**K**ingdom **R**ush

基于汇编语言的塔防游戏

刘桐彤 2012013310

杨妍喆 2012013323

邹豪风 2012013292

INSTRUCTOR: 王朝坤

**C**OURSE: 计算机网络体系结构(2)

**D**ATE: 2015/04/15

**C**ONTENT

[1. 游戏概况 2](#_Toc416874902)

[1.1 游戏简介 2](#_Toc416874903)

[1.2 操作方式 2](#_Toc416874904)

[1.3 怪与塔 2](#_Toc416874905)

[1.4 通关与失败 3](#_Toc416874906)

[2. 技术要点 3](#_Toc416874907)

[2.1 架构与逻辑 3](#_Toc416874908)

[2.2 地图寻路 5](#_Toc416874909)

[2.3 双缓冲显示 7](#_Toc416874910)

[2.5 资源索引管理 7](#_Toc416874911)

[3. 开发概况 7](#_Toc416874912)

[3.1 开发团队与分工 7](#_Toc416874913)

[3.2 文件组织图 8](#_Toc416874914)

[3.3 开发成果 8](#_Toc416874915)

# 1. 游戏概况

## 1.1 游戏简介

Kingdom Rush是一款经典的塔防游戏，玩家通过合理地分配金钱、部署建塔来阻止小怪及终极boss的进攻。

## 1.2 操作方式

|  |  |  |
| --- | --- | --- |
| 功能 | 操作 | 展示 |
| 建塔 | 鼠标点击空地  在环上选择塔型 |  |
| 升级 | 点击已建好的塔  在环上选择升级 |  |
| 卖塔 | 点击已建好的塔  在环上选择卖塔 |  |
| 按钮 | 鼠标点击 | C:\Users\Administrator\Desktop\temp.bmp |

## 1.3 怪与塔

塔： 4种类型的塔，分别可以升级。炮塔最贵，具有群攻(AOE)功能，魔法塔为攻击boss必备。其余两塔较为廉价，适合初始时购买。





怪： 4种类型的怪，可在每波攻击中看到。怪具有上下左右行走和倒地的不同动画。

## 1.4 通关与失败

玩家共有20条命，当一个小怪通过终点，即丧命一条。在7波攻击下，若攻击完成，依然还存活，则为通关。

# 2. 技术要点

## 2.1 架构与逻辑

代码分为逻辑层与绘图层。两部分较为独立，绘图层依靠调用逻辑层的接口更新数据，完成图像绘制。

在逻辑部分，为了有效组织游戏数据，定义了塔(Tower)、怪物(Enemy)、子弹(Bullet)、动画(Animate)等结构体及相应的方法。早在开发之前，我们便完成了相关结构体的设计文档。部分定义如下所示：

1、塔 TOWER

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型及范围 | 备注 |
| 类型 TYPE | Integer :[0..3] | 箭塔：1  魔法塔：2  兵营：3  炮塔：4 |
| 等级 DEGREE | Integer :[0..4] |  |
| 造价 COST | Integer :[0.999] |  |
| 售价 SELLCOST | Integer :[0..999] |  |
| 攻击力 ATTACK | Integer :[0..999] |  |
| 攻击范围 RANGE | Integer : [0..999] |  |
| 范围攻击 IS\_AOE | Boolean | 只有炮塔为TRUE |
| AOE范围 AOE\_RANGE | Integer :[0..99] |  |
| 位置 POSITION | Coord |  |

相关行为:

|  |  |  |
| --- | --- | --- |
| 名称 | 说明 | 参数 |
| CreateTower | 建造塔 |  |
| SellTower | 卖塔 |  |
| UpdateTower | 升级 |  |
| SearchAndAttack | 搜寻攻击目标 |  |

2、怪物 ENEMY

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型及范围 | 备注 |
| 类型 TYPE | Integer :[0..99] |  |
| 移动速度 SPEED | Integer :[0..999] |  |
| 起始位置 START\_POS | Coord |  |
| 终止位置 END\_POS | Coord |  |
| 当前位置 CURRENT\_POS | Coord |  |
| 当前朝向 CURRENT\_DIR | Integer :[0..3] | 0:↑ 1：→ 2：↓ 3：← |
| 当前生命值 CURRENT\_Life | Integer :[0..9999] |  |
| 最大生命 MAX\_LIFE | Integer :[0..9999] |  |
| 死亡金钱 Money | Integer :[0..999] |  |

相关行为:

|  |  |  |
| --- | --- | --- |
| 名称 | 说明 | 参数 |
| EnemyActivate | 激活怪物：在游戏地图的起点处加入该怪物 |  |
| EnemyMove | 移动 |  |
| EnemyDie | 死亡：从怪物集合中移除 |  |
| Enemy\_CheckDie | 判断敌人生命值是否为零 |  |

3、一局游戏信息 GameInfo

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型及范围 | 备注 |
| 玩家生命 Player\_life | Integer :[0..20] |  |
| 玩家金钱 Player\_Money | Integer: [0..9999] |  |
| 游戏状态 state | Integer :[0..2] | 1 胜利  2 失败 |
| 当前轮次 Round | ROUND |  |
| 总轮次数 TotalRound | Integer :[0..20] |  |
| 当前时钟 CUR\_TICK |  |  |
| 当前塔的集合 TowerArray | Array TOWER |  |
| 当前怪物集合 EnemyArray | Array ENEMY |  |
| 轮次数组 RoundArray | Round Array [0..20] |  |

相关行为：

|  |  |  |
| --- | --- | --- |
| 名称 | 说明 | 参数 |
| LoadGameInfo | 载入游戏信息 |  |
| StartGame | 开始游戏 |  |
| ResetGame | 重新开始 |  |

4、每轮信息 ROUND

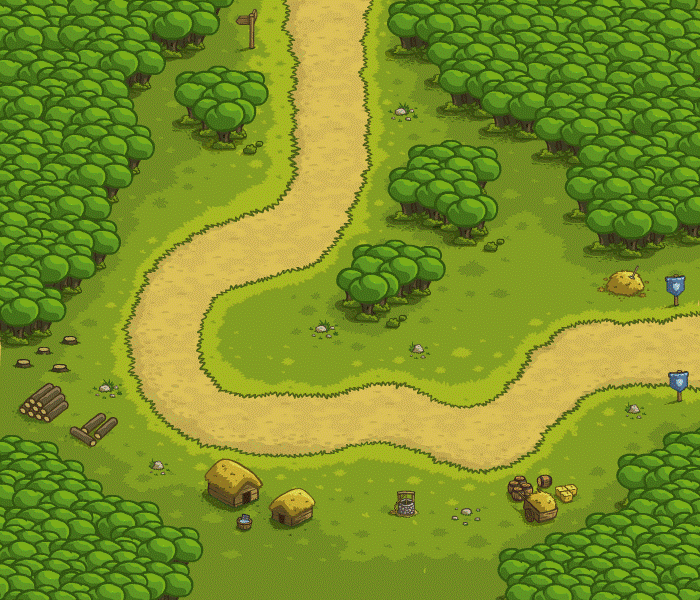
|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型及范围 | 备注 |
| 触发时刻 Trig\_Tick | Integer |  |
| 怪物数量 EnemyNumber | Integer: [0..9999] |  |
| 怪物数组 EnemyArray | Enemy Array [0..999] |  |

说明：以上列出的为部分的结构体定义。详细定义请见源代码struct.inc。

## 2.2 地图寻路

1、地图模型。

为了构建地图模型，我们首先在原地图的图片上对道路进行了标注。

**地图简化**

如图所示， 我们以原地图的png图片为素材，利用图像处理软件将道路和背景填充为两种色。然后，我们编写C语言代码来解析经过处理的地图文件，将其转换为0,1的字符点阵。于是，我们得到了一个700\*600的0,1点阵，其中0代表背景，1代表道路。这样便得到了一个比较好用的地图模型。

2、怪物寻路算法。



**寻路算法：起点与终点**

如图，怪物进入地图后，需设法沿着道路走到终点处。我们需要设计怪物的寻路算法。我们采用的方法是，在道路上每隔一段距离，设置一个驿站。怪物当前的移动方向是朝向它下一个驿站的。当它到达一个驿站后，下一个目标位置即为下一个驿站。每当怪物需要移动时，首先判断它与驿站的方位。例如，若驿站在左下方，则当前的一步应朝向左、或下方移动。当上一步向左移动时，这一步就选择向下移动，使得每次移动方向尽量与上次不同，以实现较好的移动效果。



**寻路算法：驿站标注**

## 2.3 双缓冲显示

由于在Kingdom Rush中有非常多需要绘制的对象，包括地图、空地、塔、怪物、子弹、风车、特效等等。如果采用一般的方法绘图和显示，将由于过高频率的刷新而引起屏幕的快速闪烁。为了防止这种情况的出现导致体验变差，所以我们采用双缓冲技术进行绘图。

双缓冲技术从具体上来说就是，在每次收到WM\_PAINT消息进行绘图时，分为三层DC（设备上下文）进行渲染：ScreenDC，MemoryDC，ImageDC。首先将Bitmap对象加载到ImageDC中，然后利用绘制到MemoryDC。重复以上过程，直至将所有需要绘制的对象都绘制到MemoryDC中，最后一次性把内存中的这张图复制到窗口设备中，这样就避免了闪烁情况的出现。

具体实现后绘制的效果非常流畅，没有任何卡顿情况的出现。

**注：**通过调用wingdi库提供的api：TransparentBlt，指定背景为透明。

## 2.4 资源索引管理

由于Kingdom Rush中对象众多，再加上需要对每个对象实现动画、声音效果，游戏中需要维护和管理的图片、声音资源非常之多，这对游戏的资源索引设计和管理提出了很高的要求。

考虑到塔、怪物等主要对象都有大量实例，为每个实例都导入所需资源将导致资源的大量浪费，数据冗余很严重。所以我们将数据与资源分离开来，仅在数据中保留资源的索引，资源单独设计结构来维护。从具体实现上来说，在塔和怪物等对象，添加了Type属性来作为资源的索引，另外资源单独利用BitmapInfo等结构体进行维护。通过这样的方式，尽可能地节约了资源占用的空间，也方便了资源的统一管理。

# 3. 开发概况

## 3.1 开发团队与分工

团队成员：

|  |  |  |
| --- | --- | --- |
| 刘桐彤 | 2012013331 | 18800102016@163.com |
| 杨妍喆 | 2012013323 | yangyz12@mails.tsinghua.edu.cn |
| 邹豪风 | 2012013292 | zhfthss@126.com |

团队通过集体开发与自由开发相结合，集体开发讨论、整合，自由开发高效地完成各自分工的任务。

邹豪风、桐彤主要负责逻辑架构，妍喆主要负责图片、动画与音乐，桐彤还负责建塔、开始界面等。第一次进行非精英管理的团队合作，各自选择各自的任务，并按时完成。这种模式非常考验自我对团队的负责意识。

## 3.2 文件组织图

文件主要分为四部分：逻辑代码、GUI代码、资源管理代码、资源文件。文件结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 逻辑 | struct.inc | 定义结构体 |  |
| core.inc  core.asm | 定义并实现游戏逻辑数据与定时器操作。 | 对于每个结构体，实现了其create、delete、update的相关函数。 |
| GUI | main.inc  main.asm | 实现加载窗口，图形、动画绘制、音效等 |  |
| 资源  管理 | resource.h | 定义资源文件中IDB\_XXX的序号 | 将不同类别的图片按段分布，实现索引管理 |
| resource.rc | 将icon, bmp加载到资源文件 |  |
| 资源  文件 | resource/images | 图片 | 147张 |
| resource/music | 背景音乐及音效 | 13种音效 |

## 3.3 开发成果

共计3000多行的汇编代码，让我们对汇编语言更加熟悉的同时，也增强了自己的代码控制能力。每过一周，我们都会将自己不满意的冗余代码重构，使得游戏有了良好的接口与可扩展性。

github地址：https://github.com/yangyanzhe/KingdomRush.git

demo视频： http://v.youku.com/v\_show/id\_XOTM0NjE0NDY4.html

demo视频密码：demo