技术报告

Cubo

一、综述

本游戏制作中使用到的主要工具是Unity3D和PhotoShop，由四位成员共同完成制作。四位成员的职责划分明确，余秋滨负责游戏原型设计与实现、教程制作、游戏逻辑代码实现等，王一能负责游戏关卡的设计与实现、游戏模型设计等，章海威负责音乐音效实现、UI设计等，刘耕铭负责美工设计与实现、动画制作等。王一能的技术报告放在《关卡设计报告》中，其他人将在后文分别陈述各自的技术实现。

二、游戏原型设计与实现

在制作游戏原型的时候，首先遇到的困难是对Unity3D的不熟悉。因此花了一段时间学习Youtube上的Unity官网教程。然后就开始搭建原型。我们的游戏中最核心的就是方块，因此要使用到Unity3D的3D模块中的Cube模块。

由于我们的大方块是由许多小方块组成的，因此应当将许多小方块的集合视作一个整体来进行旋转等操作。因此需要在hierarchy中设置一个空的GameObject，在其上绑定一个C sharp脚本文件CubeController.cs。然后创建一个Prefeb，模型就是Cube，在其上绑定一个脚本文件SingleCube.cs。然后在CubeController.cs中用Instantiate方法动态复制Prefeb内的Cube，并设置复制出来的Cube的Parent为这个空的GameObject。于是就建立了一个“小的个体”与“大的整体”之间的联系。在这个过程中，一个很重要的

问题就是在动态创建每一个小方块时，要计算好其世界坐标与相对坐标，为今后的运算做准备。

原型中还需要能接受玩家的输入并进行相应的反馈。首先需要在PlayerSetting里面绑定键盘鼠标映射到对应的键位，然后在脚本中可以调用相应的方法来监测输入。我们的游戏中玩家需要对方块进行旋转，无论是用键盘或者是鼠标。因此我们需要对刚才做出来的“大方块”整体进行操作，调用RotateAround方法进行旋转。需要注意的是我们最好设置一个Public变量来方便改动旋转速度，另外还要乘以Time.deltaTime变量来将动画变得更平滑。

其次还需要对场景进行一些简单的布置，诸如设置平面、设置围墙等，一个游戏原型就完成了。

三、游戏逻辑控制

在游戏逻辑设计之前首先要解决脚本间通信问题，我使用了两种方法进行通信。第一种方法是sendMessage,通过找GameObject的名字或者Tag，然后找到对应绑定在其之上的脚本，调用对应设置好的方法，来传递信息。第二种方法是对文件进行读写来共享数据。

在关卡选择中，按下不同的按钮需要进入不同的游戏关卡。由于我们的游戏内涵主要体现在不同的方块之间，场景可以不需要更换。因此我选择动态加载模型。在关卡选择按钮按下后，会使用PlayerPrefs进行数据写入（以key-value形式存储），然后进入游戏界面中通过key调出value，确定应该加载哪一关的模型。

加载模型后即开始游戏，根据游戏规则玩家可以选择消除方块。那么在玩家进行操作以后，对应的小方块不会马上消除，而是通过sendMessage给“大方块”，并传输创建时就保存了的相对位置，由大方块中的控制代码决定这个小方块是否消除正确，

并再次通过sendMessage的方式传递消息回去。这里要注意的是，由于小方块是clone创建的，因此不能通过寻找名字的方法来找到gameObject，而必须手动设置Tag。同时小方块中也要设置相应的flag变量来决定是否响应来自大方块的信息，以减少不必要的操作。

如果小方块决定被消除，那么就会调用setActive(false)来消除。这里又牵涉到音效处理的问题，由于Unity中播放声音需要音源，并且GameObject被inactive以后音源将随之一起inactive，也就是说无法发出声音，并且这一点与代码执行顺序无关。因此诸如消除方块正确、消除方块错误之类的音效不能使用小方块的音源，而是要通过寻找parent找到父音源来调用。

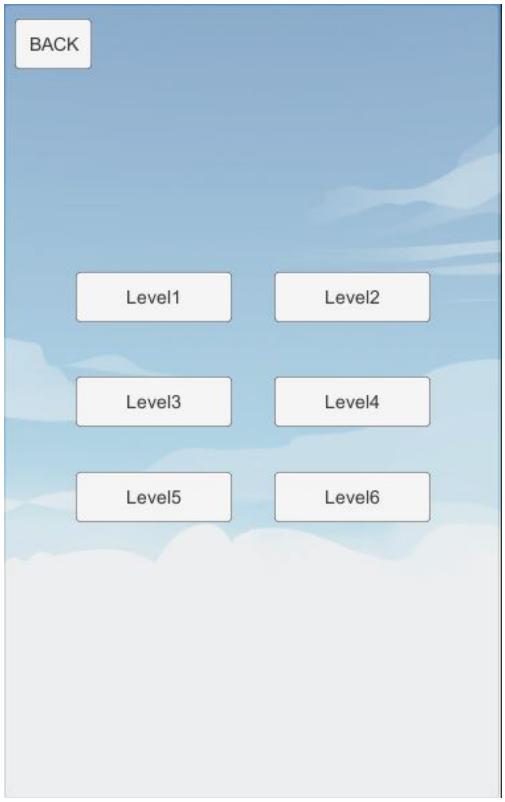
另外游戏中还需要处理许多逻辑运算问题，不过这都是代码实现问题，与Unity本身并没有直接关系，在此就不说明了。

四、UI设计

由于游戏是采用Unity3d开发的，因此主要用到了Unity3d中的API。我主要负责游戏界面的布局，游戏场景的切换，以及游戏音乐的添加等部分的开发。涉及到Unity3d 中的UI模块和Audio模块，用到了UnityEngine中的Application，AudioSource，Mathf等类以及UnityEngine.UI中的Button, Resources, slider, Toggle 等类。

游戏主要界面有5个，分别是主界面，教程界面，设置界面，关卡选择界面以及游戏界面。我设计了游戏的主界面，关卡选择界面和设置界面。设计的主要理念是追求简洁，使玩家能清晰地理解游戏的逻辑，迅速地掌握游戏的玩法，从而快速地投入

到游戏中。接下来对各个界面进行分析（以下截图是UI设计时的截图，不是经过美工后的最终游戏截图）。



上面左图是游戏主界面，右图是游戏的关卡选择界面。主界面中有4个Button控件，点击按钮可以分别跳转到教程界面，关卡选择界面，设置界面或者退出游戏。关卡选择界面由一个返回按钮和若干个关卡按钮组成。点击返回按钮可返回到主界面，点击关卡按钮可进入相应的关卡并开始游戏。

右图是设置界面。游戏界面可进行游戏相关环境的设置，比如音效等。设置界面采用滑动条和复选框等控件，从而方便玩家直观地进行设置。

为了实现场景的切换，我通过调用Button中的OnClick() 函数，将按钮点击事件与相应的函数绑定，然后调用Application.LoadLevel() 函数，这个函数会销毁当

前的场景并加载指定的场景，从而实现场景的跳转。

五、音乐音效设计

音乐是游戏中不可缺少的元素，能起到很好地烘托游戏气氛的作用。游戏中使用了三段不同的背景音乐，分别用于不同的场景，相比于只使用一段背景音乐可以使玩家不容易产生审美疲劳。为了与本游戏的特点相符，采用都是比较欢快活泼的音乐。

在Unity3d 中，声音都是通过Audio模块来管理的。

要在游戏场景中播放背景音乐，需要在场景的任意对象上绑定一个AudioSoure组件来管理音频片段。但是，这样有一个问题，就是当场景切换后，绑定的对象会被销毁，导致背景音乐无法连续播放。为了实现背景音乐在不同场景中连续播放，我使用了DontDestroyOnLoad(this.gameObject) 函数来防止对象在场景切换时被销毁。

而要在不同的场景中播放不同的音乐，需要使用Application.loadedLevel 来判断当前处于哪个场景，然后通过设置AudioSource.clip 来指定要播放的音乐片段。

在设置界面中，可以通过滑块来控制背景音乐的音量。这个功能是通过将AudioSource的volumn 值设置成滑块的值来实现的。滑块(slider)是UI中的一个控件，每当滑块的值被改变时都会触发value changed 事件，将该事件绑定到指定的函数，就可以在其中实现相应的功能。

六、美术设计

我做的第一件事就是修改鼠标样式。在网上找到了一个和我们游戏主题风格比较相符的图案，并参考了一些文档，将游戏中的鼠标样式修改了一下，结果发现效果非常不好。仔细分析过后得出结论：原图的分辨率大约是 1024*1024，而鼠标样式的大小大约是 50*50，再加之这张图的色彩比较丰富，在如此高比例的缩放下很容易出现边缘的模糊。思考过后我利用 PS 将图前面的脸部消去，并且将色彩调为灰色，这样一来再缩放就和谐很多了。

原图

压缩后

锤下的状态

之后试了一下 Maya，找了一些素材，改了下 Skybox，总体来说效果都不太满意，不过还是挑选了相对来说还算看着顺眼的素材，简单美化了一下游戏界面。在这里还遇到过一个问题，我利用某个 material 给方块上色的时候发现游戏启动后会变得非常卡，这大概是因为方块的数量太多导致的吧。

再之后我主要完成了主界面的美化，首先组长@余秋滨找到了一张比较合适的图片，交给我后首先将它原来本身的一些文字 PS 处理掉，然后挑选合适的字体，利用 PPT 的艺术字来制作 LOGO。我们的游戏名是 Cubo，这里我用一个自转的方块来替代了最后一个 o，感觉效果还不错。方块自转是利用了 transform.RotateAround 和 Time.deltaTime 实现。

再之后需要处理点击事件，这个开始的时候一直觉得很困难，以为只能利用 Button 对象才能够实现点击事件，后来根据自己对结构理解发现 Button 不过是一个属性，于是尝试给 Text 添加 Button 属性成功添加了点击事件，那这样一来就可以设计出更酷炫的样式了。在字体库中挑选了一个还算有个性的字体，复制到资源文件夹，会自动导入工程，接下来就可以使用了。

最后根据这个背景/风格修改了关卡选择和设置界面，完工。