# 开发环境的搭建

## 2.1 Unity游戏引擎安装

2019版本的Unity游戏引擎官方推荐通过Unity Hub来管理安装。通过Unity Hub可以下载安装任何版本的Unity游戏引擎或者相关组件，还可以管理磁盘中所有不同版本的Unity游戏引擎以及所有Unity工程项目。去Unity官网下载安装Unity Hub，再启动Unity Hub并登录Unity账号去安装最新稳定版本的Unity游戏引擎，相关组件勾选Android Build Support以及中文语言支持（VS Community也可勾选），之后Unity Hub就会自动后台下载并安装。安装完，打开Unity Hub在Projects选项卡下点击new就可以新建工程。

## 2.2 Mono开发框架的安装

因为Unity引擎的多平台发布依赖于Mono对于.net框架的可跨平台的实现，并且Mono中还包含了C#的编译器，所以Mono也是必须要安装的。Mono的安装可直接在Mono官网下载安装即可。

为何Unity能够跨平台发布原生APP？ 其原理就在于Mono重新实现了.net标准中叫做CIL（Common Intermediate Language，微软通用中间语言）的一种代码指令集的编译与反编译。CLI可以在任何支持CLI（Common Language Infrastructure，通用语言基础结构）的环境中运行。这样跨平台只需将代码编译成CIL，然后再在各个平台运行时，用各个平台的CLI去解释运行CIL或者将其编译为该平台下的原生代码，而Mono就是实现了不同平台下的CLI。

## 2.3 代码编辑器VS Code安装与配置

代码编辑器我选用的是Visual Studio Code，因为其比较轻量，占用内存比较小，开启关闭速度快。当然也可以用Visual Studio，VS能够提供更完整的代码提示补全机制。VS Code可直接在官网下载安装，安装完成后还需去安装Debugger for Unity插件用于Unity运行调试，以及C#支持插件用于让编辑器能识别C#代码从而可以实现多个脚本代码关联、查找引用、代码格式规范、代码提示补全等功能。因为该C#支持插件是基于OminSharp的并且在windows下依赖于.net Framework，所以还需安装.net Framework Dev Pack，并且不能安装最新的4.8版本（因为c#插件不支持4.8版本，4.7版本为宜）。

## 2.4 JDK的安装

因为该项目最终要发布到Android平台，所以必须要安装JDK。JDK直接在官网下载安装即可。安装后还需配置环境变量，以windows为例：

在系统变量ixa新建一个名为JAVA\_HOME的变量，变量值为jdk的安装目录。（用于指定jdk的安装路径，很多Java程序会用到这个变量，接下来的PATH和CLASSPATH的变量也会使用到该变量）

在系统变量下查看PATH变量，如果没有就新建一个。点击编辑，在变量值文本框的末尾添加上“%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;”。（Path指定了一个路径列表，当执行一个可执行文件时，如果在当前路径下不能找到这个文件，就会依次安装Path下的路径去一一查找，直到找到为止，否则就会报错。因为Java的编译命令（javac)，执行命令（java）和一些工具命令（javadoc，jdb等）都在其安装目录的bin下以及其安装目录下的jre的bin下，因此将这两个路径添加到PATH变量中）

在系统变量下查看CLASSPATH变量，如果没有同样新建一个。点击编辑，在变量值文本框的起始位置添加上“.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;”（CLASSPATH也是指定了一个路径列表，是用来搜索Java编译时需要用到的类。最前面的“.”代表了当前目录，包含了该目录后，就可以到任意目录下去执行需要用到该目录下某个类的 Java 程序，即便该路径并未包含在 CLASSPATH 中。

## 2.5 Android Studio的安装

因为最新版本的Unity引擎构建Android apk舍弃了原先的内置构建，转用了Android Studiod的gradle构建，同时发布到Android平台还需要用到Android SDK，所以要去安装Android Studio。Android Studio在官网上下载安装。在安装过程中，设置安装路径不应带有中文字符以及空格。安装完成后，紧接着会提示安装Android SDK，我因为自己手机是Android 6.0的缘故选择的是API 21，高版本的API会导致无法发布到低版本的Android手机上。安装完成后，需要新建一个项目并打开，这时右下角会显示正在下载安装gradle，gradle下载安装完，并且空项目构建完就可以关闭Android Studio了。

## 2.6 Unity Android Build Support 配置

Unity还需设置发布平台，打开Unity创建完工程项目后，选择左上角File->Build Settings->Android->Switch Platform即可切换发布平台到Android。此外，接下来点击android下player settings，在弹出来的Project Settings的窗口里选择Player选项卡，设置好Company Name、Product Name、Default Icon后，打开下面的Resolution and Presentation选项卡设置Default Orientation为Landscape Left（左侧横屏）。这样发布到Android平台的所有准备就全都做好了。

## 2.7 Source Tree的安装

这个游戏项目是交给GitHub管理的。Sourcetree是一款GitHub代码管理工具，有了这个工具可以不用敲Git指令很方便地实现创建、克隆、提交、push、pull、合并等操作。需要先去官网下载安装git，然后再去下载安装Sourcetree，安装过程中还需登录BitBucket账户。安装完成后，还需给本机提交权限。首先要去生成本机的ssh密钥，通过在cmd中输入指令“ssh-keygen -t rsa -C ["youremail@example.com"](mailto:\"youremail@example.com\") ”来完成，之后登录GitHub，打开setting->SSH keys，点击右上角 New SSH key，把生成好的公钥id\_rsa.pub放进 key输入框中，最后在SourceTree中配置好SSH密钥和SSH客户端(SSH客户端选择OpenSSH)即可。

# 总体设计

## 3.1 人物设计

### 3.11 人物形象设计

人物的形象由几张身体、手、脚、眼睛表示人物身体部分的sprite图片拼凑而成。这样，可以通过改变人物身体各个部位的位置、旋转角度、缩放大小，让人物做出不同的姿势、动作。

### 3.12 人物动作设计

人物的动作使用Unity的动画编辑器来完成。相比较于在人物脚本代码里通过FixedUpdate去每帧切换人物的Sprite图片的方式，这样可以不再依赖于帧动画图片素材。Unity的动画编辑器提供了一个帧时间轴，我们只需给场景中结点在这个时间轴的某个帧上定义关键帧即可，当播放这个动画时，Unity会自动计算出从一个关键帧到另一个关键帧的过渡帧。

通过Unity的动画编辑系统，需要制作出人物的若干个独立动作：默认动作、开始奔跑、奔跑中、刹车、跳跃、死亡。人物不同动作之间切换还需要做过渡，这个就依赖于Unity提供的动画融合功能，通过设定一些参数就实现同一个物体的不同动画间的流畅转换（这个在实现里会详细阐述）。此外，这些人物动作的切换还需脚本代码控制，Unity的动画控制系统会提供设置动作转换的条件参数，我们需要在脚本代码里设定这些参数，来起到控制切换的作用。

### 3.13 人物运动控制设计

人物运动的实现有两种方式，一个是在脚本里每帧根据玩家的按键去刷新人物位置，另一种是通过Unity物理引擎来实现。我选择是后者，因为物理引擎能够提供更加丰富多彩的人物运动表现，比如人物奔跑起来后不能立马停下的加速度、角速度引起的翻滚、重力的加持影响等等，这些都只要调用人物的刚体组件提供的物理引擎相关的方法即可。

### 3.14 人物影子设计

这个游戏还有个人物影子的设定，用于复现上一次死亡前玩家操作的人物的行为，用于提醒玩家上一把遇到的坑以及增加游戏趣味性。这个通过每帧去记录人物的帧数据，然后在下次重生时将这些帧数据应用到人物影子上播放出来即可。

## 3.2 游戏场景设计

## 3.2.1 场景的视图布置

场景的视图主要考虑到背景、人物、活动地面、地面装饰。我们将背景的sprite图片层级设为最低，循环铺设在场景的最后面，人物各个部位的sprite层级设为最高来避免遮挡，然后就可以根据设计依次铺上地、水以及树、花等地面装饰来完成一个美观关卡视图。

### 3.2.2 人物活动地面

地面是场景中支持人物运动的关键所在，所有地面必须绑定上2D碰撞器组件来支撑人物，除此之外，不同坡度的地面决定了人物站在这块地面上运动时所施加的力的方向和大小，所以还需绑定一个地面脚本来修改人物移动时所施加的力。

### 3.2.3 场景限制区域

场景不是无限大的，所以为避免人物走到非法区域，所以左右两边需要设置空气墙，底下需要设置一个死亡线，当人物落到这个死亡线时会置人物死亡并重生。右边的空气墙还有触发切换到下一关的作用，当人物碰到右边空气墙时，就会触发碰撞回调调用切换场景的方法。

### 3.2.4 场景切换

场景的切换使用异步加载的方式，这样可以让两个关卡切换衔接起来更加自然流畅，当触发切换关卡时，原场景会还在，并且会慢慢变黑，此时第二个场景在后台加载，加载完成后才将黑色遮罩慢慢变透明。

## 3.3 摄像机设计

因为是2D游戏，所以摄像机选用正交摄像机，它能忽视场景物体的远近，呈现出2D画面。

### 3.3.1 摄像机跟随

摄像机需要去跟随人物移动，那么只需在每帧里去调整摄像机到相对于人物的一个固定位置即可。为了实现美观，我在每帧中将摄像机调整到刚好能使人物在屏幕中处于黄金比例的一个位置上。

除此之外，为避免摄像机照摄到非法区域而导致穿帮，摄像机的移动还需设定区域限制，摄像机的区域限定需要通过接下来要介绍摄像机动态的正交大小来计算出具体值。

### 3.3.2 正交摄像机的手机适配

为了能使游戏场景在各个手机上都能按照合适的比例显示出来，在摄像机的脚本里需要根据当前手机屏幕的分辨率去动态调整摄像机的正交大小，并根据调整好的摄像机正交大小去计算出摄像机的移动左右区域边界以及人物跟随时相对与人物的位置。

## 3.4 陷阱、怪物、道具设计

### 3.4.1 陷阱、怪物触发人物死亡

触发人物死亡都写在对应陷阱、怪物脚本中的碰撞、触发回调中，当人物碰到陷阱、怪物的碰撞器时就会去调用人物脚本提供的死亡方法。人物的死亡动画的末尾绑定一个复活的回调方法，当人物死亡动画播放完就会自动去调用复活方法。

### 3.4.2 人物复活时陷阱、怪物的重置

人物复活时，场景中所有的物体都要重置回原先的状态，所以所有的陷阱、怪物的脚本都要提供一个reset方法去恢复状态，reset方法会在初始化的一开始利用c#的事件与委托机制绑定在人物脚本的一个事件上，而这个事件会在人物复活方法中被调用，这样就实现了人物重生时的陷阱、怪物重置。

### 3.4.3 陷阱、怪物的触发唤醒设计

对于需要触发的陷阱、怪物，可以利用双重碰撞器来实现，最里面层的碰撞器包裹陷阱、怪物体积用来代表实体，来触发人物碰到死亡的回调，之后再嵌套一层碰撞器勾选上trigger用于触发调用陷阱、怪物唤醒的方法。这样做的好处是，可以在可视化场景编辑器中直观地配置陷阱、怪物的触发半径。

### 3.4.4 陷阱的寻路方式

陷阱的寻路，可以为陷阱配置一个坐标点队列与移动速度，然后让陷阱根据配置的速度依次沿着这个队列中点去移动，这样就能很简单地去规划陷阱的移动路径。地面也可以用这个方法，实现移动载人的效果。

### 3.4.5 怪物的跟踪方式

怪物的跟踪取决于人物距怪物的方位，因为这是一个2D横板游戏，所以可以将方位简化为只有左和右，通过每一帧去判断人物距怪物的左侧还是右侧从而让怪物往相应的位置移动，一旦人物进入怪物的攻击范围，怪物就会立即发动攻击。此外，怪物的跟踪不能是全图的，所以还需设定一个跟踪范围，或者当人物一旦脱离怪物的检测范围，怪物就会返回原先的导航寻路点。

### 3.4.6 射击型怪物的射击方式

射击型怪物子弹的射击通过物理引擎来实现，通过给子弹刚体以冲量而将子弹射击出去，射击的角度由当前帧怪物与人物间的方向向量来决定，子弹在生成好后需要设定多少时间后销毁，防止其一直占用内存。怪物的射击可以先调用射击动画，在射击动画的末尾绑定射击方法的回调，这样可以实现怪物将射击动作做完后才将子弹射击出去。

### 3.4.7 下落型陷阱

下落型陷阱可以通过在触发回调里改变陷阱刚体所受重力倍数来实现，坍塌的地面、高空落刺都是这样实现的。

### 3.4.8 加速、减速陷阱

加速、减速陷阱通过在人物触发陷阱时，每帧里去将人物刚体的速度乘以倍率来实现，在人物离开陷阱范围后恢复人物速度来实现。

### 3.4.9 中途存档点

当人物触碰到中途存档点时，触发trigger回调，在那里面会重设人物的复活位置以此来实现中途存档点的功能。

## 3.5 UI界面设计

### 3.5.1 人物控制的虚拟按键

用于代替键盘控制人物左右移动、跳跃，为了能有按下反馈，需要在PointerDown、PointerUp、PointerExit事件绑定的回调中去切换它的图片。（具体如何控制人物移动会在详细实现中说明）

### 3.5.2 暂停键

用于暂停游戏，通过修改Unity的系统时间为0来实现。当点击暂停后，会产生一层灰色遮罩，暂停键也会变为红色，当再次点击暂停键时，就会继续游戏。

### 3.5.3 上、下一关

用于切换关卡，点击后会切换到上、下一关，方便调试关卡。

### 3.5.4 开始页面

游戏的启动页面，设计为游戏标题的浮动，以及点击任意处标题飞出，人物跑着出现并开始游戏。这些通过动画编辑器以及脚本里绑定点击事件triggerEvent回调来实现。

# 详细设计与实现

## 4.1 人物动作与移动控制