暑培-Unity3D

SAST

2023 年 8 月 9 日

目录

当前进度

Canvas

Unity 所有 UI 部件的载体。

- Screen space(overlay)
  Unity 把游戏界面渲染后，在画面某处上方加上 UI 元素。
- Screen space(Camera)
  将画布设置在某个相机前面。
- World space
  将画布视作场景的一部分。

Button

按钮功能的绑定：
先把按钮点击触发的入口定义好：

```
public void BtnClick() {
    debug.Log("Button clicked");
}
```

接着把带有上述函数的脚本，挂接在某个 GameObject 上。
然后打开右面板 (Inspector)，把挂接该脚本的 GameObject 绑定，
并且找出脚本，选取该功能函数。

动态 UI 部件

类似于 Button，也带有状态改变时触发行为入口，不过都叫 OnValueChanged。
下面列举不同 UI 部件状态改变所触发的函数，具体带上什么输入。

- Slider: float，滑杆的相对位置。
- Toggle: bool，是否有勾选。
- Scrollbar: float，类似于 Slider.
- Dropdown: int，下拉表单的被选序号。
- InputField: string，当前输入框里的文字。
  注意：其还带有 OnEndEdit，指的是用户把焦点移向别的地方所触发的行为。

Image, RawImage, Text

例子有

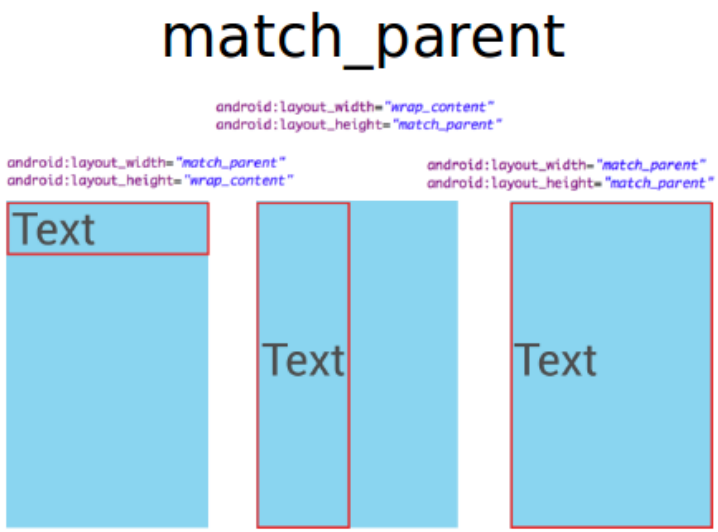
- Image: 渲染 Sprite.
- RawImage: 渲染图片。
- Text, 渲染文字，常见于其他动态部件的子部件。
- Panel: 可以设置不透明度的 Image, 一般为单色，作背景、进度条用。

TextMeshPro: Unity 提供的终极文本渲染方案，可以更清晰的显示文字。
但不建议在一般开发阶段使用，可以在方案定型之后，把出现的汉字成集，然后导进去用。

当前进度

Rect Transform

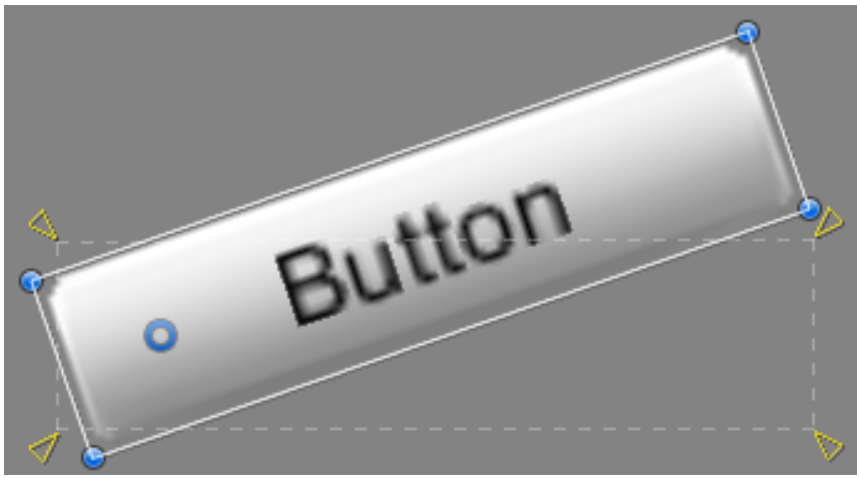
上次课：Transform
由坐标、方向和缩放唯一确定物体姿态。
不同显示器有不同大小，如何确定 UI 部件的相对位置？
Recall: Android: match parent, wrap content.



这里 Unity 提供类似 Android 的 match parent 功能。

锚点与支点

请注意：这里的坐标都是相对于父的 RectTransform 而言的。
和上次的 UV （纹理）一样，都是 0 到 1 之间的值。
默认左下角为原点，右上角为 (1,1)。
因此建议所有的 UI 部件都要塞在画布内。
支点：Pivot，中心点。物件的缩放、旋转围绕支点进行。



蓝点即为支点。

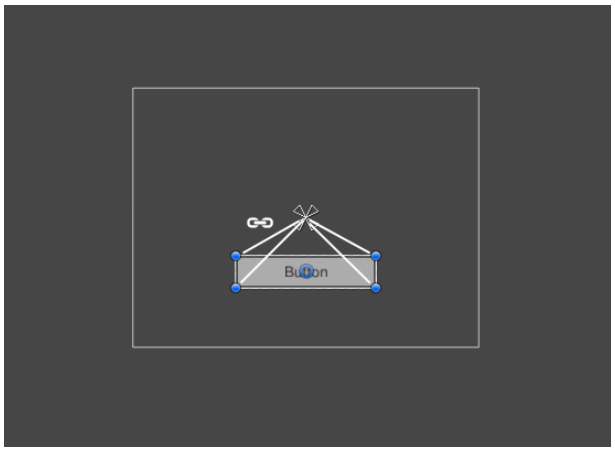
锚点

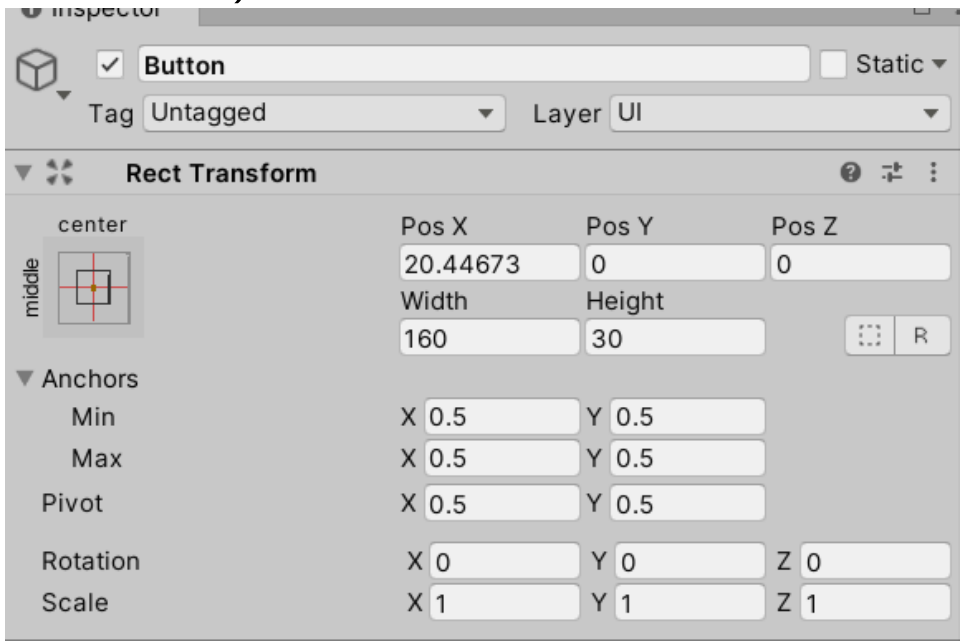
Anchor，又叫固定点。当父部件姿态调整的时候，子物件与锚点的相对位移不变。
实际上，Unity 允许用户定义一组锚点，由其左下角和右上角确定。
按锚点的重合情况，可以用不同方法定义 UI 部件的相对位置。

- 锚点重合，由 (posX, posY, posZ) 和长度、宽度决定。
- 锚点在某个方向重合。
  重合的方向，用位置和长 (宽) 度决定；
  不重合的方向，用与锚点距离决定 (left/right 或 top/bottom)。
- 锚点不重合，就用关于那组锚点形成的矩形，和部件各自矩形边的相对距离决定。

锚点重合的情况

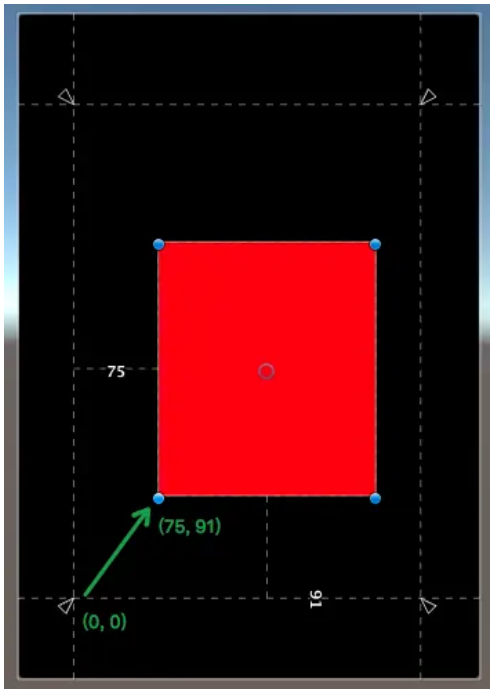
锚点重合，由 (posX, posY, posZ) 和长度、宽度决定。

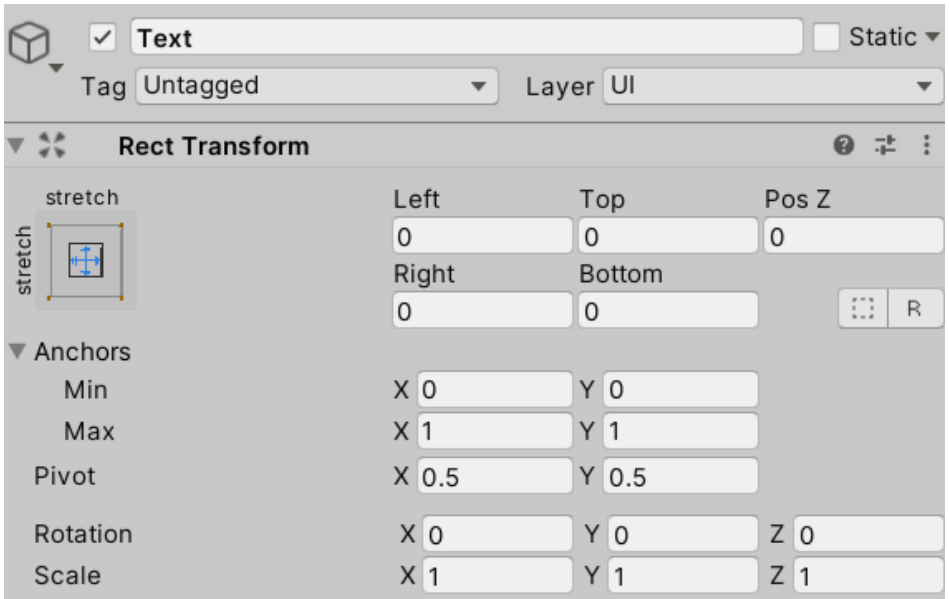




锚点不重合的情况

锚点不重合，就用关于那组锚点形成的矩形，和部件各自矩形边的相对距离决定。





支点和坐标的关系

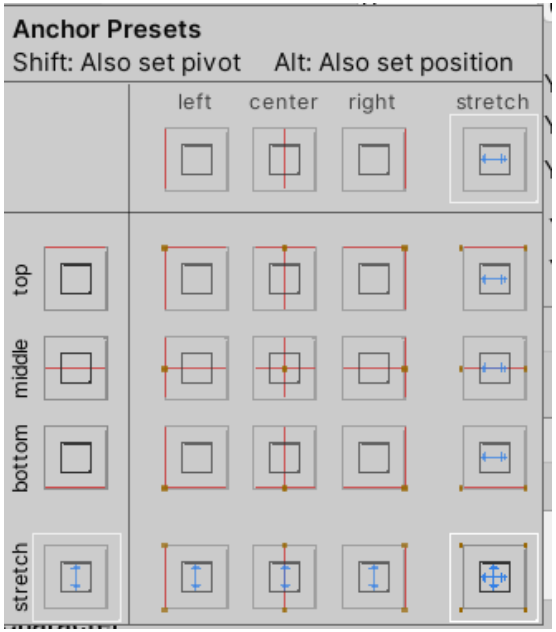
支点也是 0 到 1 的值，当锚点重合，支点决定 (posX, posY, posZ) 在物体上的位置。
例如，支点是 (0.5, 0.5)，边长为 (160,30)，则那个长方形的位置就该是

$$(\mathtt{posX}-80,\mathtt{posX}+80)\times(\mathtt{posY}-15,\mathtt{posY}+15)$$

如果锚点不重合，用 (left/right) 或 (top/bottom) 决定。

锚点预设

锚点的快捷设置
* 自己定义锚点、支点很麻烦，有没有做好的？
利用 Anchor Presets 提供的预置值即可。



当前进度

帧率

视频播放：在极短的时间内，显示多张图片，刷新速度为**帧率**。
类似的，Unity 在一定的时间内，刷新游戏状态，其刷新频率为 Unity 的帧率。
用 `Time.deltaTime` 获取刷新时间间隔。
RECALL: 设某物体有时变速度，其相对零时间位置的位移为

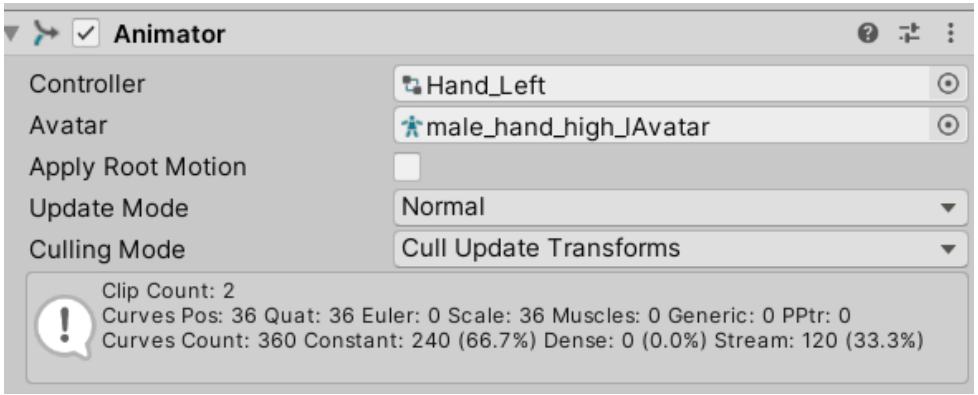
$$x(t) = \int_0^t v(\tau)d\tau \approx \sum_{k=1}^{t/\Delta t} v(k\Delta t)\Delta t$$

可以这样近似

```
float cur_time = 0.0f, cur_x    = 0.0f;
void update() {
    cur_time += Time.deltaTime;
    cur_x += velocity(cur_time) * Time.deltaTime;
}
```

Animator 组件

在场景上添加 Animator 组件，可以看到该组件带有以下字段。

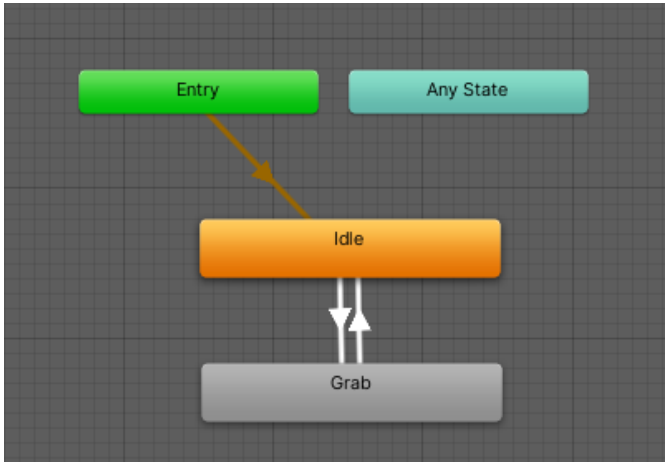


说明:

- Controller: 本质上就是一个状态机，稍后说明。
- Avatar: “人偶”，通常由 FBX 文件定义，可以用 Blender/Maximo 等工具生成。
- Culling Mode（剔除模式）：状态更新方式 (Always Animate, Cull update transform, Cull completely)。Always animate 表示永远播放动画；Cull update transform 表示只有在摄像头瞄准才播放，但仍然更新位置；Cull completely 表示没有瞄准就不更新。

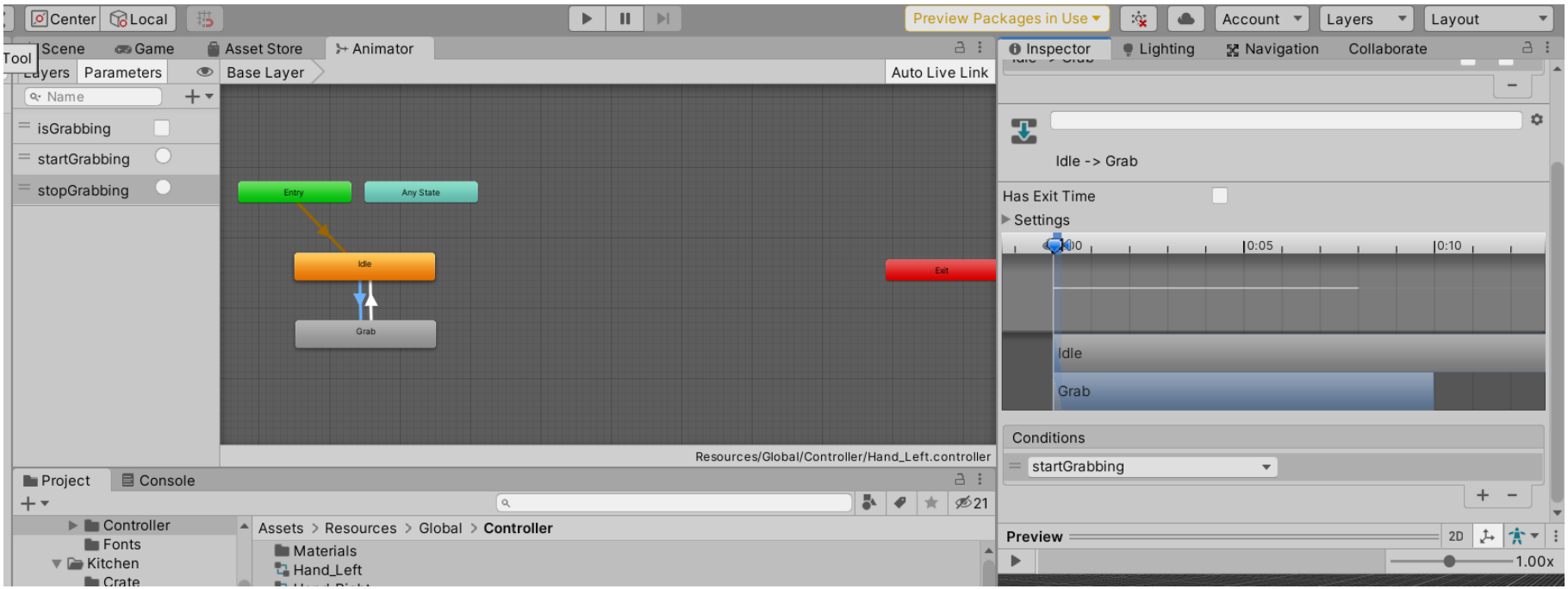
Animator Controller

动画状态机
可以指定不同的状态，处于某种状态下，可以播放动画。



打开 Animator Controller，点击某个状态，在右方 Inspector 下的 Motion，把某个 FBX 文件内嵌的 Animation Clip 拽进去即可。

状态转移



先在左方定义状态机的参数，然后在右下方定义状态转移的触发条件。

状态转移方法

1. 直接切换到某个状态。
   `animator.Play("state\_name");`
2. 更改状态机参数，实现状态转移
   `animator.set{Bool,Float,Trigger}("paarm\_name");`

当前进度

IEnumerator

配合 yield 使用，可以实现一些带有时延的操作。

```
IEnumerator fn(int sec) {
    yield return new WaitForSeconds(sec);
    SceneManager.LoadScene("Home");
}
```

说明：SceneManager 是场景管理工具，一般用 LoadScene 切换场景。

目录

结束