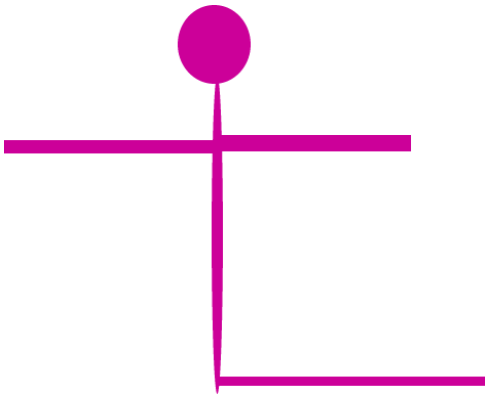
Flash 游戏项目

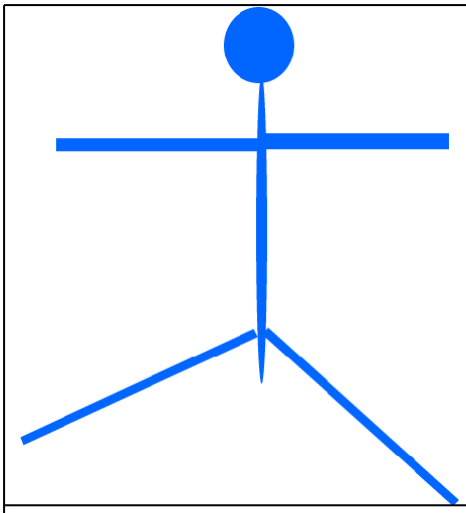
----------——《XXXXXXXXX》

故事背景：

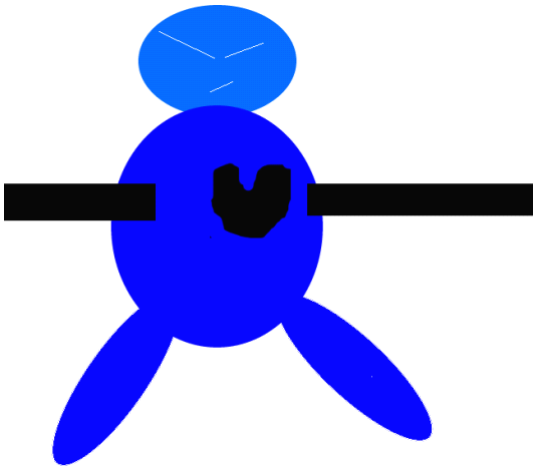
500 年前，一个村庄里面住了一个很平凡的人，这个人叫平凡。 有一次，平凡在外旅途的时候见到一个村民被欺负，平凡出手相助了这个村民，但是恶龙不服气，心里面要挟村庄里面的村民。这时候，平凡为了拯救这村庄跟恶龙进行激烈的打斗。但是在打斗的时候，被一个村民人暗算了。这个村民叫背叛，平凡和恶龙进行了 500 回合的打斗，最后牺牲了。恶龙消失了，人们为了感谢纪念这个人，作了一个纪念碑叫勇士，勇士一直收获村庄。
但是恶龙从黑暗中醒来了，村庄也遭受到恶龙的继续破坏，这时候有一群自发组织的人，叫勇者，他们无谓恶龙的邪恶，和这恶龙进行一场激烈的打斗。

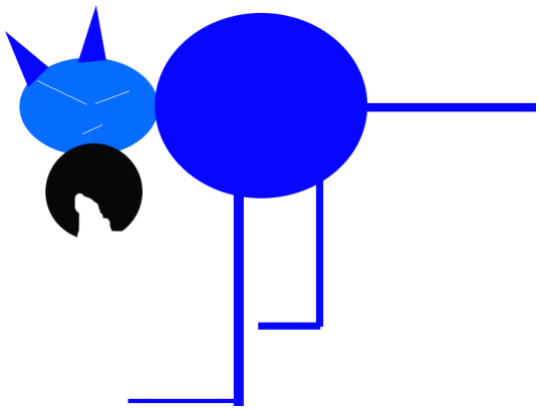
主角群：

主角	属性
	勇士： 技能：左踢
	勇士： 技能：右踢

	勇士： 技能：一起踢

怪物群：

怪物	
	黑心的邪恶老板，特点黑色的心，黑色的容貌，黑色的脸， 武器：黑心刀 技能：杀无赦

	丑陋的黑心猫：黑心老板的打手 特点，黑色的宿根，邪恶的容貌。 武器：黑心一激 技能：啥无邪

开发项目类型：一款基于多人同时玩的在线大型 Flash 游戏

开发平台：

Flex builder 3。

基本系统：

消息处理系统、场景显示及行走系统、打斗系统三大主要部分。其中又以消息处理系统为核心模块，其余部分紧紧围绕它运行。

消息系统：处理游戏中不同的状态，游戏根据不同的状态作出不同判断。
场景显示：
行走系统：
打斗系统：打怪物的系统

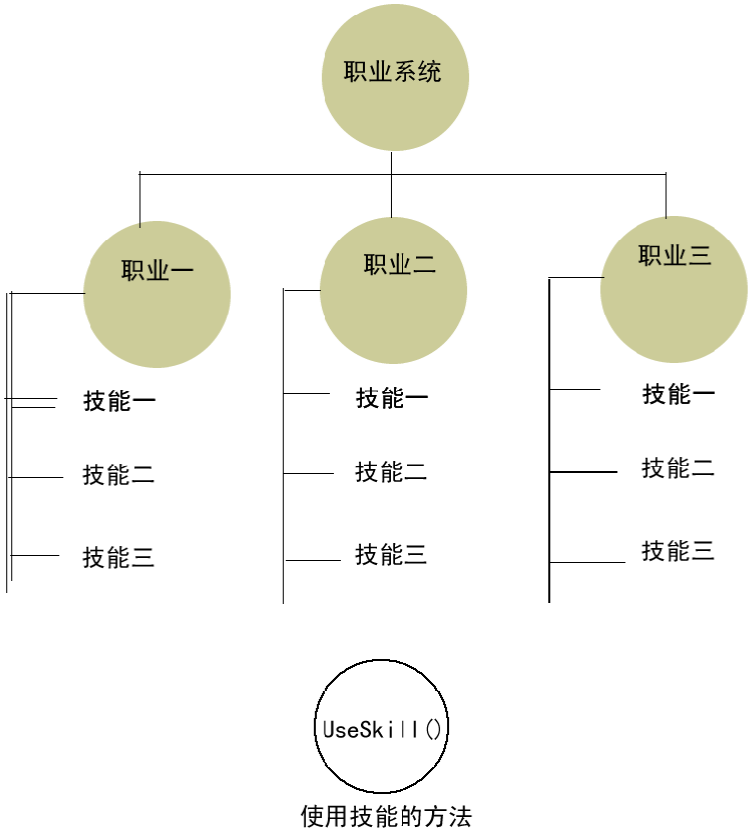
其他系统：

职业系统，操作系统，世界系统，角色系统，Ai 系统；

职业系统：游戏中有不同的职业选择，会分派不同的技能。这些时候需要我们去考虑到职业系统。

操作系统: 游戏中玩家的操作流程。是玩家与用户交互的一种手段

世界系统：讲述游戏文化内涵
角色系统：玩家操作游戏人物进行游戏的人物；
Ai 系统：游戏中的大脑中心；是宏观控制每一个角色进行活动的上帝；



每一种技能都有一种使用的方法

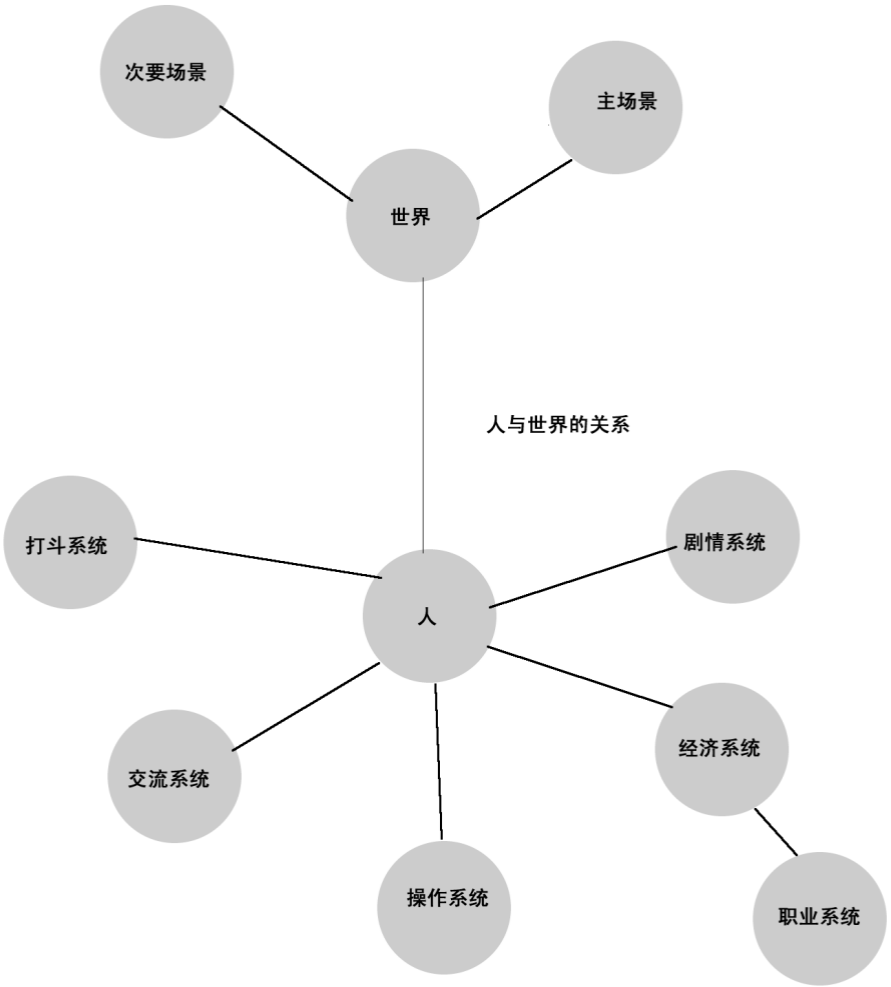
```
class Skllone
{


   UseSkill()

}
```

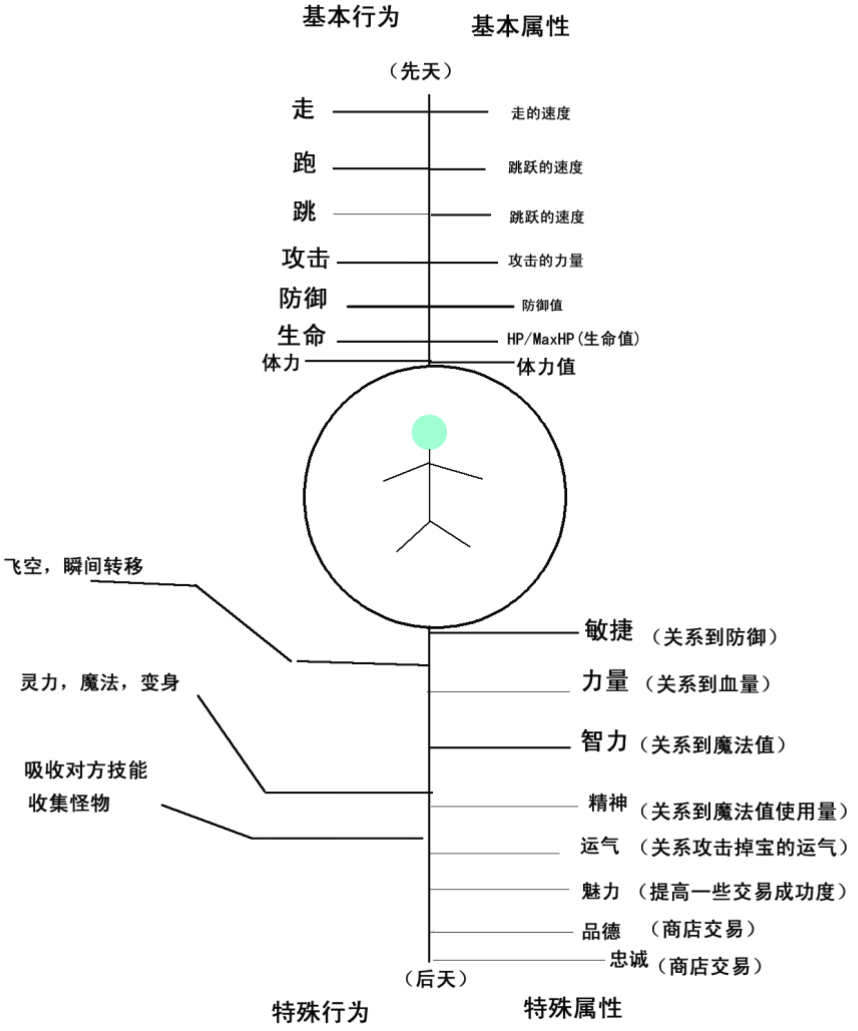
职业系统与能力

世界系统：



角色系统

说明：（一个人出生后，不经过学习就有的属性，一个人学习后产生的一些属性）



游戏主类（GameMain 类）：

GameMain 是游戏主入口：

1.初始化窗口：设置窗口风格
2.进入系统消息循环

```
WndProgress（message：int）
{
   Switch（messgae）
  {
   case WM_Init:
         //初始化窗口   ；
      break
   case WM_Paint:
          //渲染窗口
```

```
        break;
    case WM_Close:
        //退出系统
       break;

    case WM_KEY_DOWN:
        //键盘事件
        break;
  }

  Return 0;

}
```

项目开发主要类和类库包：

游戏主类（GameMain 类）：是游戏主程序入口

地图管理(MapManager 包)：用于加载和卸载地图等

物品管理（ItemManager 包）:用于管理道具物品，武器，套装等

声音管理（SoundManager 包）: 用于管理声音，包括加载音乐，设置音乐特效等

角色管理（CharacterManager 包): 包括常见用于创建主角和配角

事件管理（EventManager 包）:每一个情节作为一个类进行管理

对白管理（WordManager 包）:用于管理语言，读取一些基本通知信息，npc 说话

场景管理（SenceManager 包）:用于场景切换，更新场景，加载场景，卸载场景

通信管理(communionManager 包)：socket 通信，连接服务器的

操作管理(ControlManager 包)：用于设定游戏的操作键盘，和鼠标

游戏管理(GameManager 包): 包括游戏设定，音乐，背景默认设定。

游戏 Ai 管理（AiManager 包）：设定怪物一些 Ai，NPC 一些行为

游戏效果管理（EffectManager 包）:管理一些招式效果类以及动画

游戏错误管理（ErrorManager 包） 管理一些错误的情况

一 地图管理：（MapManager 包）

这个包下有很多的类用于地图上的管理。包括基础类 Map，功能加载地图，

Map 类下有以下的方法：

实现方案:

方案 1.元素构成的地图

将一些地图区分不同图片元件，将它绘制在场景里面。图片切割的大小按实际需要而定。例如：主窗口是 640x480 分辨率，切割每一块单元图片 32x32 大小。那么图片单元数目为 20x15=300 块。

如下面：

```
myMap = [ [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]];
```

那么在这个数组里面其中 “1”代表图片 1.“ 0”代表图片 0

同样的话，为了使某一些障碍物和主角进行碰撞，在数组里面 myMap 可以进行改变某一项的数值，这样可以用图片单元的隐含的信息进行判断。省去使用 hittstobject 带来的计算量。

带来的影响：

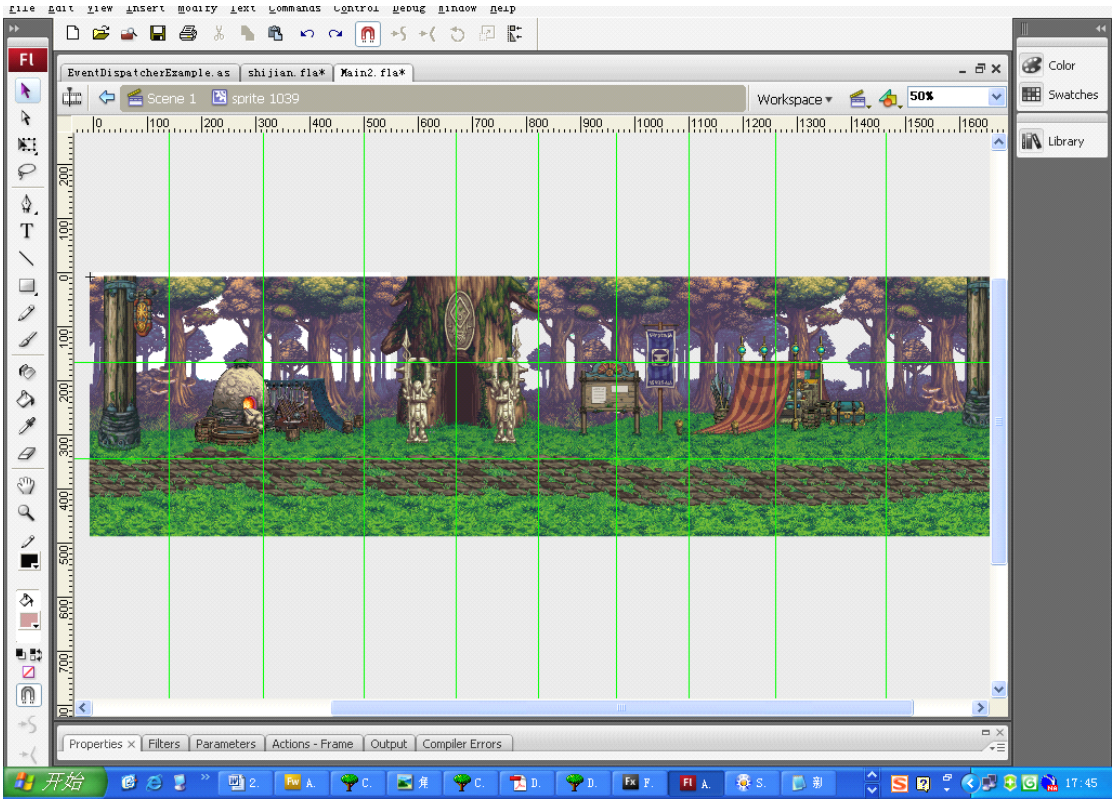
优点：

可以重用很多的图片单元模块，减轻内存的占有率。

由于地图上的许多区块是共用的，因此需要另外建立一组资料来些元素的排列顺序，这个排列顺序的资料列，我们通常都将之称为地图资料。重复使用元件，减轻内存占有。

缺点：

切割的图片单元如果过多会对游戏产生一些效率性的问题？至于这些效率性的问题在切割图片的时候需要注意考虑。



基类：

Class Map

{

基本属性：

MapName 地图名称或者索引号

MapPath 地图路径

Speed 为地图加入移动的属性值，这个值受英雄移动速度影响，因此制作时候需要特别处理

基本方法

LoadMap(path:String)；//加载地图

UnLoadMap(MapID:Map);//卸载地图

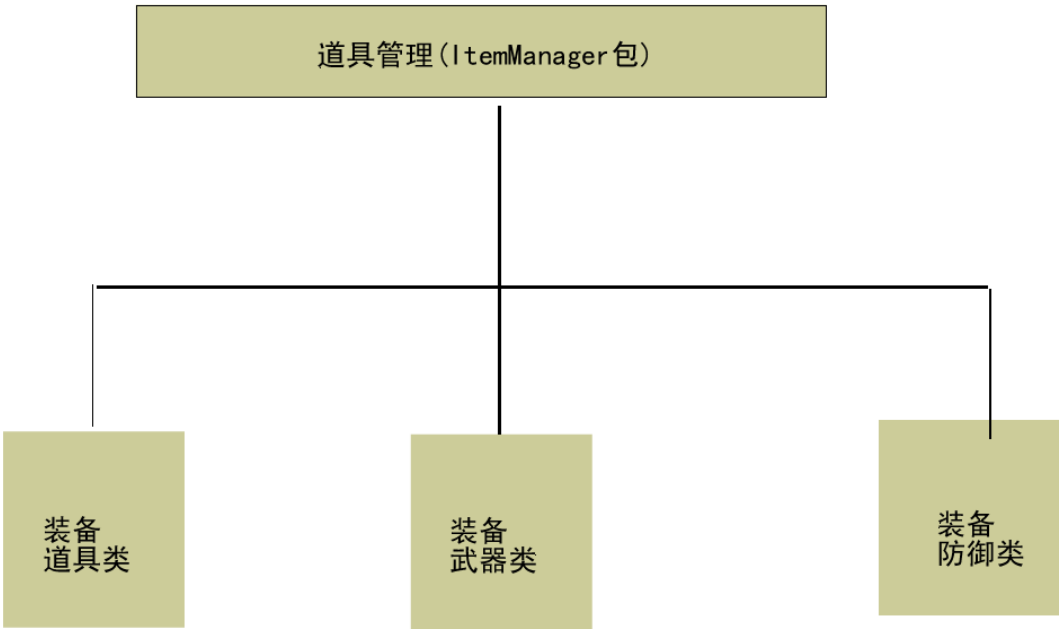
ChangeMap(path:String);//切换地图

DrawMap（map：Array）：//画制地图

}

二．道具管理(ItemManager 包)

这个包是负责角色的使用物品包括 道具（），武器（攻击） ，装备（防御）



2.1 基类

Class item

{

ItemName 物品名：

基本属性

基本方法：

//可以返回一些属性值，可以是

}

//道具类

Class Goods{

基本属性：

ItemName://道具名或者索引

道具附加的属性，包括火，冰，风，水 土，木等附加属性

基本方法：

//返回物品附带的属性(get)

}

例如可以是书：增加智力
可以是草药（当 hp 低于最大值的时候可以增加 hp 属性），
可以是有钱的宝石：增加金钱
可以是增加血量的石头：Maxhp

//（攻击）武器类

Class Weapon
{
基本属性：
WName // 武器名
WDamage //武器所带的属性
DamageValue//武器所带的附加效果

基本方法：
Damage（）:int //返回武器的破坏属性值（get ）return WDamage
}

武器中分为很多的武器，有它的共性，同样也有它的区别，同样可以呈现出不同的状态

问题集
如何解决武器切换问题？程序实现（实现的时候是需要更改不同的动画来达到更换武器的效果）

//（防御）装备基类 这里带有一点防御类

class Equip
{
基本属性：
EquipName 装备名称：

基本方法：
//装备带来的是返回一些属性值的(get)

}

装备可以有很多种的装备 如衣服，首饰，鞋子，但是装备都有一定共性。如名字，返回属性值的方法。（Return ）
如增加一些防御值，防御逃避几率。
衣服：增加防御值 Armor：int
首饰：增加某一些防御 Armor：int
鞋子：增加移动性能，Speed：int

三．**声音管理**（SoundManager 包）

这个包负责声音的特效，加载，卸载，场景音乐播放 停止 暂停，调整音量大小，获取音量大小，还有一些显示的问题；

问题集：

1. 解决声音缓冲处理问题？令游戏更加流畅
2. 错误的处理机制？怎样设立？加载成功和加载失败的时候处理方式如何？
3. 循环播放背景音乐？以及设置动作打斗的音效？

```
Class GameSound{

        基本属性：
          var VolumeValue:int//声音响亮的值的大小
          //    Posion：int//  声音的播放位置
             LoadMusic() //加载音乐
             UnloadMusic();//卸载音乐
             StartMusic(); //开始音乐
             StopMusic();//停止音乐
             PauseMusic();//暂停音乐
             SetMusicvolume//设置音乐的音效大小（set）int
             GetMusicvolume //获取音乐音效大小 （get）int
             GetMusicName //获取音乐的名字  (get) string
}
```

四．角色管理（CharacterManager）

这个包负责创建不同的主角和配角，包括创建角色，设置角色属性，获取角色属性，删除角色，分派角色的常见行为和技能等，设置怪物 Ai。写一个接口或者建议基类共享角色中的属性和方法（当创建主角和创建怪物的时候，也一样可以使用这样方法）；

制作使用的设计模式：
Strategy 模式；
工厂模式;
单例模式；
外观模式.
Decorator 模式：

4.0 角色的不同职业（职业系统）

注意：角色分为不同的职业，不同的职业会有不同的技能和行为。因此我们需要设计一种方式去动态分派不同职业的不同行为和技能属性。
1：准备一套技能的类，封装这些每一个技能类（一个基类，派生不同的子类，但实现方法不同）
2 ：设计一种可以更换武器的方式的模式，让角色可以更换自己的装备
例如：有一个种职业叫法师，他会放魔法，但是如果一种职业是斧头工，他只是会使用武器攻击，但是法师是不会使用那个方法的。

4.1. Character 角色类的基类

Class Character//为一个基础类，包括基本的 Speed 属性，Hp 属性，包括站，走，攻击，跳跃，跑等这些共性是可以由主角类和怪物等继承。

定义角色接口：

```
Public   interface Icharacter
{
    function Stand():void//  站
    function Walk():void; //走
    function Run():void; //跑
    function   Attack();//攻击方法是多种多样的
}
```

Class Character{
角色共有属性：
Hp：生命值
Hpmax:最大生命值
Speed：//速度值
Armour://防御值
JumpSpeed:上跳的速度

AttackPower: 物理攻击值
CharacterID 角色索引编码
Level :等级

角色共有的方法：

Setproperty ();//设置角色属性值
Getproperty();//获取角色不同属性
DelCharacter();删除角色
Walk();//角色走动//【LWalk();左走 Rwalk();右走】
Run();//角色跑动// 【RRun();右跑，Lrun(); 左跑】
Jump()://跳高
Attack();//攻击

}

主角类：
Class Human extends Character
{
基本属性：
特有属性

基本方法
特殊方法

Useweapon ()//使用武器
UseItem （）//使用物品的能力
Changeweapon（）切换武器的能力

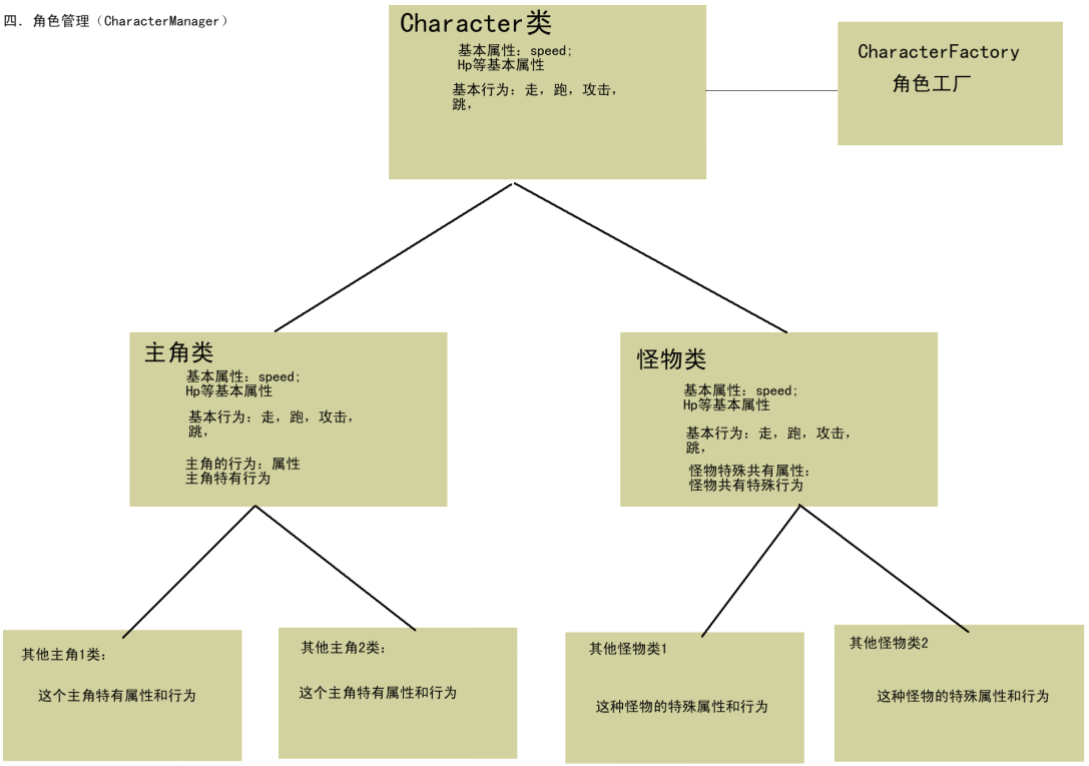
}

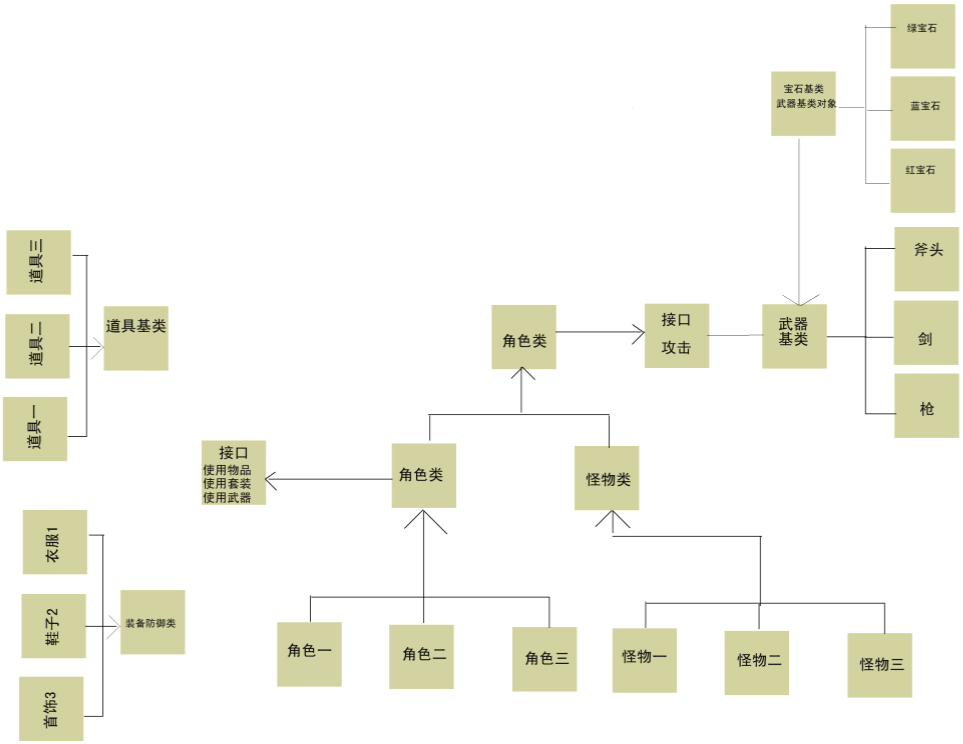
敌人类：（敌人是分不同类型：高级怪 和低级怪每种怪物的智商都不相同，Ai 设计的时候要特别注意这些）

Class Enemy
{
基本属性：
基本方法：

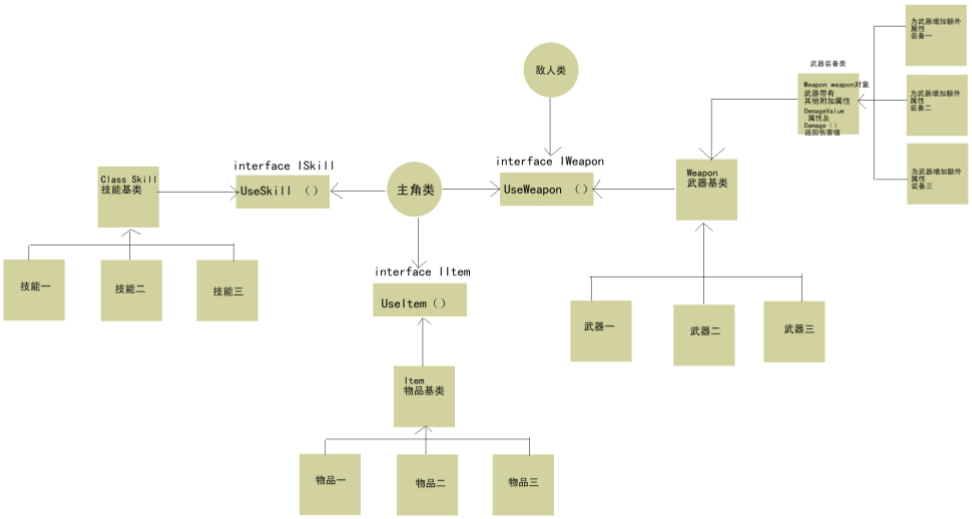
}

四. 角色管理（CharacterManager）





角色类和其他类的初步关系



角色类和其他类的初步关系二

角色创建工厂类

Class　CharacterFactory()：CharacterFactory

{

}

4.1.1 实现方法设想—工厂模式的应用

游戏一开始的时候提供了一些角色选定，我们将使用到工厂模式来创建我们所需要的角色类。（如创建主角一，主角二，主角三，主角四等）

例子：

package

{

public class CharacterFactory

{

public static var man: Character；

public static function CharacterFactory (CharacterName:String,): Character

{

```
switch (CharacterName)
{
case "1":
return new    Characterone();
break;
case "2":
return new Charactertwo ();
break;
case "3":
return new Characterthree();
break;

return null;
}
```

4.2 CharacterSkill 技能基类

角色拥有不同的职业，不同职业就会有不同能力.做一个技能类来方便为角色添加技能行为。

攻击方式：物理攻击和其他攻击

Class CharacterSkill
{
共有属性
SkillName:String//技能名称
Skillexplain:String //技能描述，包括带几点伤害，附加属性等等
DamageValue:int//技能带的伤害值；

共有方法:
Damage():int//技能所带的伤害值，附加属性

}

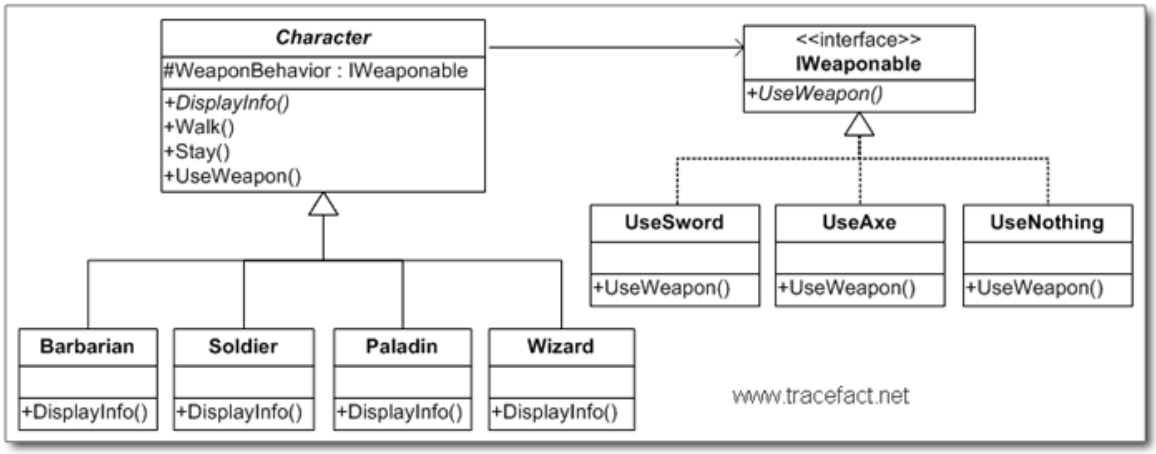
问题集：

问题一，技能列表如何制作？

问题二，能否做一个关于技能的列表类，这个类分派不同的技能给不同的角色

方案：提供一个技能使用接口，分别用不同的子类来实现这个接口
参考下面一个方案制作技能类

方案：
Strategy 模式；



举个例子：这里有四个不同的职业，可以使用武器包括剑，斧头等，甚至什么都不使用。

我们使用不同的武器子类去实现这个接口，为这个接口提供了不同的方法

Public interface Iweaponable{

Function UseWrapon（）：void;

}

Public class Character
{

Protected var WeaponBehavio：IWeaponable r; // 通过此接口调用实际的 UseWeapon 方法。 /
/ 使用武器，通过接口来调用方法
public void UseWeapon() {
WeaponBehavior.UseWeapon();
}

// 动态地给角色更换武器
public void ChangeWeapon(newWeapon: IWeaponable)
{
WeaponBehavior = newWeapon;
}

Public class UseSword implement Iweaponable
{
Public Function UseWrapon（）：void;

```
  {
      Trace("我使用了斧头")
   }
}

// 其中一个野蛮人职业，继承了角色类
Public class Barbarian extends Character
{
 Private var WeaponBehavior：UseAxe；
 Public function Barbarian()
{
 WeaponBehavior = new UseAxe();  // 野蛮人用斧

 }

}
```

4.3 Npc 创建

NPC： 游戏中的一种不可缺少的配角；在游戏作一个游戏向导的功能

不同的 NPC 都有相同的功能，会说话，会提供一些游戏信息给玩家。

```
Class NPC extends MovieClip
{
   基本方法：
  Speak（）;说话

}
```

五. 事件管理（GameEvent 包）

划分不同的情节,用不同的类来描述情节。包括不同的任务。

初步想法：写出不同的事件，每一个事件就关于一个故事情节，引发场景动画，引发一些对白言语等等；

5.1 基本概念：

在游戏中：

事件源——表示任何可以引发事件的对象。譬如，一个“人”、“坦克”、“建筑物”、“地面”。

事件——表示任何可以处理的事件。譬如，“感冒”、“射击”、“倒塌”、“有对象经过”。

响应者——表示任何对某事件感兴趣的对象。

响应器——表示对某事件感兴趣的对象对某一确定事件作出的反应。

特别的,对于过程：

通知——发生在事件与响应者之间。我们把它分为两种方式：有限听众式、广播式。对事件感兴趣的对象（响应者）只有确定的有限个（只有一个的情况下，可以叫做点对点式）的情况就是有限听众式。而对于广播式，事件并不知道会有哪些（个）对象对自己感兴趣。

它向所有可以接收事件通知的对象广播事件。

触发——响应者发现自己对特定事件需要做出相应的行动时就会触发事件处理器，并同时传递需要的事件信息给它。对于响应者，它也可以选择沉默——自己了解事件但并不作出行动。因此这个过程的决定权在响应者手上。

5.2 万事之鼻祖 Event

Event 类是所有事件的基类

我们需要一个类来表示所有事件的普遍性质。

获取或设置事件的名称

获取或设置事件的简单描述

获取或设置事件类型（枚举 EventTypes

// 获取响应者的集合

// 获取或设置事件的简单描述

// 通知响应者事件的发生

// 抛弃一个事件响应者，并把它从 Listeners 中移除。

// 抛弃所有的事件响应者

常见事件处理任务

下面是常见的事件处理任务，本章将介绍其中的每项任务：

■ 编写代码以响应事件

■ 阻止代码响应事件

■ 处理事件对象

■ 处理事件流：

■ 识别事件流信息

■ 停止事件流

■ 禁止默认行为

■ 从类中调度事件

■ 创建自定义事件类型

重要概念和术语

以下参考列表包含将会在本章中遇到的重要术语：

■ 默认行为 (Default behavior)：某些事件包含通常与事件一起发生的行为（称为默认行

为）。例如，当用户在文本字段中键入文本时，将引发文本输入事件。该事件的默认行为

是实际显示在文本字段中键入的字符，但您可以覆盖该默认行为（如果由于某种原因，

您不希望显示键入的字符）。

调度 (Dispatch)：通知事件侦听器发生了事

事件 (Event)：对象可以通知其它对象它所发生的情况。

■ 事件流 (Event flow)：如果显示列表中的对象（屏幕上显示的对象）发生事件，则会向

包含该对象的所有对象通知此事件，并依次通知其事件侦听器。此过程从舞台开始，并

在显示列表中一直进行到发生事件的实际对象，然后再返回到舞台。此过程称为

事件流。
■ 事件对象 (Event object)：此对象包含发生的特定事件的相关信息，当调度事件时，此信息将被发送到所有侦听器。

事件目标 (Event target)：实际调度事件的对象。例如，如果用户单击位于 Sprite（位
于舞台内）内的按钮，所有这些对象将调度事件，但事件目标是指实际发生事件的对象，
此处指单击的按钮。
■ 侦听器 (Listener)：对象或在对象中注册其自身的函数，用于指示发生特定事件时应通
知它。

5.3 建立事件机制或消息循环（暂定 不需要的地方）

解决问题：如何处理游戏中的事件？

如何设计调度器？

建立一个事件基类：（）暂定

Class GameEvent extends Event

{

基本的事件名

EventName

基本方法：

function get getEventname():String

function set setEventName(value:String):void

AddEvent（EventName：Event）：添加事件

RemoveEvent（）：删除事件

}

需要地方：

建立一个事件：MapEvent
这个事件继承了基类 Event 类，分别自定义两个加载时候出错的情况

```
public class MapEvent extends Event
{
    public static const MAP_COMPLETE:String = "mapComplete"; //用于加载图片完成
    public static const MAP_LOADER_ERROR:String = "mapLoaderError"; //用于加载图片出错的时候调用

public function MapEvent(type:String, bubbles:Boolean = false, cancelable:Boolean = false)
{
        super(type, bubbles, cancelable);
}
```

```
override public function clone():Event
{
        return new MapEvent (type, bubbles, cancelable);
}

}
```

六．对白管理（WordManager 包）：

用于管理语言，管理每一个 Npc 中的语言，包括读取语言，任务描述等等

Class Word{

基本属性：

Message：String

基本方法：

ReadWord();//读取文本，应用于 Npc 言语 和 任务描述语言

Speak()://说话

}

问题集：

解决 NPC 会说话的功能|？

如何解决任务的列表的提示功能？

七．场景管理（SenceManager 包）：（暂定）

用于场景切换，更新场景动画，渲染场景，卸载场景

Class GameSence{

CreateSence()://创建场景

UpdateScence()://更新场景

Render()://渲染场景，添加显示列表。渲染场景 人物等

ChangeScence(SenceName：String,MapID:Map，ScenceX：Number，ScenceY：Number)://切换场景

}

ChangeScence(SenceName：String,MapID:Map，ScenceX：Number，ScenceY：Number)://切换场景

切换场景：包括重新加载第二张不同的地图 LoadMap，重新设定图片位置。

Render()://渲染场景，添加显示列表。并重新渲染场景里面的人物；

在切换场景的时候，重新渲染地图，正确显示地图的位置。

八．通信管理(CommunionManager 包)：

socket 通信,连接服务器，读取服务器的数据。读取和写入数据库；
打斗时候的各种情况记录

网络游戏中最重要的一个地方，这个关于通信的是影响到游戏是否能够顺利进行。

九．操作管理(ControlManager 包)：

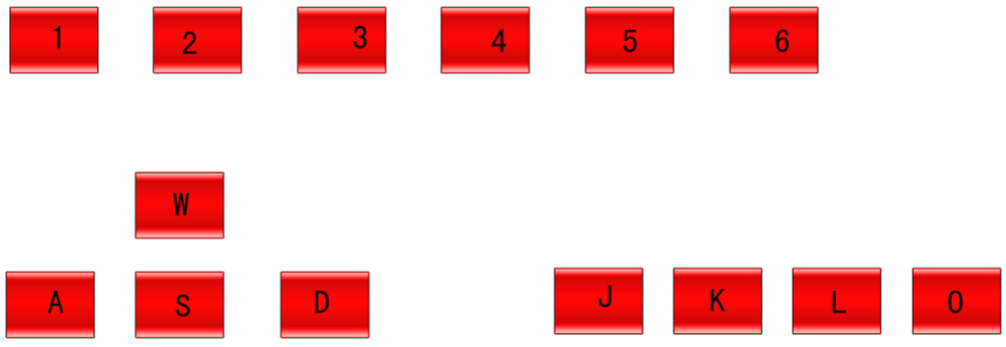
用于设定游戏的操作键盘，和鼠标，实现与键盘 鼠标进行交互 管理用户输入的数据

问题集：
1.怎样可以实现到调整操作键盘的键值？
2.怎样才可以通过控制人物的动作？
3.怎样才可以通过组合键实现招式的变化？
4.怎样才能使控制的角色实现到 8 个方向走动？
5 怎样实现一种按键处理机制，而不是纯粹执行按键时候，执行行为

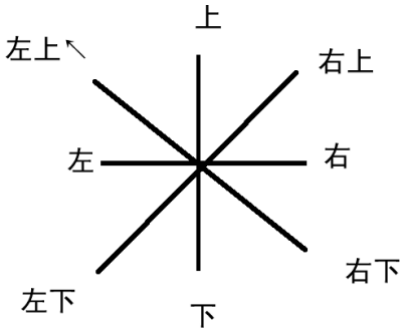
基础类：

Class GameControl{
基本属性：
上下左右键盘值
常用的控制键盘值
基本方法：
SetControl（）设置操作按键。包括上下左右方向，基本的键值调整//W .S. .A. D J. K. L. O 等 包括一些快捷键的设置
GetKeyvalue(); 获取键值
}

按键默认设置：



角色实现八个方向的控制：

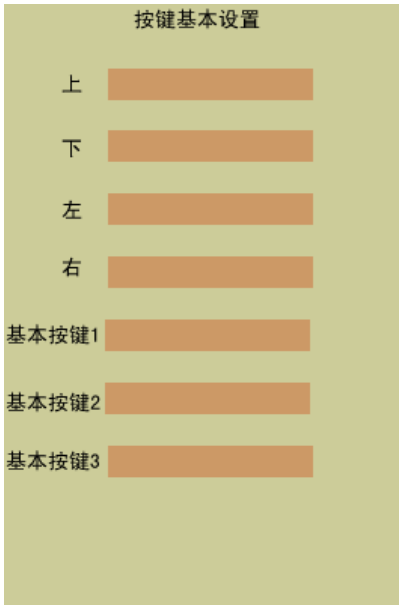


9.1 初步设想

可以用数组来保管好上下键盘输入状态，用户可以配合组合键来组合不同的输入方式和效果。

9.2 按键设置

玩家可以根据个人喜好，设定一些自己喜欢按键组合。



要解决的问题：1.显示按键当前使用的按键值？

2 这里保存的结果会影响到操作问题？是否需要进行数据库存取？

3.这功能需要吗？

十.游戏管理(GameManager 包): （暂定）

包括游戏设定，音乐大小开关，场景 特效开启 调整画面质量等。查看个人信息、技能表、自动保存问题，玩家在游戏中状态（包括时间的统计）

基本方法

GameSave();//游戏保存的问题。当玩家在游戏中死了的时候，应该如何去保存当前的数据

SoundOn(); //游戏开关 全局

//画面质量,高,中,低

十一游戏 Ai 管理（AiManager 包）（暂定）

这里包括一套关于 Ai 的算法机制，初步的一些想法：

包括一些有限状态机的设计模型，或者在有限状态机基础上去加入随机性去。

深入的技术：神经网络，包括 BP 算法，

制作的时候需要知道相关的流程图（策划部工作）

11.基本概念

基本例子说明：

1:**FSM**(Finite State Machine)有限状态机.这是最简单也最古老的 AI 技术,如果懂得程序的人员,可以将其简单的理解为一套 if,else 或者 switch,case 构成的条件判断.

我这里依旧拿原先的例子来说:

例:角色种类:熊类

case1: 当与玩家距离 50Pixel 范围内,警戒(警戒当然是种状态,属于行为规则)

case2: 当与玩家距离 10Pixel 范围内,主动攻击最近者

case3: 当受到攻击,立即攻击攻击者

case4: 攻击过程中,优先攻击对方队伍中最少 HP 人员

case5: 当攻击目标消失或死亡,则攻击 10Pixel 范围内最近者

case6: 当与攻击目标距离 10Pixel 内,追随攻击

case7: 当与攻击目标距离 10Pixel 外,警戒

case8: 当与攻击目标距离 50Pixel 外,返回初始点,在一定范围内随机移动

这可以说是比较典型的 FSM 设计,无论玩家做出什么行为,都有一套 AI 规则进行判断和行为,而且,仅仅是一套.不会有一种行为对应两套规则的可能.当然,这样即使加入更多的判断可能,我们也很容易觉得单调,因为确定性太强了,这就需要我们更多的引入随机性,即我们下面说的

基础类：

```
Class State//（状态类）
{



}
```

Class StateMachine（状态机）类

{

}

游戏效果管理（EffectManager 包）（暂定）

管理一些招式效果类以及动画。

技术难点：

1.加载地图，
2.碰撞检测，
3.物理攻击效果
4. 怪物 Ai 设计和关卡难度
5.事件机制如何设立？
6. 触发器如何编写
7.游戏卷轴设计
8.游戏角色的各种动作编程实现
9 技能列表如何制作？
10.游戏种的各种算法!
11.性能测试
12.防沉迷系统

加载地图？

第一要控制地图的加载的大小，如果太大了，加载起来就不会流畅，占有的内存就会增加起来，如果在网络差的情况下。时间消耗是玩家不可以忍受的。

第二．加载的地图如何使它发生移动，整体上如何实现卷轴的游戏？

第三．如何切换地图？如何判断切换地图的标志？切换后,如何将正确显示地图的坐标?

第四．地图需要切割分模块吗？假设需要，切割后的模块如何贴图在游戏当中.?可以共用一些图片单元模块吗?

3.物理攻击效果

打斗的效果,要存入到数据库里面进行交互？那么如何设计呢？打斗进行的时候,还有打斗结束的时候,这些数据如何保存|? 如何实现?

碰撞检测?

问题一,如何检测碰撞,假设在地图上有一百个物体碰撞点如何检测?使用 HitTestObject 吗|?效率高不高?

问题二，如何使用数组特殊性来检测碰撞？

怪物 Ai 设计和关卡难度?

如何控制怪物的 Ai?每一个关卡的怪物难度都是不相同的.Ai 的好坏决定了一个游戏的趣味性?太难会伤害玩家,太容易更加没有什么乐趣. 如何平衡 Ai 是游戏一个很关键的地方.

问题 2 怪物是分等级的,分别为普通怪,中等怪, 高级怪.还有一些 boss 这些 boss 随着关卡等级的不同,Ai 会发生相应变化

问题 3.npc 需要 AI 吗?npc 是会说话的,如何让它能说话呢？

如何设计关卡难度? 使用什么方式去使用?

问题 4 如何实现下面这些 ai 功能

敌人的 ai

第四章 群聚

基本群聚

群聚实例

避开障碍物

跟随领头者

第五章 以势函数实现移动

游戏软件 AI 中如何使用势函数？

追逐/闪躲

避开障碍物

成群结队

解决问题，怪物如何移动？
怪物如何打斗？
怪物如何攻击对手？
怪物如何闪避对手？
怪物是如何巡逻？
怪物如何警示？
怪物如何群对解打？
怪物如何跟着 boss 一起打？

问题 5：如果采用 bp 算法，应该如何去实现？神经网络单元，效率产生如何？

问题 6.如何根据游戏中的不同场景，产生出不同等级的怪物，每一次怪物是否是固定分配的？不同的场景中，场景会分派不同的怪物。这些怪物分为高级怪，低级怪，中级怪。每一种怪物的智商都不一样。应该如何创建不同的怪，以及不同智商；

5.事件机制如何设立？

游戏中分了很多的事件,每一个事件都不会相同,每一个事件都会触发一些特殊的事情发生,应该如何去设计这些事件类,游戏怎样调度?才是适合呢?

例如: 那一扇门,我走过去了,当我触发了这扇门,要准备切换地图了.是否我要写这样一个事件. Door 事件呢? 触发呢? 如何是,如何做?

7.游戏卷轴设计

一张地图,当我移动角色的时候,地图会发生相应的改变,这些相对运动的效果,应该如何做?

假设了我们切割一张大的地图,这些单元模块如何整体都发生移动呢?
如果我们使用一个容器把这些图片单元模块装入，那么我们只需要移动这个容器就可以了 是这样吗?

那么容器 如何设计?

问题 2.卷轴移动带来的问题？
解决：相对位移的问题！

8.游戏角色的各种动作编程实现

通过编程，可以实现了角色的移动，包括上下左右，左上 右上左下，右下等8个方向的控制。除了这个还要设计包括跑步。跳跃等一些常见的动作

10.游戏种的各种算法!

算法：包括 Ai，地图移动和人物移动关系。

11.性能测试

游戏中会影响游戏性能的包括：画面的质量，图片加载的大小，

12.防沉迷系统