R Ruan Cantamessa · 56 minutes ago · 2 min read

How to use the CommandLine tool

What is the commandline tool?

The commandline enables developers to easily create or convert existing methods into a commandline argument making the tool very useful for debugging or creating an admin command system.

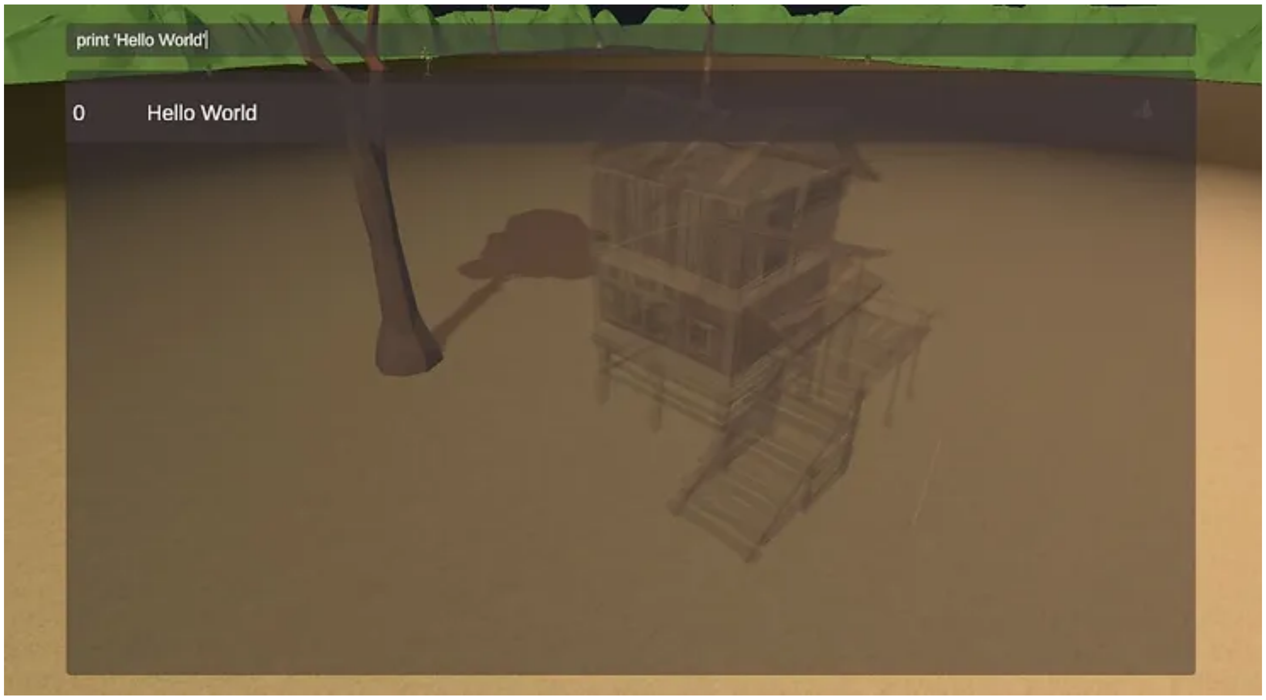
How it works

In the code snippet bellow you will notice an attribute called [Command("print")] where the string defines the name of the command to use in the commandline to reference the method. (Ensure that the commandline component is in your scene and active as it will only run commands in the scene)

```
public void ConsoleLog(object content)
{
    var textLog = SpawnTextLog();
    textLog.textarea.text = content.ToString();
}

[Command("print")]
private void Log(string text)
{
    ConsoleLog(text);
}
```

this command is a default command that comes with the commandline package as well with a default commandline UI as seen below.



```
print 'Hello World'
0       Hello World
```

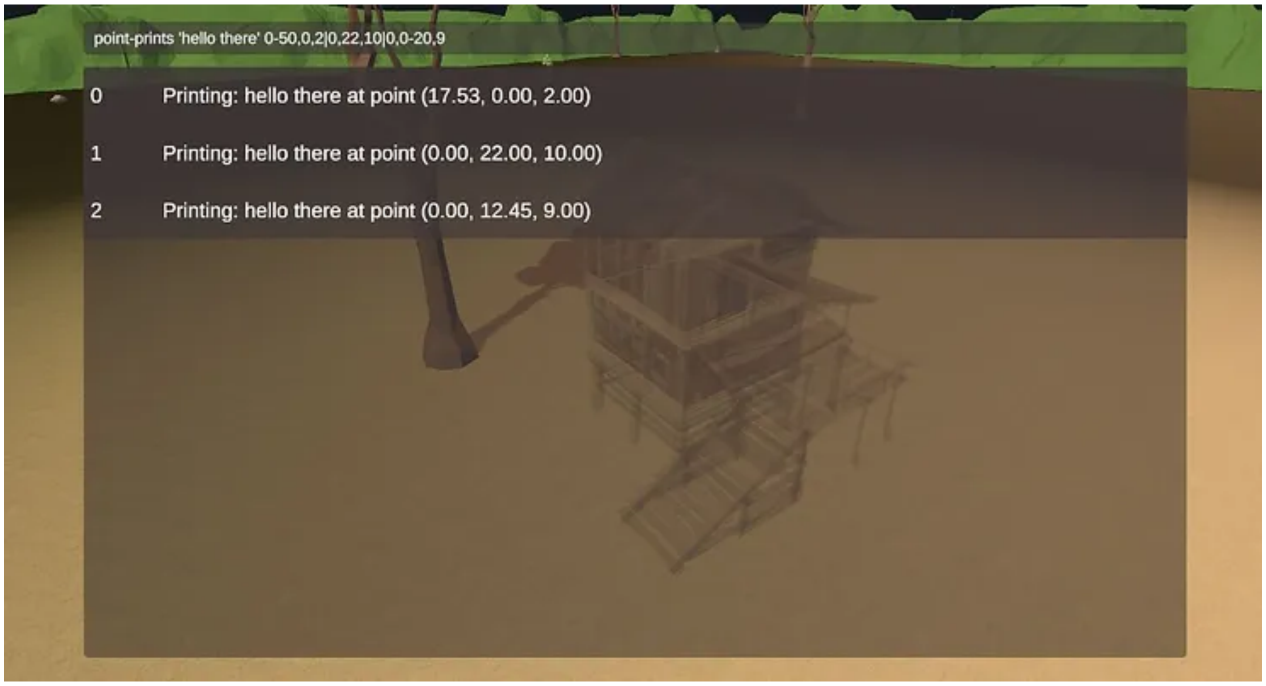
Note that single quotes that surrounding the 'Hello World' text. The commandline required center symbols or pattern to determine its type. Let us have a look at all the types you might face and the corresponding commandline representations.

Type	CommandLine Representation Example	Requires Special Symbols	Description
number	0	NO	type number includes type int, float and double.
string	'hello world'	YES	everything between single quotes are considered a string.
Vector2	0,0	YES	two commas are required to represent the separation between the x and y values
Vector3	0,0,0	YES	three commas are required to represent the separation between the x, y and z values
Quaternions	0,0,0,0	YES	four commas are required to represent the separation between the x, y, z and x values

			values
Random	1-49	YES	values placed between a dash symbol will create a random number based on the range given where the left side represent the min value and the right side represent the max value
Array or List	15|20	YES	all types above can be places in a array or list by placing them between the straight line symbol

Lets make a more advanced example displaying the use of the string, vector3, array and random types. In this example we want to print text at random points between 0 and 50. First we create our method containing a string and Vector3[] type arguments then add our command attribute on top and bind it to the command point-prints.

```
      [Command("point-prints")]
      private void LogAtPoints(string text, Vector3[] points)
      {
          foreach (var point in points)
          {
              ConsoleLog($"Printing: {text} at point
  {point.ToString()}");
          }
      }
```



```
point-prints 'hello there' 0-50,0,2|0,22,10|0,0-20,9
0       Printing: hello there at point (17.53, 0.00, 2.00)
1       Printing: hello there at point (0.00, 22.00, 10.00)
2       Printing: hello there at point (0.00, 12.45, 9.00)
```

let's break down the command and have a look at what each argument meant:

point-prints 'hello there' 0-50,0,2|0,22,10|0,0-20,9 (each argument is separated by a space)

- **point-prints:** The name of the command linked to our method.
- **"hello there":** Type string argument that matches the type in the parameter of the method.
- **0-50,0,2|0,22,10|0,0-20,9:** A list or array of type Vector3 that matches the type in the parameter of the method. We also told our commandline that we want to have a random x value at our first element and a random y value at our third element in the array.

How to add the commandline to custom UI

The commandline is defined as a singleton so all that is required for the commandline to work is for the component to be placed on the scene and then add this one liner into your custom script.

```
CommandLine.Instance.RunCommand(text);
```

Conclusion

After reading this blog I hope you will find this tool helpful and useful to your project. This version of the commandline is free and opensource available on the asset store and GitHub