

Lab7. 增强现实

林昭炜 3170105728 数媒1701

1. 实验内容

本次实验分为如下任务和我完成的情况简介

- 使用了 Vuforia SDK, 在 Unity 上进行实验
- 实现了 Image Target 的导入和显示模型
- 实现了简单的交互, 包括了单击、长按、旋转等。
- 实现了Virtual Button
- 实现了一个简单的玩具小车应用, 可以切换小车, 让小车蓄能之后移动。

2. 实现环境

编译环境: Visual Studio 2019, Unity 2019.3.3f1

运行环境: Windows 10, 16GB RAM, i7 7700HQ

Vuforia: 8.5.9


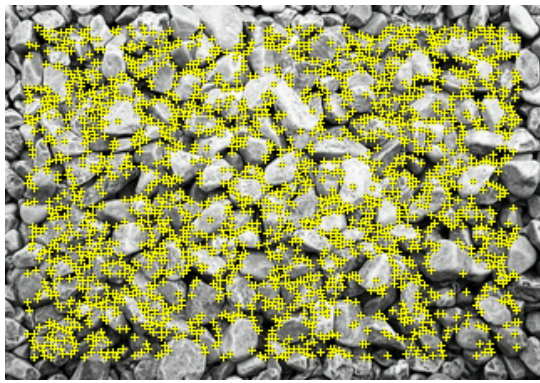
Unity 资源: Project Car - Raider, Low Poly Soldiers Demo

3. 理论基础

以下理论是从 Vuforia 官网¹学习的, 所以使用的图片都是从 Vuforia 官网获取的。

Image Target

Vuforia 很重要的一部分是 Image Target, 它主要依赖的是对图像的特征提取,

原图	提取的特征
	

而要能提取良好的特征需要图片满足如下特性:

- 丰富的细节
- 很好的对比度
- 没有重复的纹理

Vuforia 提取特征的时候依赖的是图像比较尖锐的区域，通常来说:

图像	解释
	正方形的四角都可以被提取为特征
	圆形无法提取特征
	这个物体只有两个尖锐的角可以作为特征

除此之外 Vuforia 对特征点的分布也提出了要求:

1. 数量上必须足够
2. 分布在不同区域上的特征需要尽可能的均衡

4. 实验细节

Image Target

在 Vuforia 中 Image Target 开箱即用, 只要选择 GameObject -> Vuforia Engine -> Camera -> Camera Image Target 即可。

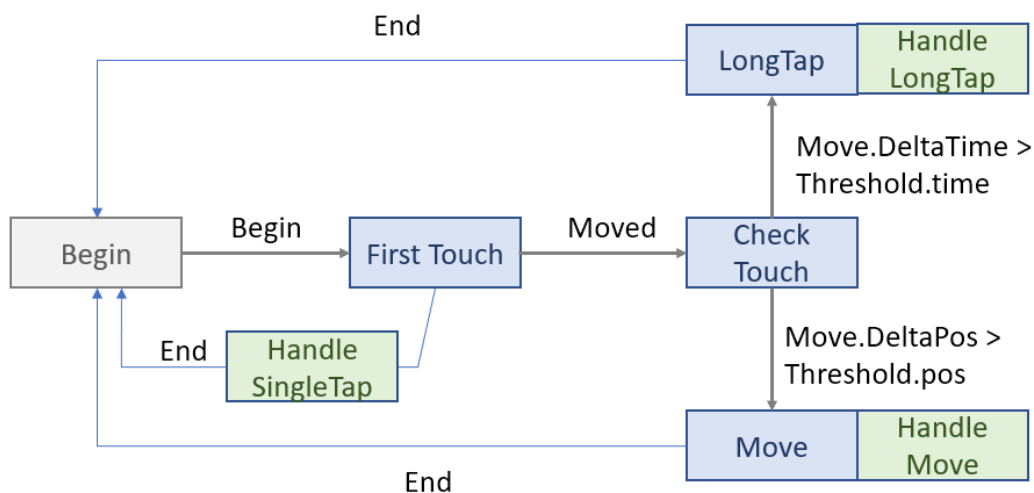
输入管理

在处理输入管理的时候, 可以使用 Touchphase:

```
1  public enum TouchPhase
2  {
3      //
4      // Summary:
5      //     A finger touched the screen.
6      Began = 0,
7      //
8      // Summary:
9      //     A finger moved on the screen.
10     Moved = 1,
11     //
12     // Summary:
13     //     A finger is touching the screen but hasn't moved.
14     Stationary = 2,
15     //
16     // Summary:
17     //     A finger was lifted from the screen. This is the final phase of a touch.
18     Ended = 3,
19     //
20     // Summary:
21     //     The system cancelled tracking for the touch.
22     Canceled = 4
23 }
```

可以直接利用 Touchphase 进行判断, 其实这样子的判断往往是不成功的, 通常总会先落到 Moved 那里, 可能的原因是 Move 敏感度比较高, 所以 Stationary 的检查不一定总能满足。

所以我画出了如下的状态机来区分 单击、长按以及滑动:



实际上 End 的检测使用 Unity 的 API 也无法检测出来，所以也需要用一个 Flag 去标志用户是否点击过屏幕而且这个点击没有触发长按或者移动等操作，如果 Flag 是真的话而且下一次没有检测到任何触摸活动，我们认为单击就完成了。

其中 Threshold 的时间和距离都是要经过实验获取体验比较好的数据，如果 Threshold 值过小，会导致移动误触，响应的，如果 Threshold.time 时间设的果过短也会产生误差。在实验中，Threshold.time = 0.5, Threshold.pos = 0.09.

最终我用输入实现了如下的功能

操作	功能
单击	隐藏或者显示物体
长按	缩放物体
滑动	旋转物体

缩放

在缩放的实现中，我利用了 Unity 的 AnimationCurve 这个功能来实现一个比较平滑的过渡转换效果:

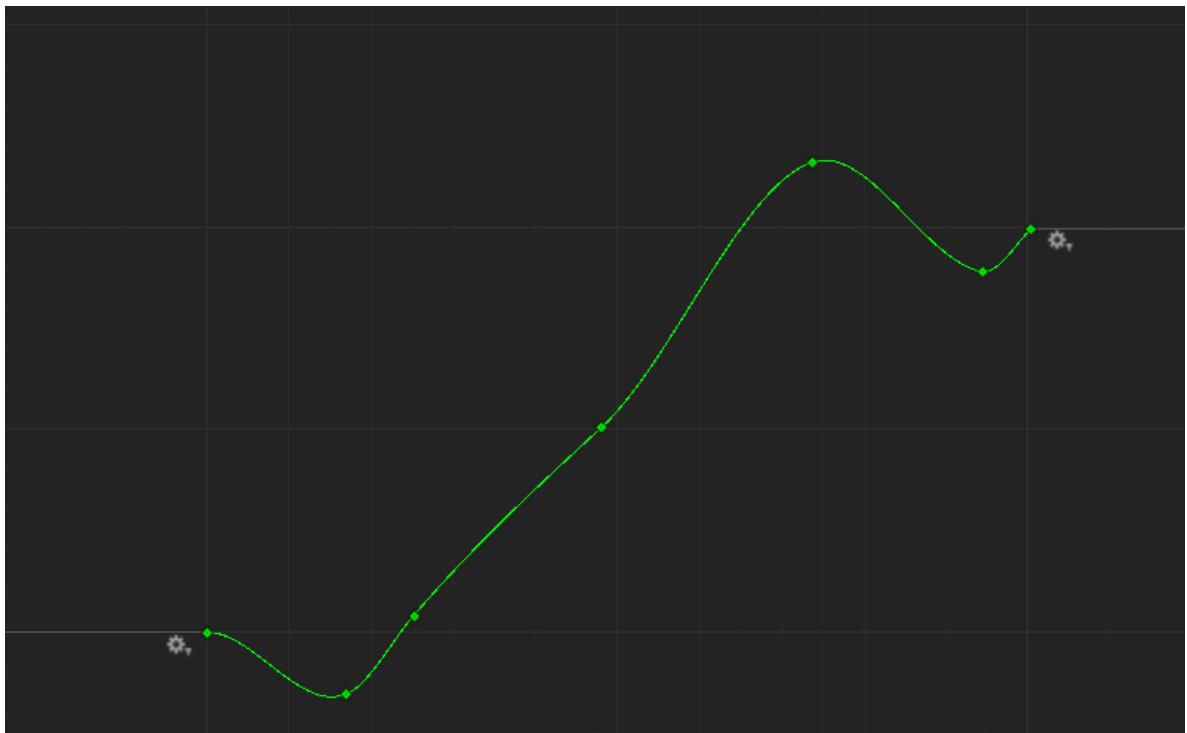
如果我们的采样步长是 0.1 的话，我们可以通过如下代码采样 Animation Curve 曲线

```

1 // 确定采样点
2 scale_cur += 0.05f;
3 // 采样
4 float sample = scale_curve.Evaluate(scale_cur);
5 // 根据采样反应到我们实际需要的scale 大小
6 float cur = scale_begin + (scale_end - scale_begin) * sample;
7 // 最终改变模型的 Scale
8 soldier.transform.localScale = soldier_scale * cur;

```

在实验中我希望有一个比较有趣的缩放过渡，所以我的缩放曲线如下



单击

在单击的实现中我简单地显示和隐藏了物体。

首先是找到物体，一般性来说有两种办法，一种是设置一个 `public` 的 `GameObject`，然后通过 GUI 实现对 `GameObject` 的绑定，还有一种办法是在 Script 里面直接调用 `GameObject.Find()` API 来找到物体。

在实现中我直接调用 `GameObject.Find()`，然后调用 `SetActive` API 来实现对物体显示和隐藏的控制。

滑动

滑动操作的时候我们可以检查滑动的 X 轴的距离，然后根据这个距离决定我们到底要旋转模型多少个角度，这样的话我们需要一个变量记录之前的鼠标位置，然后每一帧如果是滑动的话我们要计算 delta，然后更具这个给 delta 给物体添加转动

Virtual button

Vuforia 支持在 Image Target 上使用 virtual Button, 这样的话用户在触摸实体的时候可以调用响应的脚本。同样的 Virtual button 需要继承 Vuforia 的 `IVirtualButtonEventHandler` 才能使用。

它的使用框架如下

```
1 using Vuforia;
2 public class SwitchCar : MonoBehaviour, IVirtualButtonEventHandler
3 {
4     public int target = 0;
5     public GameObject control;
```

```

6     public GameObject vbutton;
7     // Start is called before the first frame update
8     void Start()
9     {
10         // 注册监听器
11         vbutton.GetComponent<VirtualButtonBehaviour>().RegisterEventHandler(this);
12     }
13     public void OnButtonPressed(VirtualButtonBehaviour vb)
14     {
15         // 在按下时候调用函数
16         control.GetComponent<Swipe>().PickShowing(target);
17     }
18     public void OnButtonReleased(VirtualButtonBehaviour vb)
19     {
20         // Do Nothing
21     }
22 }

```

场景切换

场景的切换只需要调用 Unity 的 API即可:

```

1 using UnityEngine.SceneManagement;
2
3 // ...
4
5 SceneManager.LoadScene(target);
6
7 //...

```

小车玩具

小车的交互一共分为两个部分，一个是蓄能的阶段，一个是释放的阶段，首先蓄能的阶段比较简单，我们只要提取出 x 方向的平移距离，然后相应地移动小车即可。

小车在能量释放阶段，我们可以缓慢地释放能量来获得速度:

```

1 speed += energy * 0.3f;
2 energy *= 0.7f;

```

然后利用速度更新小车的位置:

```

1 car.transform.Translate(0, -speed * Time.deltaTime / 10000.0f, 0);

```

最后我们根据摩擦力去减小速度，在这里我用一个简单公式,让小车的速度线性减小:

```

1 speed = Mathf.Sign(speed) * Mathf.Abs((Mathf.Abs(speed) - 0.05f) * 0.95f);

```

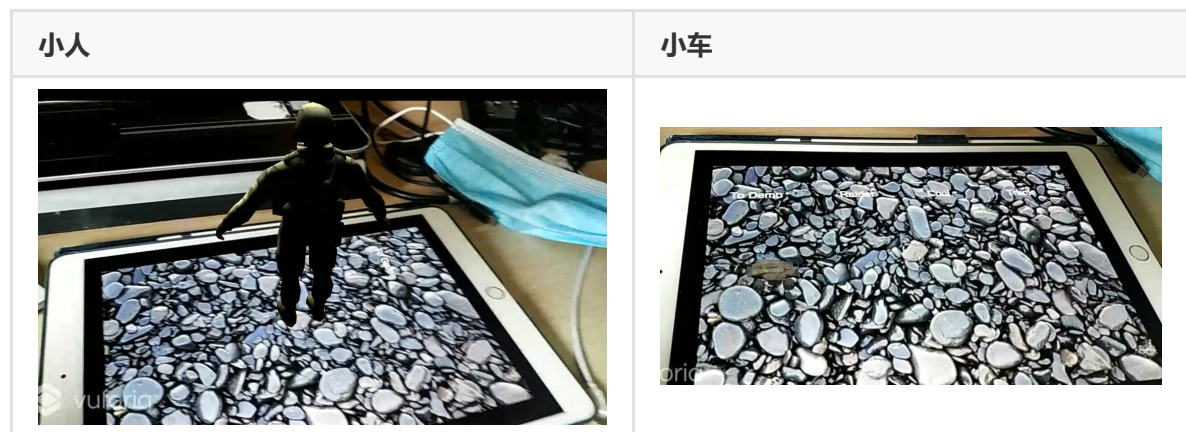
为了保证速度最终是收敛到0的，所以要加一个判断条件：

```
1  if(Mathf.Abs(speed) < 0.025f)
2  {
3      speed = 0f;
4  }
```

最后就能实现玩具小车的玩法。

5. 结果展示与分析

结果展示



分析

对于2D Image Tracking类应用，可以尝试不同的Image Target图片，多种可视角度，分析在哪些情况下跟踪容易丢失。

从官方文档来看，其是通过特征点去追踪Image Target. 我结合我的实验和官方文档人为在如下情况可能更丢：

1. 快速移动摄像机，我认为这和光流很像，因为光流的基本假设之一是运动非常小，如果运动比较大的情况下，不符合光流假设，就没法快速跟踪运动，可能需要完全重新分析，所以快速移动之后需要一点时间才能定位
2. 纹理过于简单，我自己画了一个笑脸作为 Target Image，因为笔画比较简单，能够提取的特征数有限，无法得到很好的效果。
3. 纹理重复，比如布料的纹理，虽然有很多特征点，但是提取到的纹理特征都是重复的，所以 Vuforia 没办法判断具体是布料的哪一个部位。

6. 编译运行

编译说明

在 Package Manager 里安装 Vuforia，Unity设置安卓编译。

运行说明

APK 安装即可。

1. <https://library.vuforia.com/content/vuforia-library/en/features/images/image-targets/best-practices-for-designing-and-developing-image-based-targets.html> [↗](#)