Simple Text Gizmos

This document contains all the information you need to get started using the tool.

Table of content

- Setup
- How to use
- Adding or editing symbols

Setup

Simple Text Gizmos requires no setup apart from importing it in a project.

How to use

To use Simple Text Gizmos simply call the ‘TextGizmo.Draw’ method in the ‘LombaxGuy.STG’ namespace from either ‘OnDrawGizmos’ or ‘OnDrawGizmosSelected’.

```
using LombaxGuy.STG;

public class Example : MonoBehaviour
{
    private void OnDrawGizmos()
    {
        TextGizmo.Draw("Hello World!", Vector3.zero);
    }
}
```

Adding or editing symbols

By default Simple Text Gizmos supports all letters from A-Z in both upper and lower case, the numbers from 0-9, a whitespace character, and the following symbols:
½ ( ) [ ] { } , . ; : ! ? + − * _ | < > = / \ % & @ ‟ ‛ # ¤ $ ~ ^
` ´

If you find yourself in need of a character that is not supported by default you can easily create it yourself and add it to the dictionary of known symbols.
To do this you first have to create a new PointArray. A PointArray is a struct containing an array of floats and a DrawType. The float array is structured in pairs to represent 2D points. The points of the default symbols are all defined with (0, 0) representing the bottom left and (3, 6) is the top right. If you don’t follow the same dimensions, your symbols will not have the same size as the default ones. The DrawType indicates whether the symbols should be

drawn as one continuous line or a series of line pieces.
As an example below is the point array for the number 1:

```
private static readonly PointArray POINTS_NUM1 = new PointArray
{
    DrawType = DrawType.LinePieces, // draw as line pieces
    Points = new float[]
    {
        0, 5,
        1.5f, 6,
        1.5f, 6, // point is repeated as we draw both from at to it
        1.5f, 0,
        0, 0,
        3, 0
    }
};
```

We now have to add the point array to the dictionary of known symbols. The dictionary is called ‘SYMBOLS’ and should have a char as the key and the PointArray as the value.

```
// dictionary containing all our symbols, characters are given as as unicode
private static readonly Dictionary<char, PointArray> SYMBOLS = new
Dictionary<char, PointArray>()
{
    { '\u0020', default }, // space
    { '\u0030', POINTS_NUM0 },
    { '\u0031', POINTS_NUM1 }, // PointArray is added to the dictionary
    { '\u0032', POINTS_NUM2 },
    ...
```

Editing existing symbols already have existing PointArrays and are added to the dictionary so you only have to change the points for the characters you want to edit.