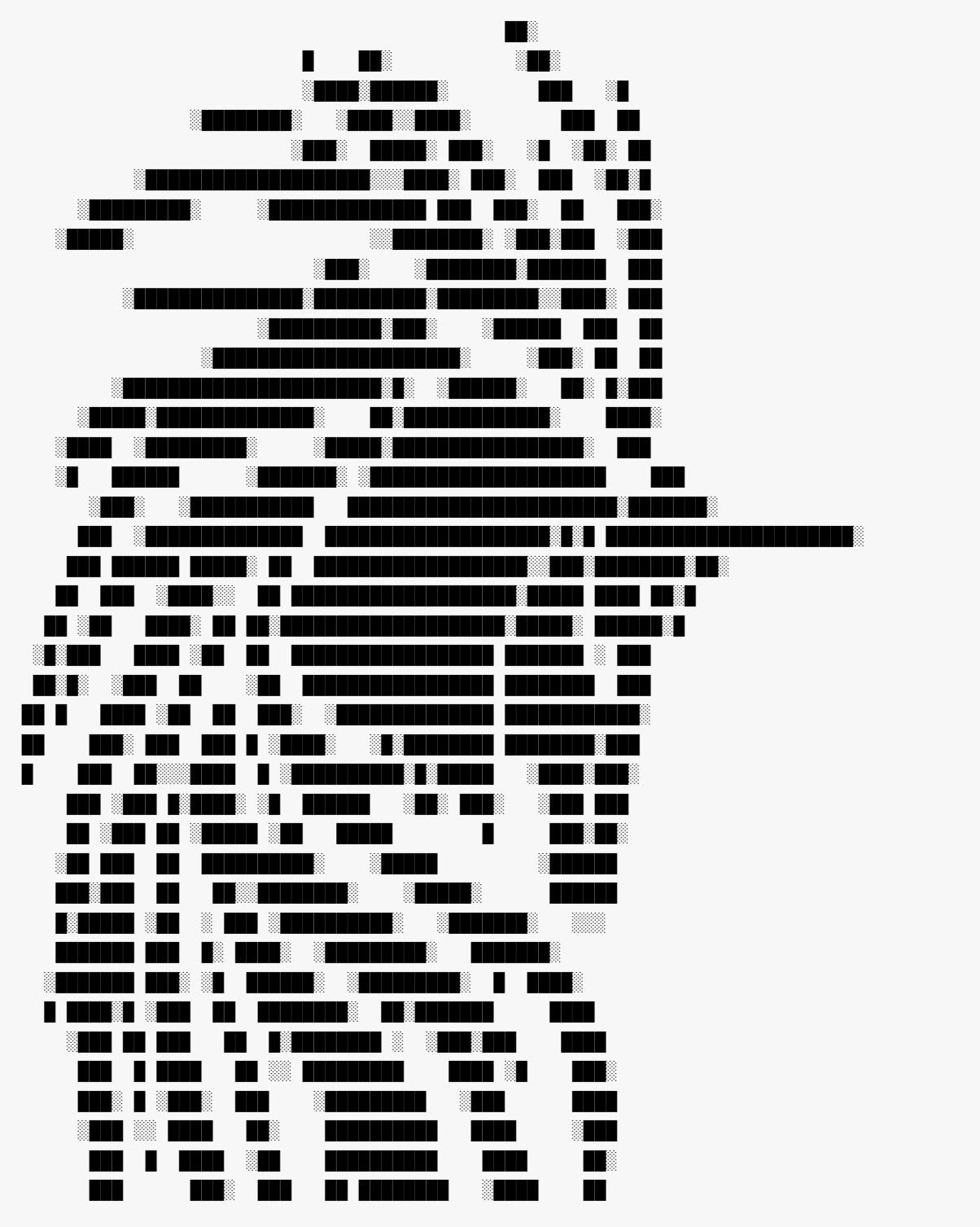
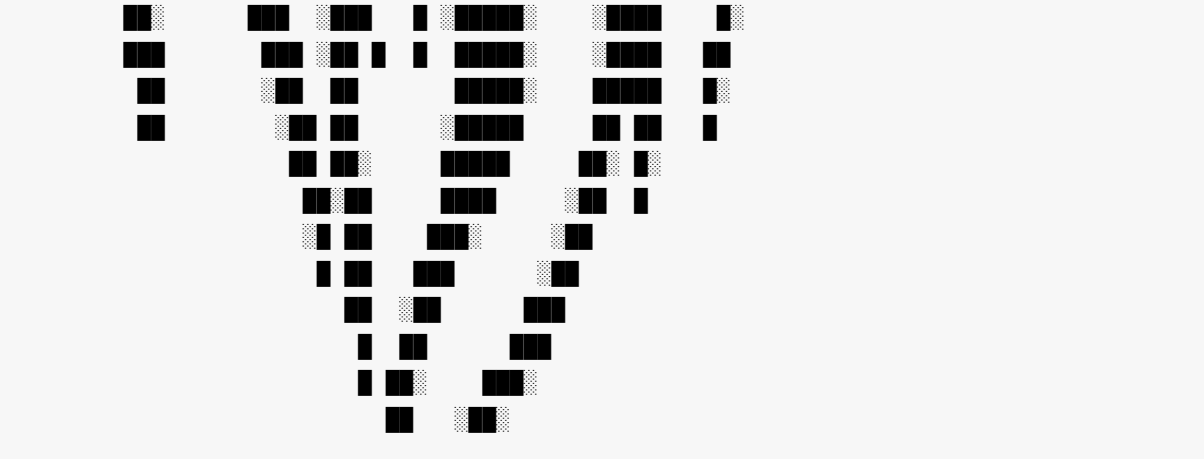
# Table of Contents

# Description

The process I learn Unity3D.
Some keypoint and note about everything.
In summary, everything I'd like to record.

Project on Gitlab





# Timeline

## Playable Track

### OnPlayableCreate

#### Timeline

### OnGraphStart

#### Timeline

### OnGraphStart

### OnBehaviourPause

#### Timelineclip

### OnGraphStart

#### Timeline

### OnBehaviourPause

### OnGraphStop

#### TimelineNone

### OnGraphStop

#### TimelineNone

### OnBehaviourPlay

#### clip

### OnBehaviourPause

#### clip

### PrepareFrame

### clip

### ProcessFrame

## Control Track

### prefab

## Activation Track

# DynamicBone

## Root

`The root of the transform hierarchy to apply physics.`

## Update Rate

`Internal physics simulation rate.`
30

## Update Mode

`How much the bones slowed down.`

- Normal
- Animate Physics

- Unscaled Time

## Damping

`How much the force applied to return each bone to original orientation.`
AnimationCurve
Damping

## Elasticity()

`How much bone's original orientation are preserved.`

Damping

## Stiffness

`How much character's position change is ignored in physics simulation.`

Damping

## Inert

`Each bone can be a sphere to collide with colliders. Radius describe sphere's size.`
(Dynamic Bone ColliderCollidersDynamic Bone ColliderDynamic Bone Collider)
Damping

## Colliders

Dynamic Bone Collider

# Spine

## Spine.Unity.SkeletonAnimation

SpineSpine

### state

Spine.AnimationStateAwake

```
// Trank0“stand”
skeletonAnimation.state.SetAnimation(0, “stand”, false);

// Track0“run”Track0“run”
skeletonAnimation.state.AddAnimation(0, “run”, true, 0f);
```

### loop

### AnimationName

```
skeletonAnimation.AnimationName = ""
```

### skeleton.FlipX

bool

### timeScale

## Track

TrackTrack:

```
// Track 0Track 1
skeletonAnimation.state.SetAnimation(0, "run", true);
skeletonAnimation.state.SetAnimation(1, "shoot", false);
```

# TrackEntry

TrackEntry:

```
var trackEntry = skeletonAnimation.state.GetCurrent(myTrackNumber); // null
```

## TrackEntry.Time,TrackEntry.lastTime

Spine30

```
// 10"dance"
var trackEntry = skeletonAnimation.state.SetAnimation(0, "dance", false);
trackEntry.Time = 10f/30f;

// TimeTrackEntry
skeletonAnimation.state.SetAnimation(0, "dance", false).Time = 10f/30f;

// lastTime.Time lastTime0 Time0.TimeUpdate/
```

## TrackEntry.animationEnd

## TrackEntry.TimeScale

SkeletonAnimation.timeScale
timeScale0timeScale = 0

# Spine.Animation

Spine.AnimationTimeline

# Spine.AnimationState

## Start

()

## End

()

## Complete

## Dispost

## Interrupt

## Event

Spine-Unity `AtlasAsset`

`MeshRenderer` `SkeletonRenderer` `AtlasAssets` Unity

`MeshRenderer` `AtlasAsset`

AtlasAssetSkeletonRendererSpine

SkeletonRendererMeshRenderer

AB

draw calls

MeshRenderer.material

`Renderer.material` SkeletonRenderer

`Renderer.sharedMaterial` Spine

Unity Renderer.SetPropertyBlock `SkeletonRenderer` `SkeletonAnimation` `MeshRenderer` MeshRendererMaterialPropertyBlock **MaterialPropertyBlock mpb = new Material mpb.SetColor("_FillColor", Color.red); // "_FillColor" GetComponent<MeshRenderer>().SetPropertyBlock(mpb);**

> - Renderer.SetPropertyBlockMaterialPropertyBlocks
> - MaterialPropertyBlock `SetPropertyBlock` MaterialPropertyBlock
> - : `Shader.PropertyToID(string)` IDIDStringMaterialPropertyBlockSetterID

## Spinemesh

SpinemeshmeshUVmeshSpinemesh

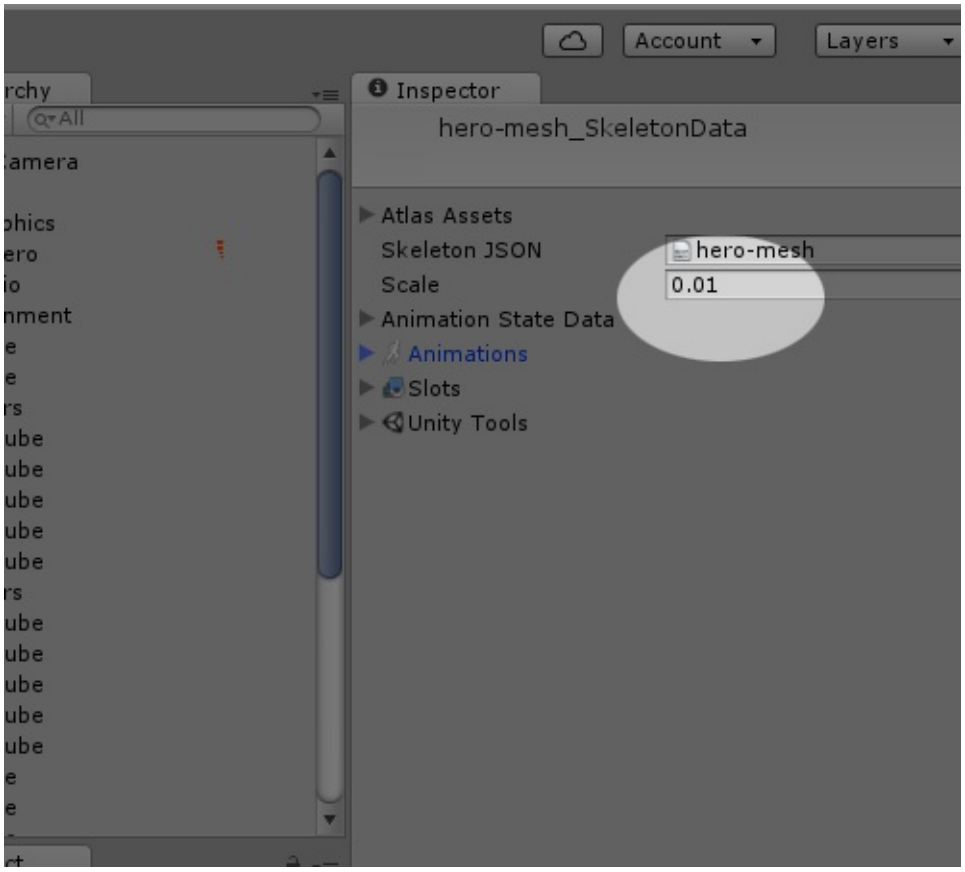
**Sorting LayerSorting Order** `SkeletonRenderer` / `SkeletonAnimation` Inspector `MeshRenderer` sorting layer sorting order .

MeshRendererInspector `MeshRenderer` serialized/storedSkeletonRenderer

**Sorting Layer** **Sorting Layer**

# Spine

Spine 1:10.01




# or

Skeleton.FlipXSkeleton.FlipY

- Unitydraw calls
- 2D
- XYUnity2D

- 

## UISkeletonGraphicUI

## Spine.Unity.SkeletonGraphic

UISpine

### SkeletonData

#### Animations

# Refs

- Unity(API)
- API

# C#

## C#

C C++ C `Del` void C#

```
public delegate void Del(string message);
```

:C#

```
// Create a method for a delegate.
public static void DelegateMethod(string message)
{
    System.Console.WriteLine(message);
}
```

C#

```
// Instantiate the delegate.
Del handler = DelegateMethod;

// Call the delegate.
handler("Hello World");
```

```
for (int i = 1; i <= 4; ++i)
{
    RigisterFullPathObjectEvent("ResetBtn"+i, p =>
    {
        Debug.Log("click");
        int x = i;
        MainPlayer.Inst.PlotProp().Reset(x);
        LoadPlotInfo();
    });
}
```

```
for (int i = 1; i <= 4; ++i)
{
    int x = i;
    RigisterFullPathObjectEvent("ResetBtn"+x, p =>
    {
        Debug.Log("click");
        MainPlayer.Inst.PlotProp().Reset(x);
        LoadPlotInfo();
    });
}
```

- MSDN C#

# XLua

## lua

- **lua**

  `luaenv.DoString("print('success')");`

- **requirelua**

  `luaenv.DoString("require 'LuaCode'");` "*.lua.txt"

- **loader**

```
luaenv.AddLoader(delegate(ref string filepath)
{
  string code = Resources.Load(filepath).ToString();
  return System.Text.Encoding.UTF8.GetBytes(code);
});
// ResourcesLuaCode.lua.txtLuaCode.lua.xml
luaenv.DoString("require('LuaCode.lua')");
```

## C#Lua

- **`LuaEnv.Global`**

  `string a = luaenv.Global.Get<string>("a");`

# SUIFW

```
    #region
    //
    public override void Display()
    {
        base.Display();
    }

    //Freeze
    public override void Redisplay()
    {
        base.Redisplay();
    }

    //
    public override void Freeze()
    {
        base.Freeze();
    }

    //(“”)
    public override void Hiding()
    {
        base.Hiding();
    }
    #endregion
```

```
CloseUIForm();
```

```
RigisterFullPathObjectEvent("Btn", p =>
ClickBtn());
```

BtnClickBtn()

```
CurrentUIType.UIForms_Type = UIFormType.Normal;
```

```
    //UI
```

```
    public enum UIFormType
    {
        //
        Normal,
        //
        Fixed,
        //
        PopUp
    }
```

```
CurrentUIType.UIForms_ShowMode = UIFormShowMode.HideOther;
```

```
    //UI
    public enum UIFormShowMode
    {
        //
        Normal,
        //
        ReverseChange,
        //
        HideOther
    }
```

```
    //UI
    public enum UIFormLucenyType
    {
        //
        Lucency,
        //
        Translucence,
        //
        ImPenetrable,
        //
        Pentrate
    }
```

```
UIManager.GetInstance().ShowUIForms(UIProConst.Form);
```

Form

## Display

: `Awake()` → `OnEnable()` → `Start()`

- Unity UISUIFW——LiuGuozhu
- UnityEvent System – Dispatcher , pdf

# GPU Instancing

GPU Instancing

# C#

- (C#)

# Mask

## Image

## Shader

### Transparent

# Shader

_Time float4 Time (t/20, t, t*2, t*3), use to animate things inside the shaders.

_SinTime float4 Sine of time: (t/8, t/4, t/2, t).

_CosTime float4 Cosine of time: (t/8, t/4, t/2, t).

unity_DeltaTime float4 Delta time: (dt, 1/dt, smoothDt, 1/smoothDt).

```
_Time.x = time / 20
_Time.y = time
_Time.z = time * 2
_Time.w = time * 3
```

[Toggle]
01

```
[Toggle]_Start("if Start", int) = 0
```

_Start

```
_TimeStart += (_Time.y - _TimeStart) * (1-_Start);
```

# Miscellaneous

## IEnumerator

```
IEnumerator Cro()
{
    while (True)
    {
        yield return new WaitForSeconds(1);
    }
}
```

```
StartCoroutine(Cro());
```

```
StopCoroutine("Cro");
```

```
StopAllCoroutines();
```

```
IEnumerator Cro0()
{
    while (True)
    {
        // Cro0Cro1
        yield return StartCoroutine(Cro1());
    }
}
```

125KGC

`yield return null;`

```
public static implicit operator ( )
{
   return ;
}
```

```
public static explicit operator ( )
{
   return ;
}
```

```
public static OperatorTest operator + (OperatorTest o1, OperatorTest o2)
{
    OperatorTest o = new OperatorTest();
    o.Value = o1.Value + o2.Value;
    return o;
}
```

## Lambda

```
delegate int del(int i);
static void Main(string[] args)
{
    del myDelegate = x => x * x;
    int j = myDelegate(5); //j = 25
}
```

```
    (input-parameters) => expression
```

```
() => SomeMethod()
```

```
    (x) => x * x
```

```
    x => x * x
```

```
    (x, y) => x == y
```

```
    (int x, string s) => s.Length > x
```

# ref

UI

```
    go.SetSiblingIndex(index);
```

# Serializable

`[System.Serializable]` ScriptableObjectnamespace(2018.1.0f2)

# Mask

Mask

uxux `Random.Range(u, x)`

## List<>

```
new System.Random().Shuffle(aList);
```

```
Dictionary<int, StudentName> students = new Dictionary<int, string>()
{
    { 111, "1"},
    { 112, "2"},
};
```

## Raycast Target

ImageText `Raycast Target`

## EventTriggerScrollRect

EventTriggerScrollRectContentScrollRect ButtonWithLongPressButtonUpdate() IsPressed()ButtonOnClick

## Horizontal Layout Group

## Vertical Layout Group

## Grid Layout Group

DOTween `DOTween.To()`

- 
- `[SerializeField]` `class` `[Serializable]`

# C#Miscellaneous

## List

- ToArray()

```
List<string> t = new List<string>(); //original
List<string> t2 = new List<string>(t.ToArray()); // copy of t
```

- ForEach

```
list1.ForEach(i => list2.Add(i));
```

- ConvertAll()

```
namespace Delegates
{
    class X
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }

    class Y
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            List<X> x = new List<X>();
            for (int i = 0; i < 100; i++)
                x.Add(new X { Id = i, Name = string.Format("x_{0}", i.ToString()) }
);
            // copy x to y
            List<Y> y = new List<Y>(x.ConvertAll<Y>(e => { return new Y { Id = e.Id
, Name = e.Name }; }));
            Debug.Assert(x.Count == y.Count);
        }

    }
}
```

ListAsReadOnly

```
List<int> a = new List<int> {1, 2, 3, 4, 5};
IList<int> b = a.AsReadOnly();
```

## .aspx)

```
list.Sort((info0, info1) =>
{
    return info0.CompareTo(info1);
});
```

## .aspx)

```
Console.WriteLine("Path.AltDirectorySeparatorChar={0}",
    Path.AltDirectorySeparatorChar);
```

# UGUI Rich Text

## b

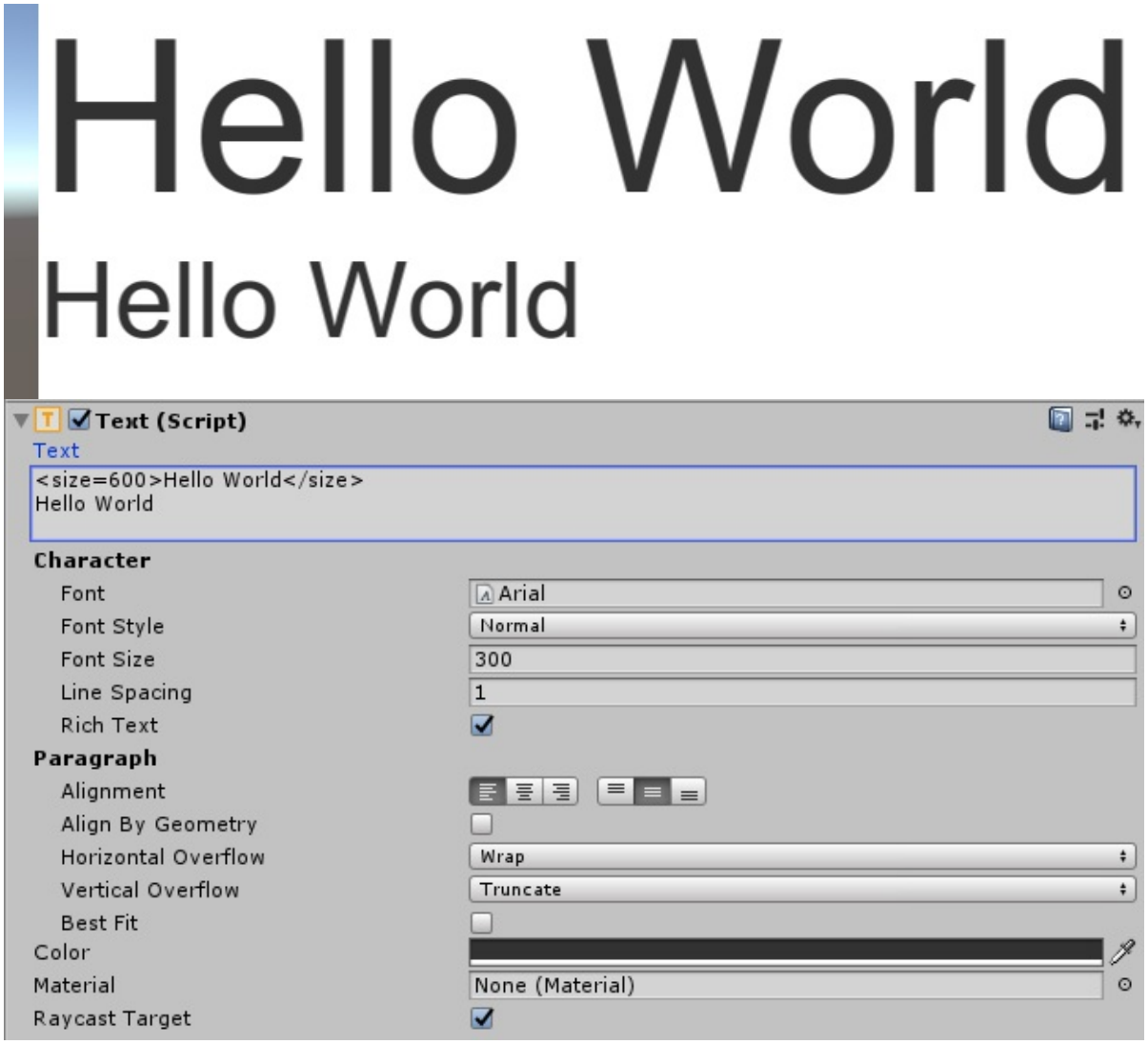
```
<b>Hello World</b>
Hello World
```


## i

```
<i>Hello World</i>
Hello World
```


## size

```
<size=40>Hello World</size>
Hello World
```


Hello World
Hello World
Text (Script)
Text
<size=600>Hello World</size>
Hello World
Character
Font
Arial
Font Style
Normal
Font Size
300
Line Spacing
1
Rich Text
Paragraph
Alignment
Align By Geometry
Horizontal Overflow
Wrap
Vertical Overflow
Truncate
Best Fit
Color
Material
None (Material)
Raycast Target

## color

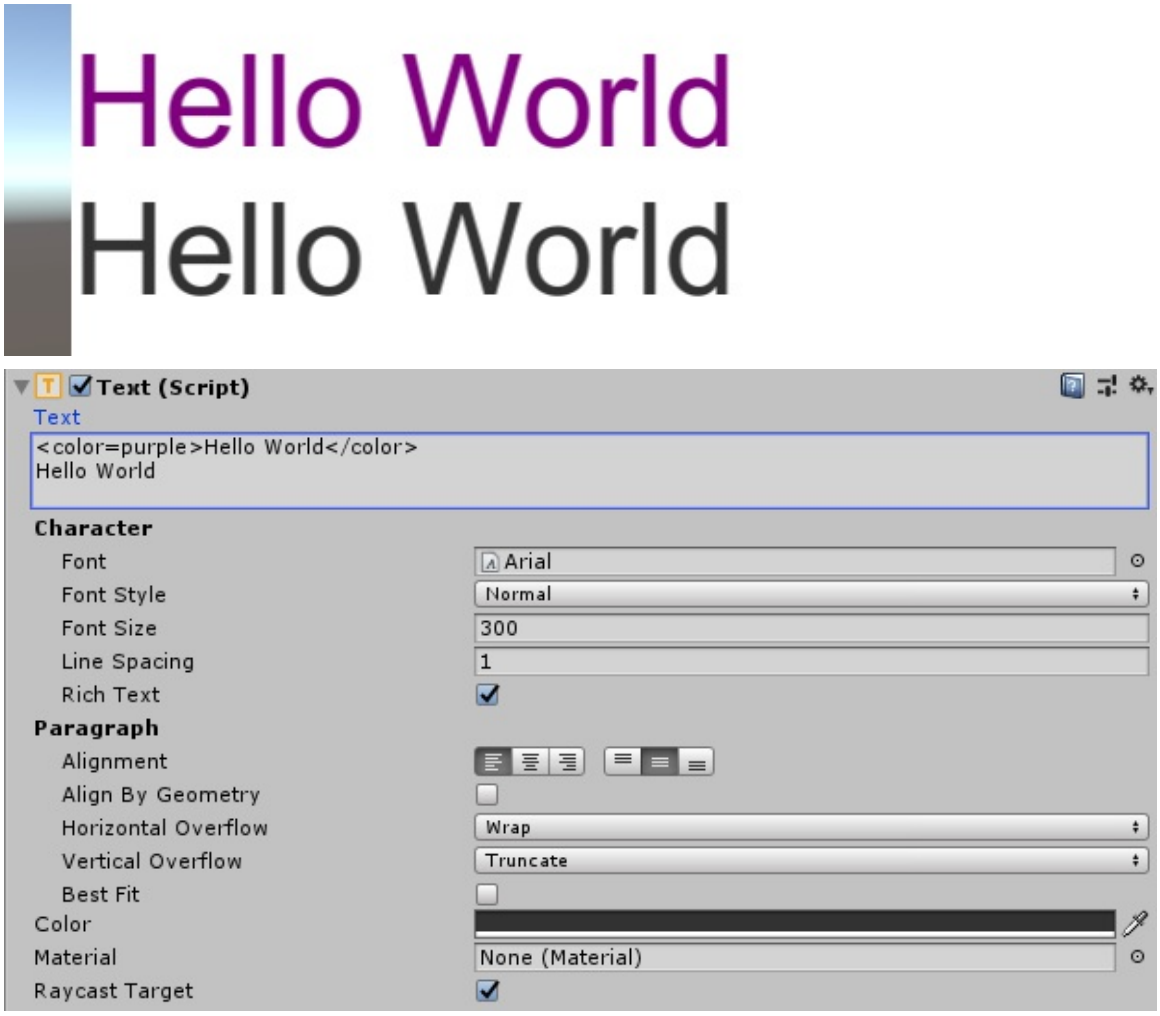
| Color name | Hex value | Swatch |
|---|---|---|
| aqua (same as cyan) | `#00ffffff` | |
| black | `#000000ff` | |
| blue | `#0000ffff` | |
| brown | `#a52a2aff` | |
| cyan (same as aqua) | `#00ffffff` | |
| darkblue | `#0000a0ff` | |
| fuchsia (same as magenta) | `#ff00ffff` | |
| green | `#008000ff` | |
| grey | `#808080ff` | |
| lightblue | `#add8e6ff` | |
| lime | `#00ff00ff` | |

| | | |
|---|---|---|
| magenta (same as fuchsia) | #ff00ffff | |
| maroon | #800000ff | |
| navy | #000080ff | |
| olive | #808000ff | |
| orange | #ffa500ff | |
| purple | #800080ff | |
| red | #ff0000ff | |
| silver | #c0c0c0ff | |
| teal | #008080ff | |
| white | #ffffffff | |
| yellow | #ffff00ff | |



- 

```
<color=purple>Hello World</color>
Hello World
```

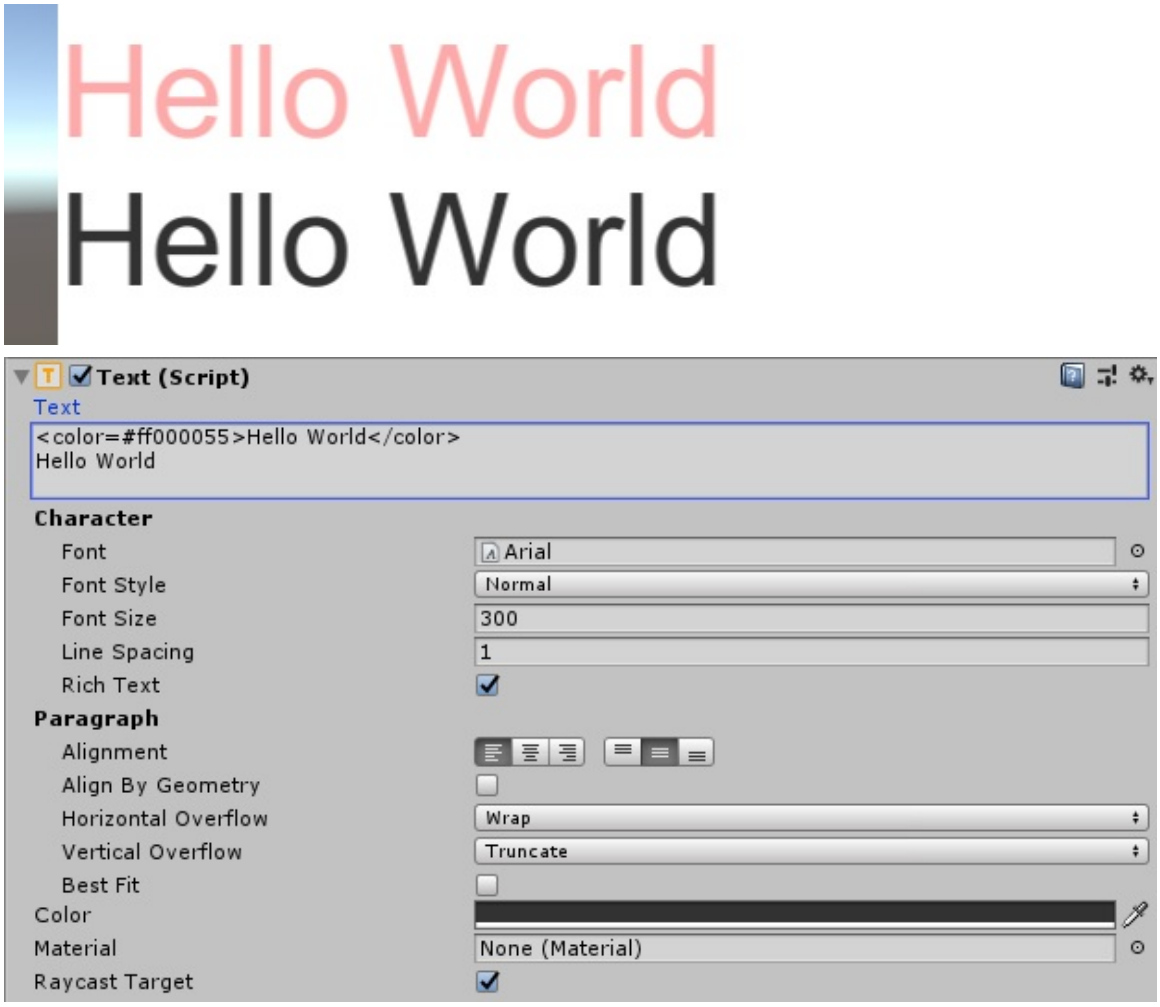



- RGB

```
<color=#ff0000>Hello World</color>
Hello World
```


- RGBA

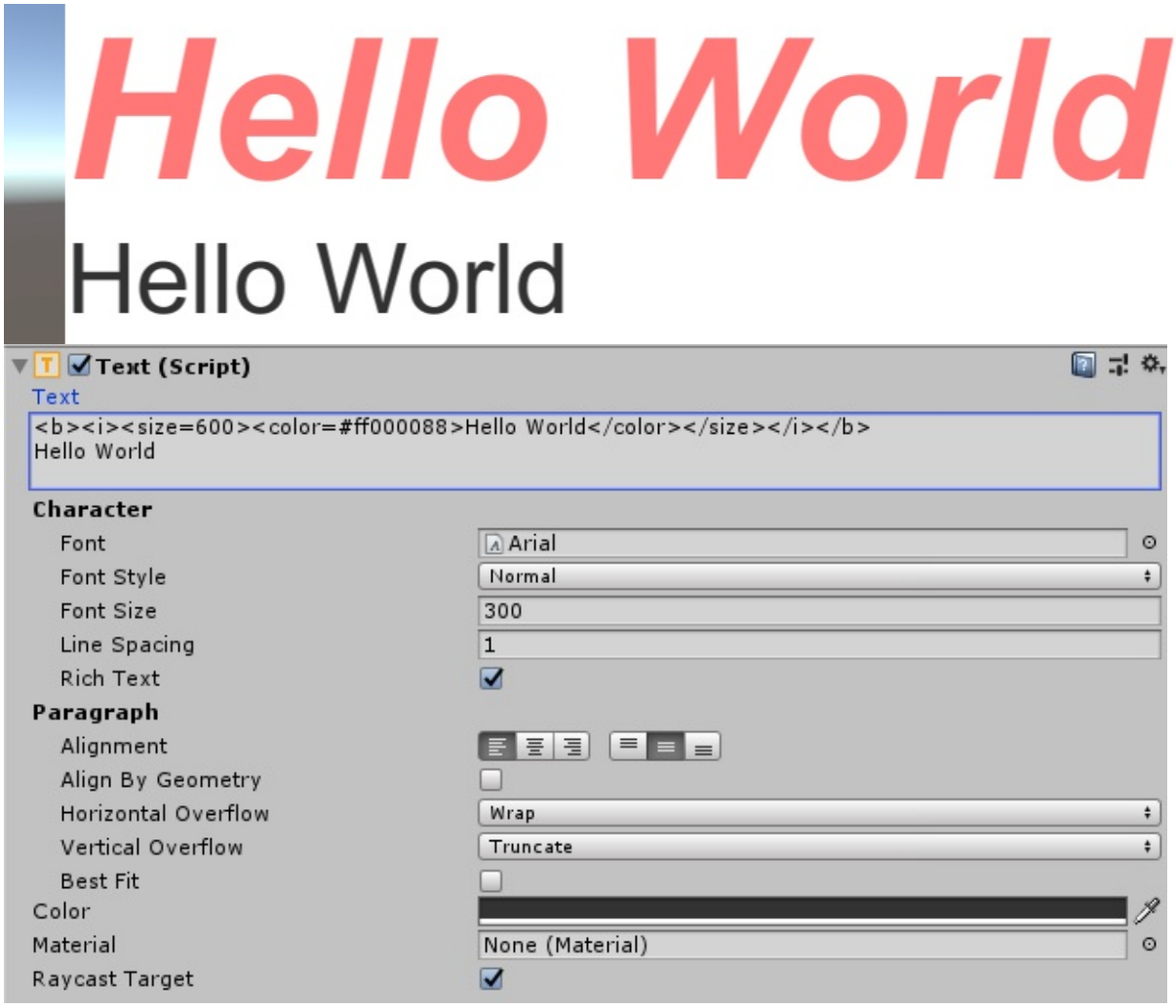
```
<color=#ff000055>Hello World</color>
Hello World
```




## material

text mesh `<material=1>TestTest</material>`

```
<b><i><size=600><color=#ff000088>Hello World</color>
</size></i></b>
Hello World
```




# Refs

UGUIRich Text -
Unity - Manual: Rich Text
unity4.6Ugui--------UGUI Rich Text -