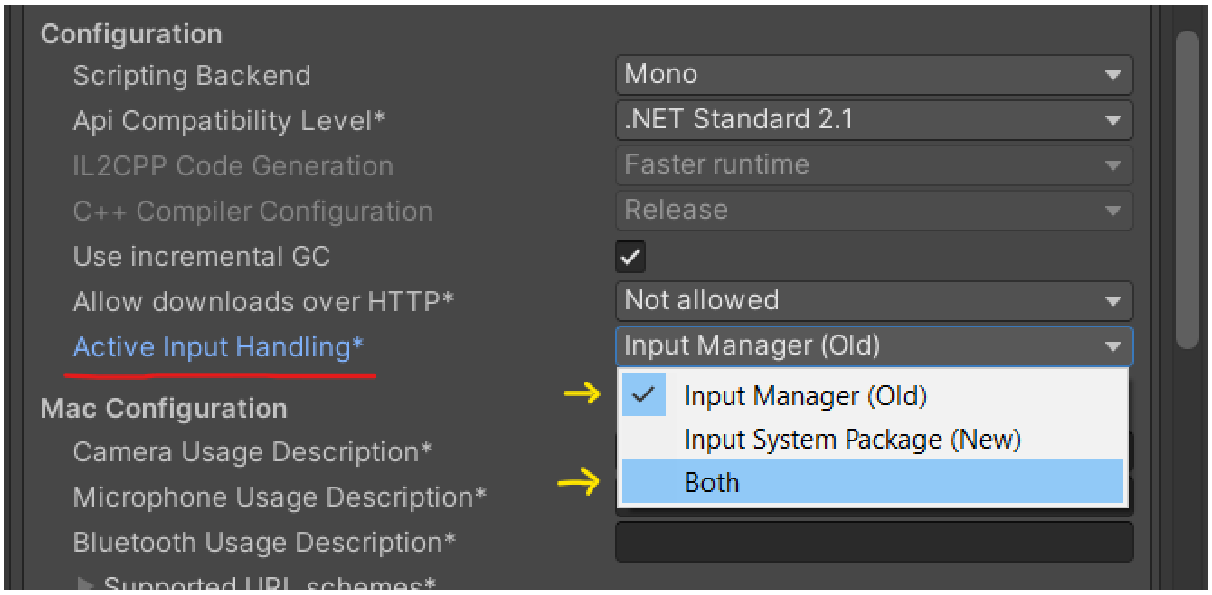
Quick start

Requirements:

❗ Odin inspector asset required

❗ InputManager should be enabled
Use 'Input Manager (Old)' or 'Both'

Check: ProjectSettings/Player/Other Settings/Configuration



Configuration
Scripting Backend: Mono
Api Compatibility Level*: .NET Standard 2.1
IL2CPP Code Generation: Faster runtime
C++ Compiler Configuration: Release
Use incremental GC: ✓
Allow downloads over HTTP*: Not allowed
Active Input Handling*: Input Manager (Old)

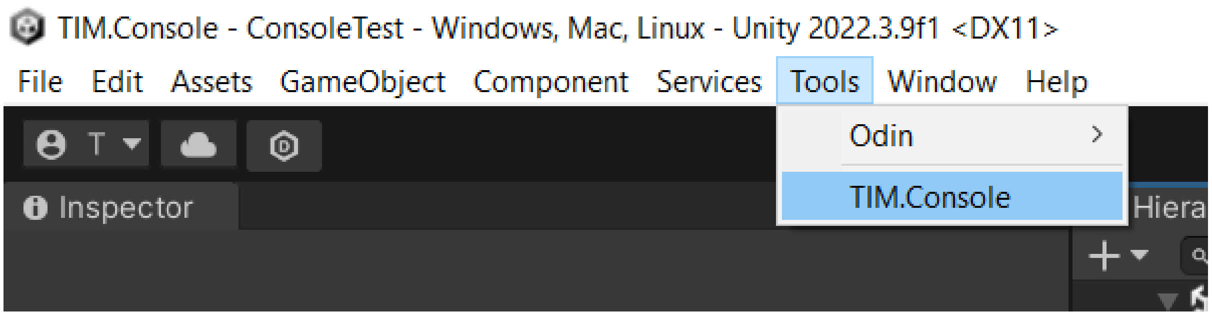
Input Manager (Old)
Input System Package (New)
Both
Mac Configuration
Camera Usage Description*
Microphone Usage Description*
Bluetooth Usage Description*
Supported URL schemes*

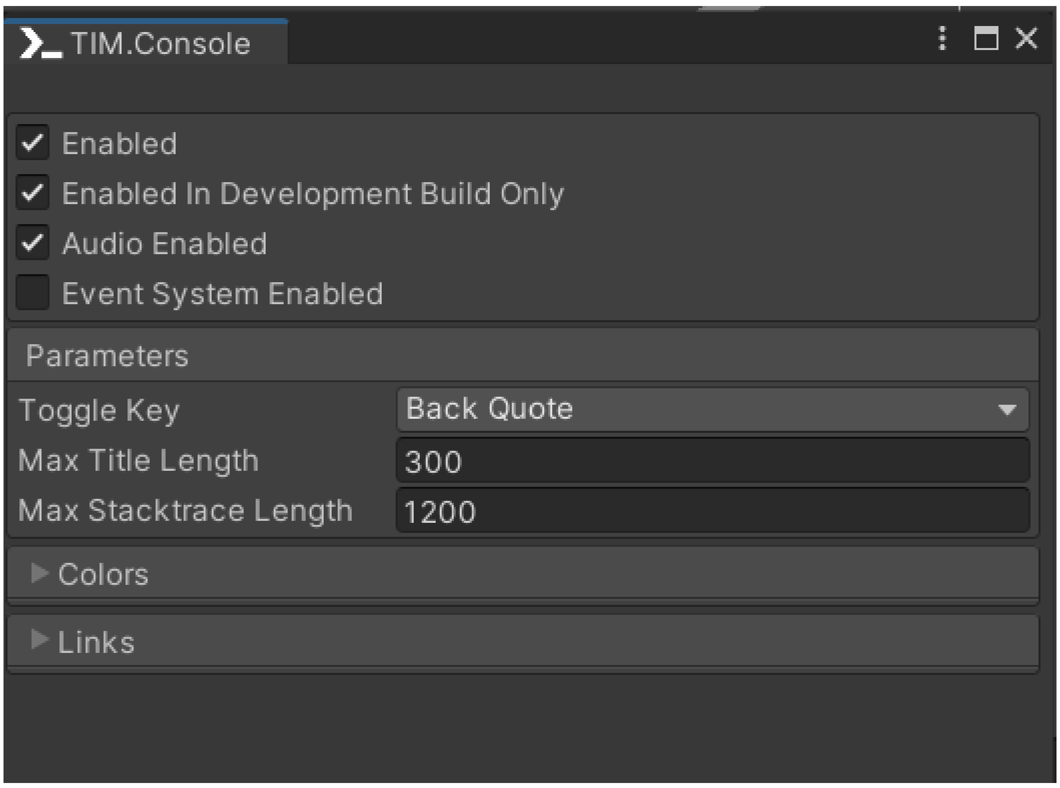
Import and play!

When you imported TIM.Console just enter Play mode and press ` to open the console

> if the Console is unavailable for interaction, most likely you just don't have an EventSystem in the scene. Then you just need to add an EventSystem or enable Console's EventSystem in the Editor window (described below how to do it)

Editor Window


TIM.Console - ConsoleTest - Windows, Mac, Linux - Unity 2022.3.9f1 <DX11>
File Edit Assets GameObject Component Services Tools Window Help
Odin
TIM.Console
Inspector
Hiera



✓ Enabled
✓ Enabled In Development Build Only
✓ Audio Enabled
☐ Event System Enabled

Parameters

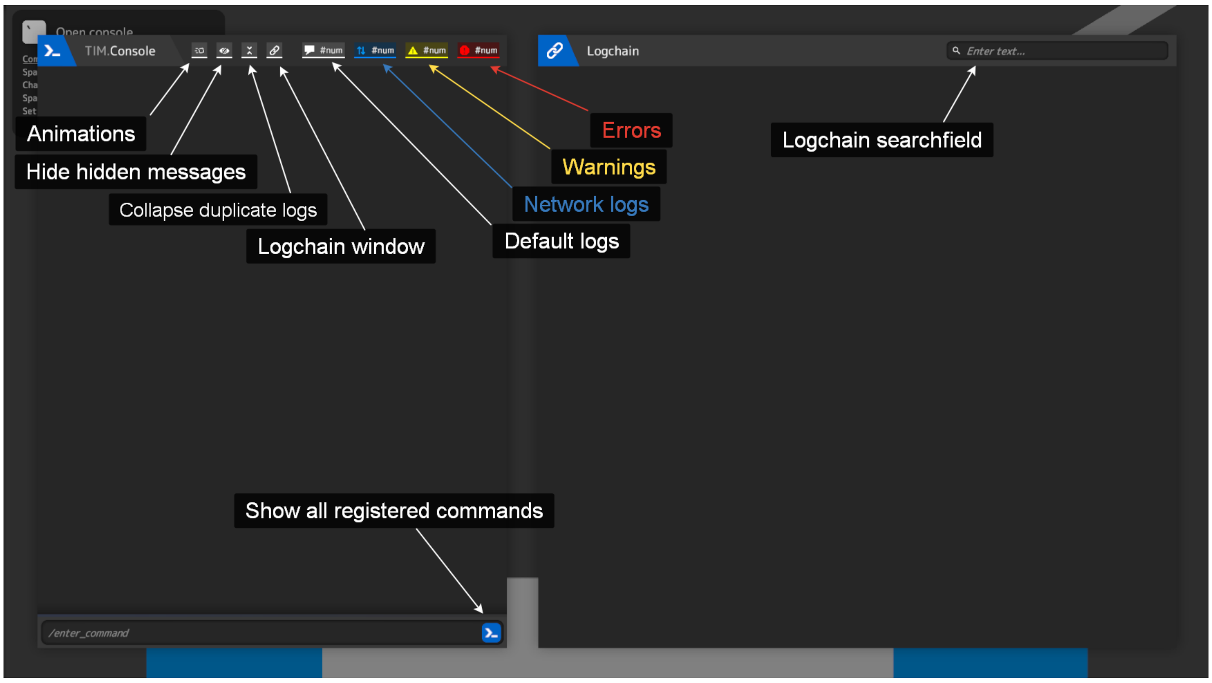
Toggle Key	Back Quote
Max Title Length	300
Max Stacktrace Length	1200

▶ Colors

▶ Links

Enabled - if you disable console will not work
EnabledInDevelopmentBuildOnly - console will work in build only if you check 'Development build' in BuildSettings
AudioEnabled - disable if you don't want to have sounds of Console
EventSystemEnabled - check if you don't have EventSystem on scene
ToggleKey - keyboard button to open the Console
MaxTitleLength - Max length of log title
MaxStacktraceLength - Max length of stacktrace of log to save

User Interface

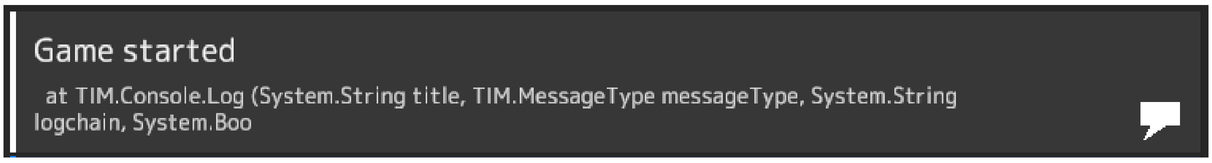


Log

How to do logs in Console

You still can call `Debug.Log()` or `print()`
Also you can use `TIM.Console.Log()`
Here is an example:

```
TIM.Console.Log("Game started");
```

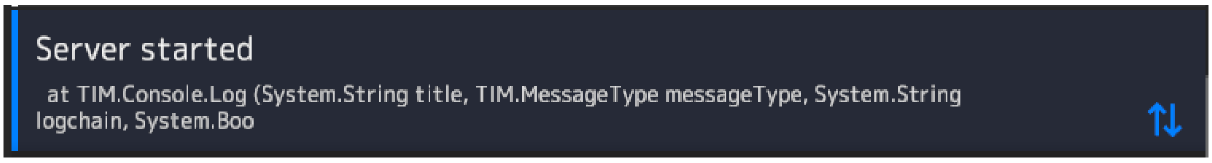


```
Game started
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```

How it looks like in Console

Also you can change the MessageType of your message:
Message types:
💬 Default
🟨 Warning
🟥 Error
🟦 Network

```
TIM.Console.Log("Server started", MessageType.Network);
```



```
Server started
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```

Logchain

Sometimes an in-game event does not happen immediately, it can be divided into several stages. 'Logchain' was created for this purpose

Example:

You are downloading an image from Internet in 3 stages:

1. Web request
2. Downloading
3. Converting downloaded data to Texture2D

You can group all these messages into one Logchain. Just give it name.
For example: "Image downloading". Here is how you can print a message:

```
Console.Log("Your log", MessageType.Default, "Image downloading");
```

So how our method can look like:

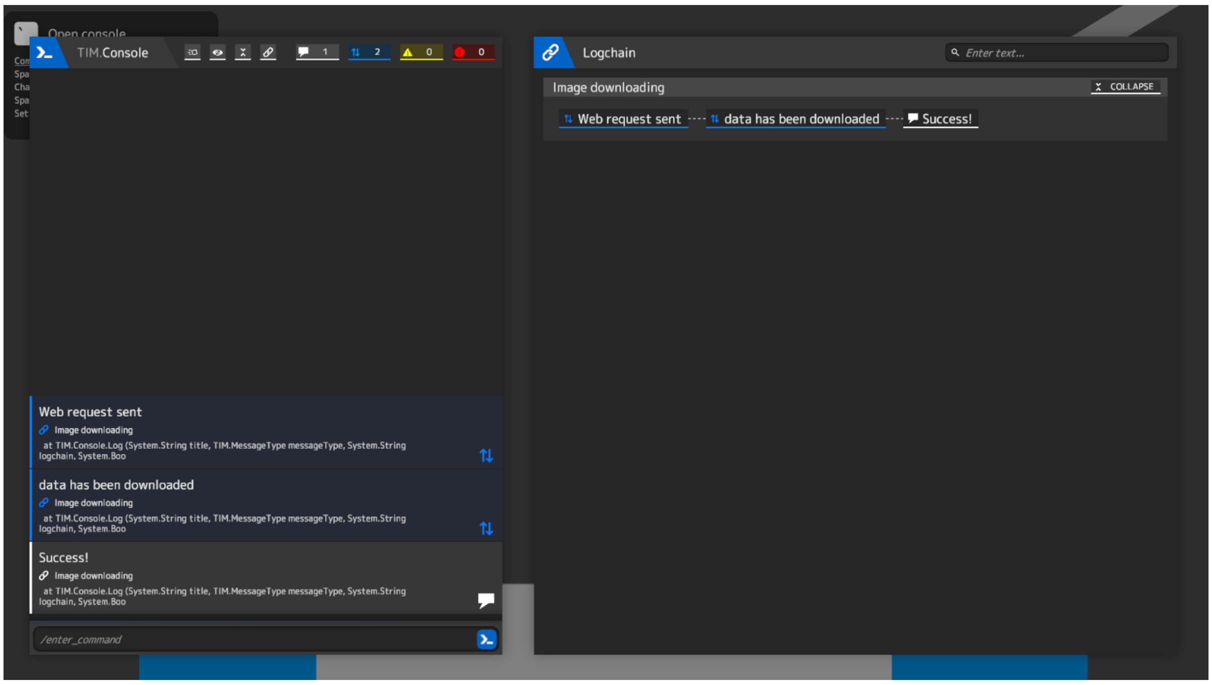
```
IEnumerator DownloadImage(string imageUrl)
    {
        WWW www = new WWW(imageUrl);
        string logchain = "Image downloading";

        Console.Log("Web request sent", MessageType.Network, logchain);
        yield return www;

        if (www.error != null)
        {
            Console.Log(www.error, MessageType.Error, logchain);
        }
        else
        {
            Console.Log("data has been downloaded", MessageType.Network, logchain);
            Texture2D texture = www.texture;

            if (texture == null)
            {
                Console.Log("Data doesn't contant a texture!", MessageType.Error,
logchain);
            }
            else
            {
                Console.Log("Success!", MessageType.Default, logchain);
            }
        }
    }
```

Lets try it!

Execute our coroutine by StartCoroutine() method. And paste image URL as argument.

```
private void Start()
{
    // I found random image from internet and copied the URL:
       StartCoroutine(DownloadImage("https://www.sunhome.ru/i/wallpapers/221/frukti
-oboi.orig.jpg"));
}
```

Open console and you can see that messages are displayed in Console window and in Logchain window as well



```
Web request sent
Image downloading
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo

data has been downloaded
Image downloading
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo

Success!
Image downloading
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```

```
Image downloading
Web request sent ---- data has been downloaded ---- Success!
```

Hidden messages

Hidden messages are hidden in Console (but not hidden in Logchain window)

Lets take our last example with downloading image. If we will download images many times we will have many messages in Console. But we can organize it better! We can hide logs in Console but keep them visible in Logchain window. You just need to set last argument `hidden` of `TIM.Console.Log()` function to true:

```
Console.Log("Web request sent", MessageType.Network, "Logchain
name", true);
```

And how our function looks like now:

```
IEnumerator DownloadImage(string imageUrl)
    {
        WWW www = new WWW(imageUrl);
        string logchain = "Image downloading";

        Console.Log("Web request sent", MessageType.Network, logchain, true);
        yield return www;

        if (www.error != null)
        {
            Console.Log(www.error, MessageType.Error, logchain); // we don't want
to hide errors
        }
        else
        {
            Console.Log("data has been downloaded", MessageType.Network, logchain,
true);
            Texture2D texture = www.texture;

            if (texture == null)
            {
                Console.Log("Data doesn't contant a texture!", MessageType.Error,
logchain); // we don't want to hide errors
            }
            else
            {
                Console.Log("Success!", MessageType.Default, logchain, true);
            }
        }
    }
```

What we will see in Console:

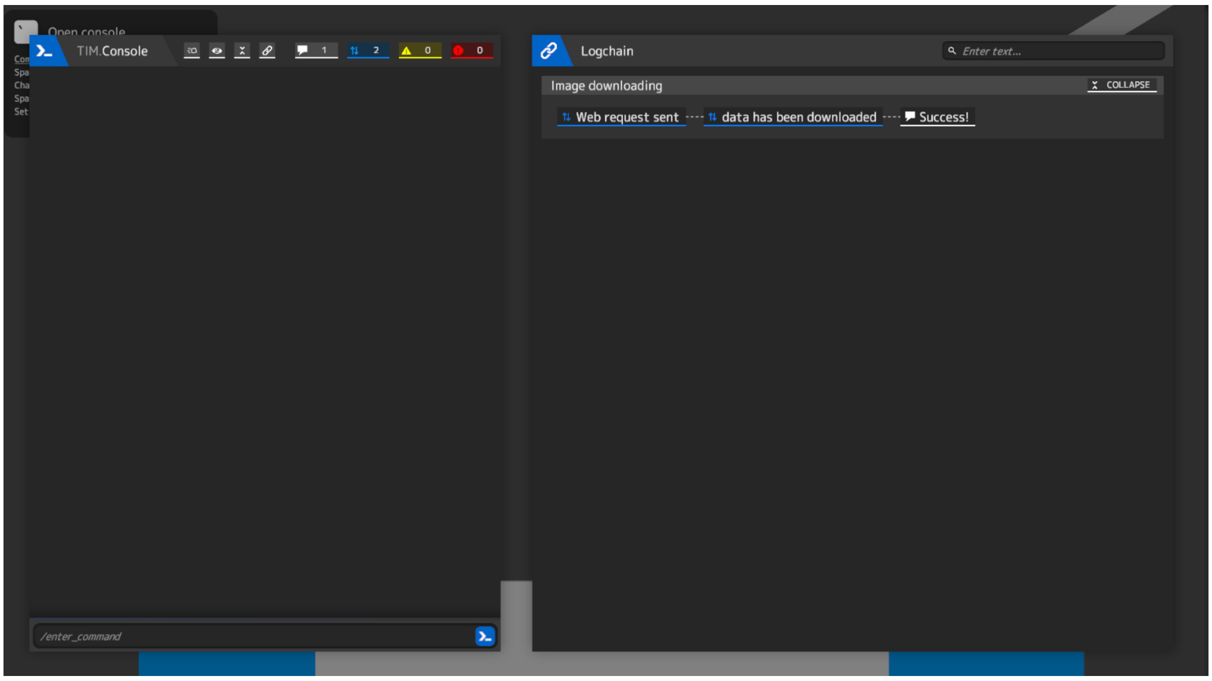
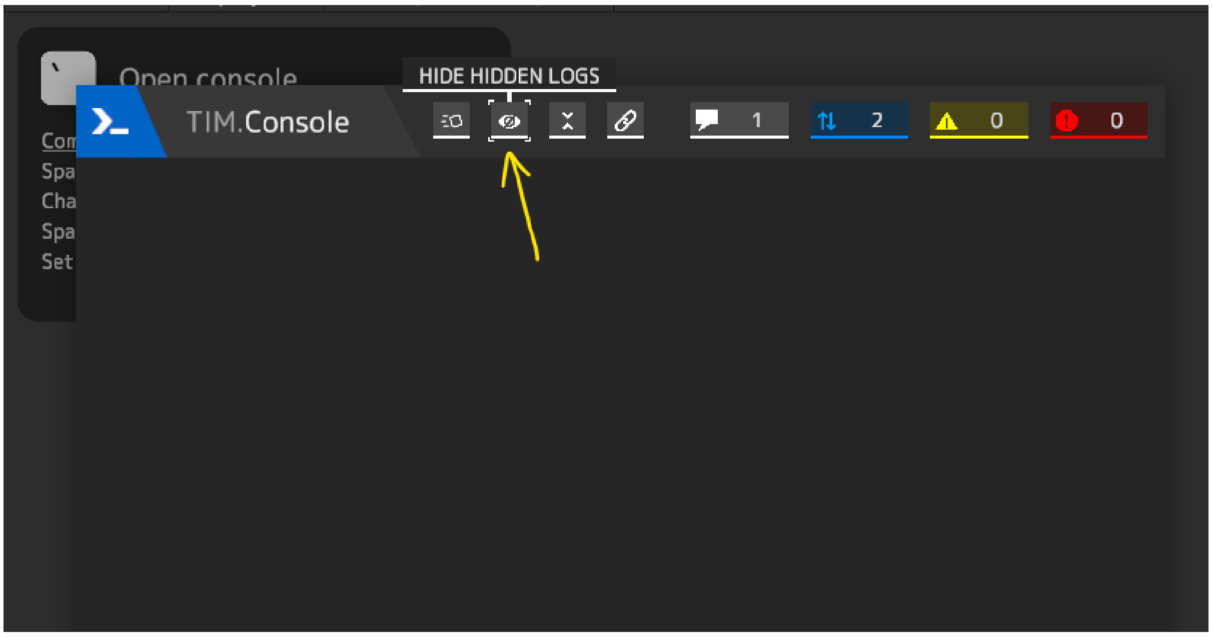


Image downloading

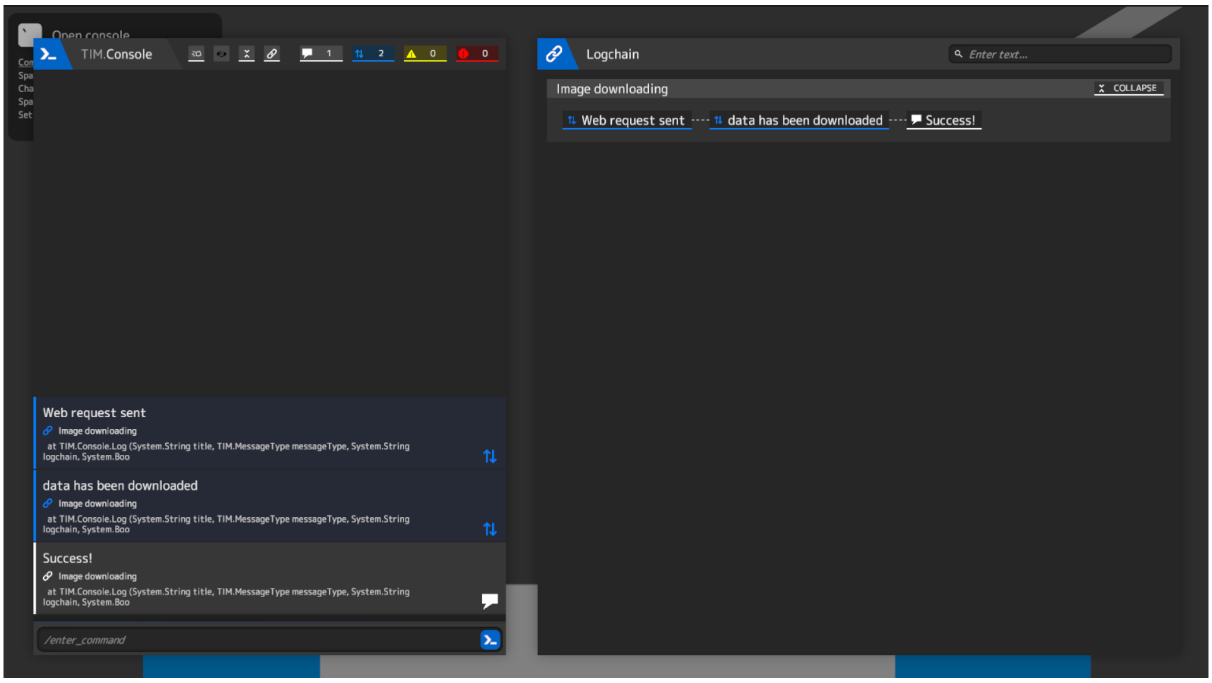
Web request sent ---- data has been downloaded ---- Success!

- Console is clear. But Logchain exists.

You can enable visibility of hidden messages by clicking this button:



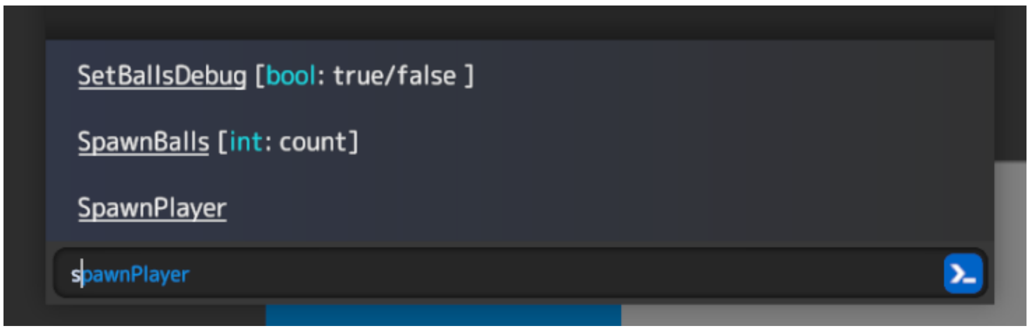
Toggle it and what you can see now:



Web request sent
Image downloading
```
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```
data has been downloaded
Image downloading
```
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```
Success!
Image downloading
```
 at TIM.Console.Log (System.String title, TIM.MessageType messageType, System.String
logchain, System.Boo
```

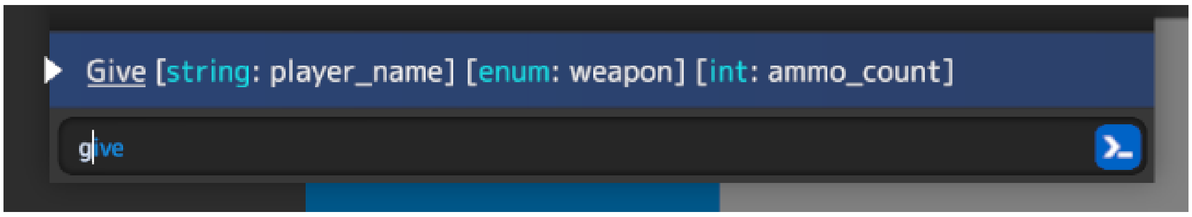
Commands

You can create your own commands.



```
SetBallsDebug [bool: true/false ]
SpawnBalls [int: count]
SpawnPlayer
spawnPlayer
```

Every command has its formula!
You create formulas by your self. Commands can be of any complexity and structure!



```
Give [string: player_name] [enum: weapon] [int: ammo_count]
give
```

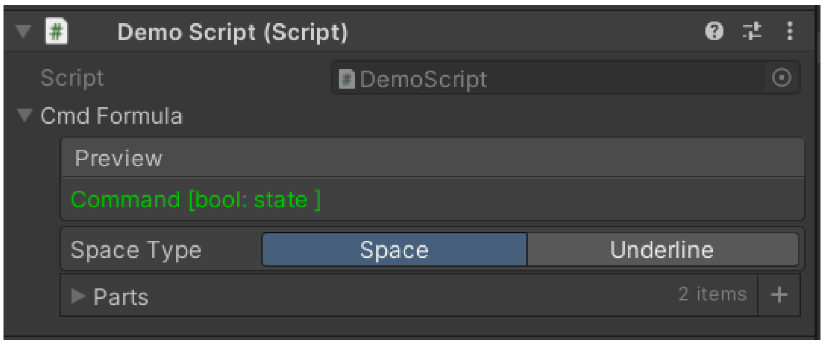
How to create command

For example we want to create command that will spawn balls in our scene.
It contains 2 parts: sentence "Spawn_balls" + Integer "count"
Spawn_balls [int: count]

1. Create `CmdFormula`

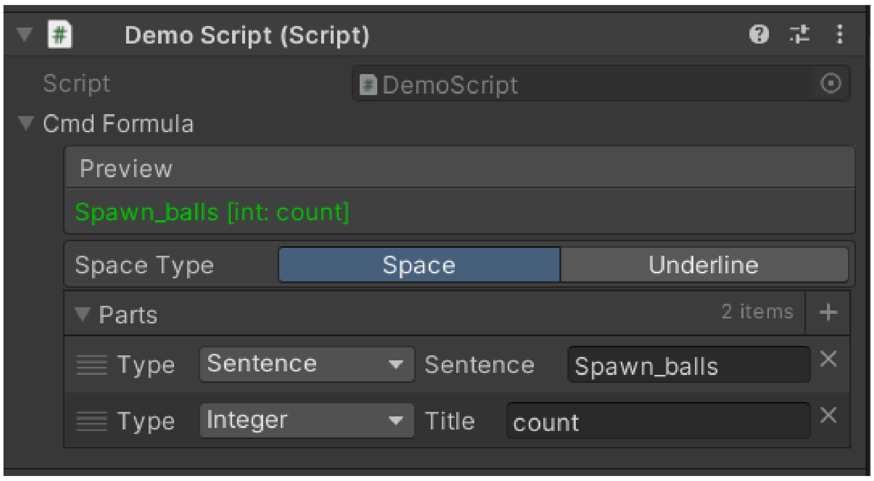
Create `CmdFormula` field in your script.

```
using TIM;
using UnityEngine;

public class DemoScript : MonoBehaviour
{
    public CmdFormula SpawnBallsFormula;
}
```



Demo Script (Script)
Script: DemoScript
Cmd Formula
Preview
Command [bool: state]
Space Type: Space | Underline
Parts: 2 items

Select **SpaceType**: Space or Underline
➕ You can add new part (or remove e excess part) and customize it. We need 2 parts.
Select `PartType` for each part and our formula should look like this:



Demo Script (Script)
Script: DemoScript
Cmd Formula
Preview
Spawn_balls [int: count]
Space Type: Space | Underline
Parts: 2 items
Type: Sentence — Sentence: Spawn_balls
Type: Integer — Title: count

2. Create a method that will be called when executing the command

```
using TIM;
using UnityEngine;

public class DemoScript : MonoBehaviour
{
    public CmdFormula SpawnBallsFormula;

    private void OnSpawnBallsCommand(CmdInputResult result)
    {

    }
}
```

CmdInputResult contains user's input string already separated in parts and parsed to necessary types! So to get balls count you just need to get value from Part that contains balls count value:

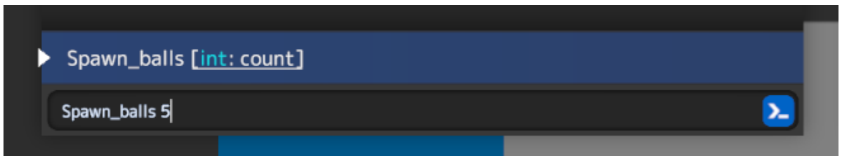
```
using TIM;
using UnityEngine;

public class DemoScript : MonoBehaviour
{
    public CmdFormula SpawnBallsFormula;
    public GameObject BallPrefab;

    private void OnSpawnBallsCommand(CmdInputResult result)
    {
        int ballsCount = result.Parts[1].Integer;

        for (int i = 0; i < ballsCount; i++)
        {
            Instantiate(BallPrefab);
        }
    }
}
```

3. Register your command

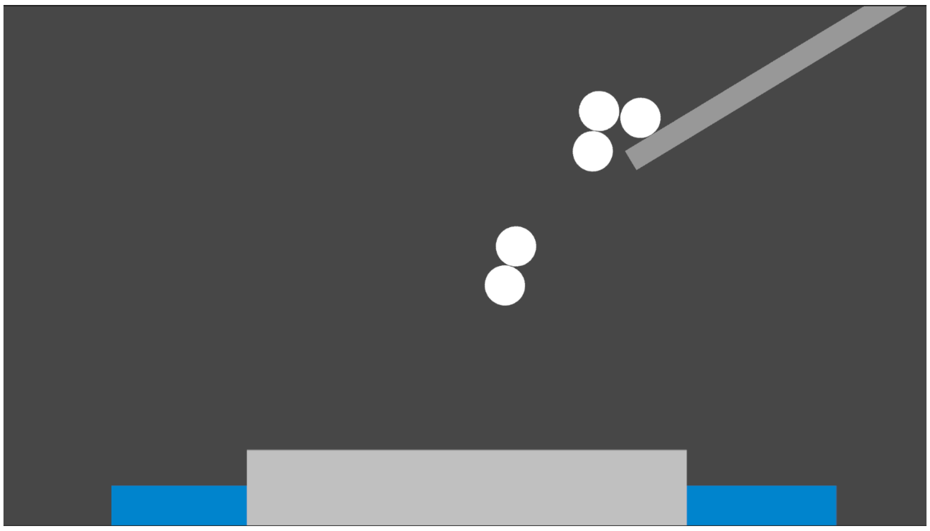
You need to register your command in Console

```
using System;
using TIM;
using UnityEngine;

public class DemoScript : MonoBehaviour
{
    public CmdFormula SpawnBallsFormula;
    public GameObject BallPrefab;

    private void Start()
    {
        TIM.Console.RegisterCommand(SpawnBallsFormula, OnSpawnBallsCommand);
    }

    private void OnSpawnBallsCommand(CmdInputResult result)
    {
        int ballsCount = result.Parts[1].Integer;

        for (int i = 0; i < ballsCount; i++)
        {
            Instantiate(BallPrefab);
        }
    }
}
```

✅ All done!

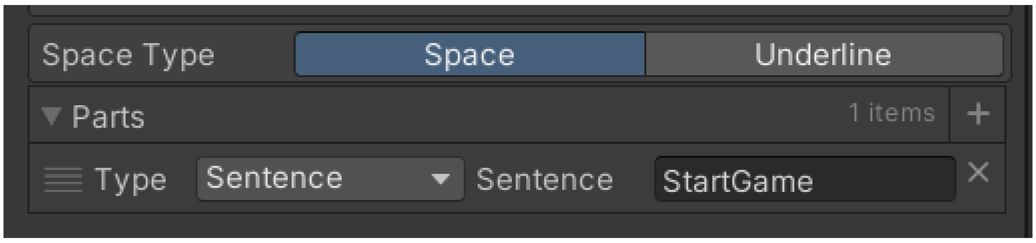


```
Spawn_balls [int: count]
Spawn_balls 5
```



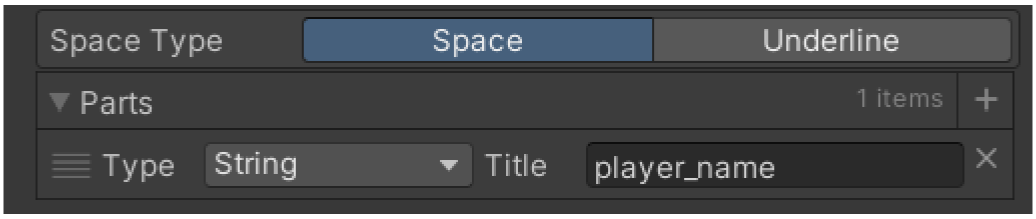
PartType C#

Sentence - fixed string value


Space Type | Space | Underline
Parts | 1 items
Type: Sentence | Sentence: StartGame

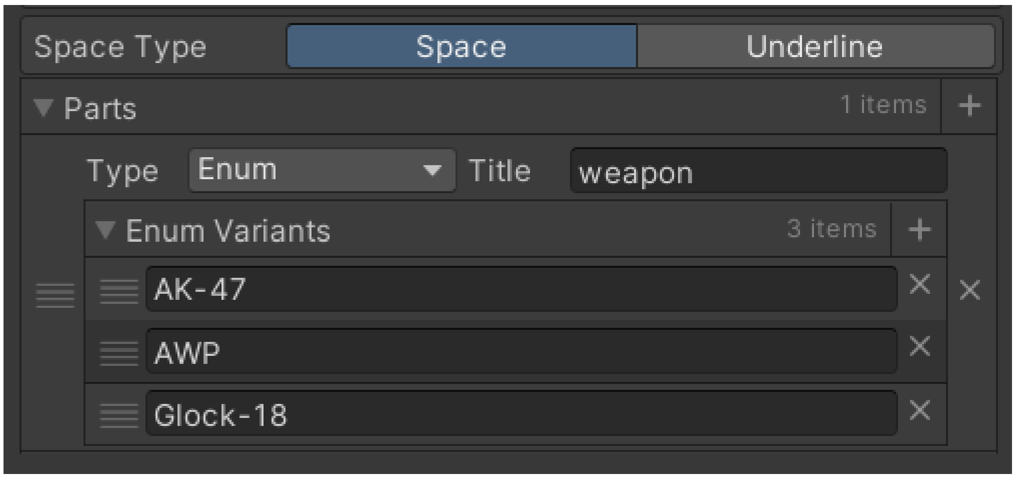
sentence

String - user's custom value. Value that user can write for example: "Johnny Depp"


Space Type | Space | Underline
Parts | 1 items
Type: String | Title: player_name

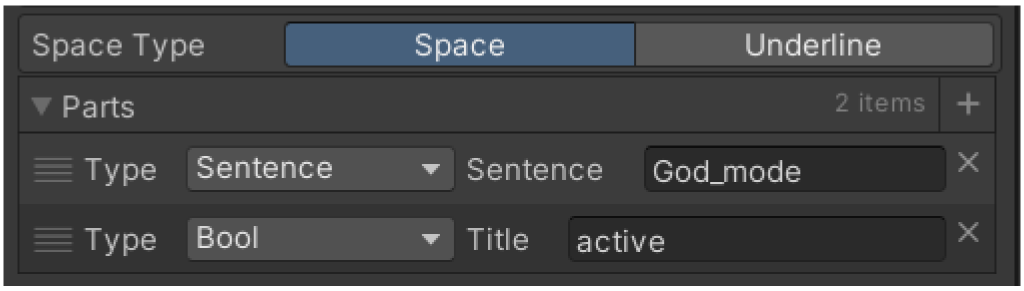
string

Enum - list of given values. Example:


Space Type | Space | Underline
Parts | 1 items
Type: Enum | Title: weapon
Enum Variants | 3 items
AK-47
AWP
Glock-18

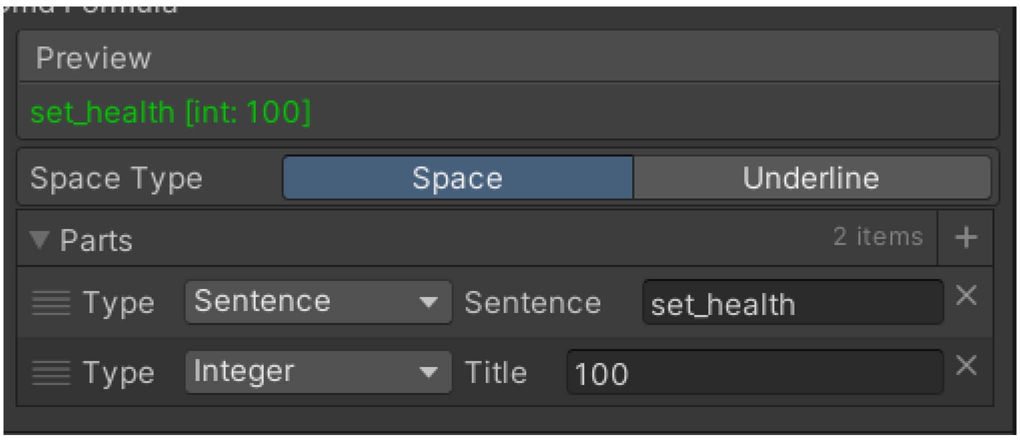
enum

Bool - possible input values: "true", "false", "1", "0"


Space Type | Space | Underline
Parts | 2 items
Type: Sentence | Sentence: God_mode
Type: Bool | Title: active

example of 'sentence' + 'bool'

Integer - Example of possible value: "21"


Preview

set_health [int: 100]

Space Type: Space | Underline

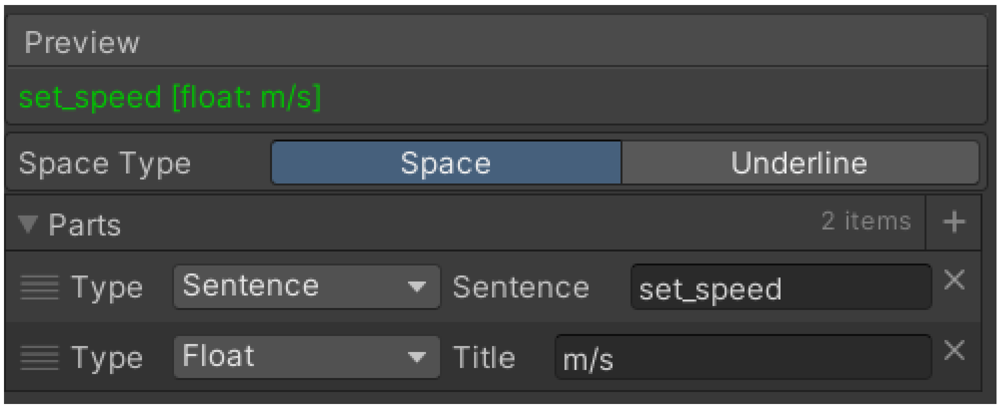
Parts — 2 items

Type: Sentence — Sentence: set_health

Type: Integer — Title: 100

sentence + int

Float - Example of possible values: "15,15" ("15.15" will be incorrect, only comma symbol works)


Preview

set_speed [float: m/s]

Space Type: Space | Underline

Parts — 2 items

Type: Sentence — Sentence: set_speed

Type: Float — Title: m/s

sentence + float

CmdFormula C#

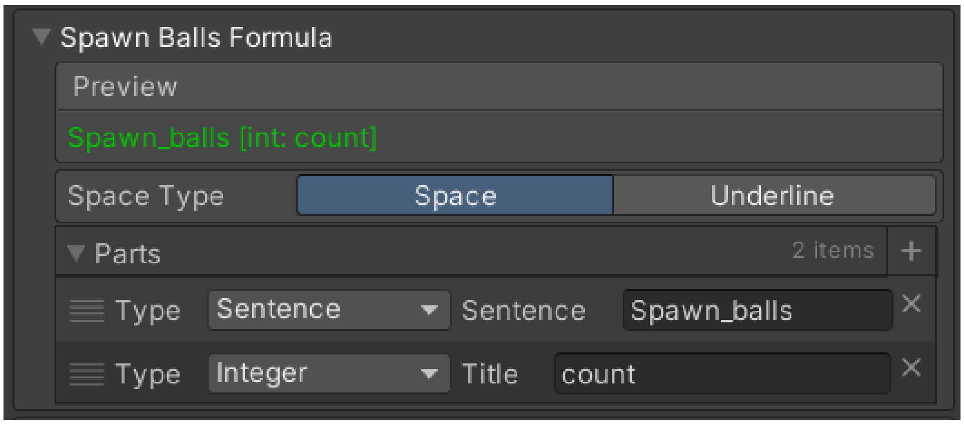
`CmdFormula` contains a list of `CmdFormulaPart`, and a type of space between them: ` ` or `_`
You can create new `CmdFormula` in your script like this:

```
CmdFormula formula = new CmdFormula()
{
    SpaceType = CmdSpaceType.Space,
    Parts = new List<CmdFormulaPart>()
    {
        new CmdFormulaPart("Download_image"),
        new CmdFormulaPart(CmdPartType.String, "URL")
    }
};
```

Just don't forget that you can't use your space symbol in Parts in **Sentence** and **Enum variants**

CmdInputResult C#

Contains list of `CmdInputPart`. List is created based on list of Parts from CmdFormula.
You can select necessary part and extract data from it:
Data type depends on PartType.
If this Part has Integer PartType, so you can extract int value from it:
Example:


Spawn Balls Formula
Preview
Spawn_balls [int: count]
Space Type: Space | Underline
Parts — 2 items
Type: Sentence — Sentence: Spawn_balls
Type: Integer — Title: count

So here is a method that is called when the command is executed:

```
private void OnSpawnBallsCommand(CmdInputResult result)
{
    int ballsCount = result.Parts[1].Integer;
}
```

Lets take a look at all Part Types:

Sentence

```
public int PartIndex;

void OnCommandExecuted(CmdInputResult result)
{
    string sentence = result.Parts[PartIndex].String;
}
```

Float, Int, Bool

```
public int PartIndex;

void OnCommandExecuted(CmdInputResult result)
{
    float floatValue = result.Parts[PartIndex].Float; // if float

    int intValue = result.Parts[PartIndex].Integer; // if int

    bool boolValue = result.Parts[PartIndex].Bool; // if bool
}
```

Enum

`EnumVariant` returns index of selected variant
`EnumVariantString` returns selected variant

```
public int PartIndex;

void OnCommandExecuted(CmdInputResult result)
{
    int enumVariantIndex = result.Parts[PartIndex].EnumVariant;
    string enumVariantString = result.Parts[PartIndex].EnumVariantString;
}
```

Support

For any questions, bug reports, feature requests

Discord: https://discord.gg/PRwCwdPppp

E-mail: funnymanwin@gmail.com

i prefer Discord, because anybody else can find our discussion and maybe find a solution for himself