**编 号：**

**审定成绩：**

**重庆邮电大学**

**智能游戏开发设计报告**

**设计（论文）题目： 智能游戏开发设计 – 坦克大战**

**学 院 名 称：计算机科学与技术**

**学 生 姓 名：韦鑫桥、白皓月、王淇锐**

**专 业：计算机科学与技术**

**班 级：04051702**

**学 号：2017212007、2017211990、2017**

**指 导 教 师：舒禹程**

**时 间：2019年 12月**

**摘要**

本游戏抱着学习cocos2d-x游戏引擎，巩固所学的c++知识，最终定位为2D跨平台坦克大战游戏，游戏大体设计上跟随90经典坦克大战。结合视频教程及网上各种博客资料，以cocos2d-x引擎（c++语言）进行游戏程序的编写。

完成内容有：可以单双人游戏，有20个关卡，初始时可以自由选择关卡，过关后开启下一关卡；坦克移动可以按着两个方向键让其“自动寻路”，当某一方向阻碍时尝试往另一方向移动，不必遇到阻碍时再按另外的方向键；制作了玩家的ui控件：血量显示和物品栏，可缩放和拖动到自己舒适的位置；制作了物品栏专门显示玩家获得的生效的物品（buff），buff有叠加的新效果；玩家控制坦克的按键是由xml文件初始化的，可以更改xml文件来设置自己喜欢的方向键、开火键、“造路”键（获得一种特殊的叠加buff开启造路功能，可以在坦克移动过程中在坦克身后更改地图元素）

**【关键词】**游戏 2D c++ 坦克大战 cocos2d-x

**目 录**

前 言

第 一 章 游戏类型及内容定位

第一节 选题需求分析

一、当前国内游戏市场分析

二、结合结合当前广受欢迎的小游戏

第二节 游戏内容定位分析

一、国内手游的特点分析

二、风格定位

三、内容定位

第 二 章 游戏设计分析

第一节 游戏内容(情节创意)的构想

一、整体构想

二、游戏细节描述

第二节 游戏的功能设计

一、游戏整体功能

二、功能模块的组成

第三节 关键模块代码、效果图

第 三 章 游戏功能测试

1. 测试计划方案
2. 游戏模块功能分析

一、游戏模块测试、集成测试

二、游戏整体风格、体验反馈

第 四 章 总结分析

1. 游戏制作总结分析
2. 组内分工说明及感想收获

结 论

致 谢

参考文献

**前 言**

近年来，中国游戏市场增长呈现理性态势，参考2015年度产业报告数据：2015 年，中国游戏(包括客户端游戏、网页游戏、社交游戏、移动游戏、单机游戏、电规游戏等)市场实际销售收入达到**1407.0 亿元**人民币，同比增长**22.9%**。整个市场的上升逐渐稳定，上涨趋势也再慢慢呈理性态势，游戏市场的发展正在渐渐的进入成熟期，摆脱“忽高忽低”毫无规律可循的浮躁期。本学期，我们也顺应时代趋势，选择了“游戏开发设计”课程。作为计算机人，即便日后不从事游戏开发工作，也应该对游戏开发有一定的了解。

一个学期的课程学时不多，我们所学也只是九牛一毛，不过，通过本课程的学习，我们从最初的只会玩游戏、只能从玩家角度评价游戏，到后来学习游戏开发的一些原理技术：游戏中的高级图形技术、三维场景的组织与描绘等内容，内容基础、宽泛，却指导了我们进一步学习的方向。

俗语讲：师父领进门修行在个人，通过最后一次的课程设计大作业，我们除了翻阅书本，还去图书馆查阅相关书籍、观看游戏开发设计及C++进阶课程等相关视频，从中更加深了对本课程的理解及对知识的掌握。

编者

2019年12月21日

**第一章 游戏类型及内容定位**

**第一节 选题需求分析**

**一、当前国内游戏市场分析**

据2015年度产业报告—中国市场游戏发展现状与前景趋势显示：中国游戏市场增长呈理性态势，客户端游戏市场发展进入固守阶段，单机游戏市场快速发展，移动游戏市场持续增长。

而其中的手机游戏行业，据《中国手机游戏行业调查分析及市场前景预测报告》显示：随着智能终端性能的不断提升与迅速普及，移动开发平台月销售额10万以上的手机网络游戏约占总比20%，手机网络游戏用户、充值金额持续上涨，2013年以来增长尤其明显，未来手机网络游戏市场无疑仍将保持一个良好的增长态势。中国手游市场现阶段还处于手机游戏扩展阶段，在手游用户群体扩展的同时，还需要不停的培养游戏用户的消费概念。但因为中国手游用户群体基数大，所以中国手游市场将在近几年超过日本、美国等手游成熟国家成为世界上最大的手机游戏市场。中国手游占国内整个发行商市场份额17.9%，继续保持在市场第一位;触控科技紧随其后以16.2%的发行商市场占有率，排在第二位;飞流游戏与乐逗游戏处于发行商市场的第二梯队，分别行业占比达到13.2%、13.0%;昆仑游戏、蜂巢游戏与热酷科技构成发行商市场第三梯队，分别占比达到11%、9.1%、7.8%。2014年全年国内手游市场规模超过200亿元，比2012年的112.4亿元增长70.9%。

从产业链角度看，手游研发环节进入门槛较低，处于发展初期，该环节行业集中角度相对端游和页游较低，手游行业间的并购持续快速增长。

**二、结合当前广受欢迎的单机手机小游戏**

1.如今各式各样的手游很多，让人眼花缭乱。而老少咸宜的游戏更容易流行开来，单机也让这些游戏在火车、地铁等网络不畅的地方也能玩。因此我们想做一个简洁化的单机手游。目前定位为坦克大战游戏。

2.当前较流行的“开心消消乐”就是难度适宜，画面精美，题材到位的手机益智游戏。

3.为了增强一些游戏性，我们添加了简单的过关级别(关卡)和分数。

**三、技术可行性**

1.我们将本次完成作业坦克大战游戏功能定位为在较细致全面的基础上尽量简单些，目前组员有一定编程基础，经过一个学期的学习，加上最后几周的自我学习，有了一点游戏设计概念和知识基础，设计简单的小游戏基本可行。

**第二节 游戏内容定位分析**

1. **一些目前大热的小游戏的特点分析**

我们注重参看了一些有自己风格的游戏。

① 像开心消消乐，风格Q萌，画风精致可爱。这也是大多数小游戏的风格。

② 设置的音效随操作结果的不同而有所区别。音效体验作为游戏的重要环节，除了根据以往经验设置，还应该反复调试，以最佳感官体验为准。

③ 简单游戏的受众目标极广，游戏应该简单，符合大众能力水平，但是如果游戏设置关卡太简单就无法挖掘用户潜在兴趣，所以关卡及游戏形式的设定这也是一个考验分析能力的技术活。

1. **游戏风格定位**

1.综合我们现在的能力和水平，以及产品受众规模、知名度尚无等因素，我们游戏风格定位于功能简洁。基于这是一个坦克大战战斗性游戏，背景设置较为“严肃”，以一张张地图（关卡）为背景。

**三、 游戏内容定位**

游戏分为几个模块。

1.玩家按特定的键开火、移动（默认玩家一键WASD移动，键J开火，玩家二方向键移动，数字键2开火）、造路（玩家一K，玩家二1）。

2. 每个坦克初始血量都是100，子弹击中坦克即掉20滴血。

3.ui控制块，包含显示玩家所需的信息，目前有两个：血量和物品栏

4.敌方AI控制：控制敌方坦克的各种行为

5.物品管理：管理物品的生命周期、各种效果和显示

**第二章 坦克大战游戏设计分析**

**第一节 游戏情节的构想**

**一、游戏整体构想**

1.游戏很简单，通过方向键改变方向，长按移动坦克，碰撞到物品时获得相应物品的buff（可叠加），我机一旦被敌机撞击，游戏随即结束，用户可选择开始下一轮游戏或退出。

2.敌坦属性有发射子弹速度和移动速度，敌坦初始出现三个，当一个敌坦被子弹击中到血量到0就消失在屏幕上，随后在指定位置中随机一个位置生成一个敌坦（属性随机）

3.每击毁一架敌坦，玩家即可得到10分积分，所以玩家总共得到100分即可进入下一关。

**二、游戏细节描述**

**（一）基本功能实现中的细节**

1.游戏背景采用tmx的地图，进入游戏时可以左右拖动选择关卡（使用PageView实现）；坦克与地图的碰撞时通过每一帧检测坦克的前速度行（坦克每一帧还要移动速度大小的像素，防止移动过头检测过慢）像素中是否有实体元素，通过使用getTileGIDAt函数得到GID，GID在Tiled软件中可以观察到时什么砖块。

2.敌机出现在屏幕上方三个出生位置，随机生成属性（速度，射速）不同的坦克，死一个生成一个，并且附带出生特效。

3. 开始游戏玩家坦克位置在屏幕左下方，初始化时附带无敌效果，持续3秒。

4.当玩家击毁一个敌坦时随机在地图中的t1-t10（Tiled Map地图的object层）中生成一个物品buff。

5.玩家和敌坦均继承Tank类，Tank中以枚举变量定义了射速、移速和方向，Tank继承Sprite，拥有addController方法以让自己的控制器是自己的子节点，鉴于cocos2d-x与mvc设计模式的违和感，采用实体-控制模式，ui则镶嵌其中，控制器的加入能很好的降低实体对每一帧的检测压力，让事件的先后过程更加清晰

6.为了方便场景跳转，创建了一个场景管理器，在AppDelegate.cpp中首先运行场景管理器的默认场景，后面的每一个场景类均包含SceneManager的指针，方便直接用场景指针跳转场景

7.对发射子弹的处理更贴合现实世界，首先Tank类有一个私有成员数据Vector<Bullet \*>弹夹，按下开火键时在坦克内部判断弹夹是否为空，为空时创建弹夹容量多的子弹，bullet->p\_tank = this，交代子弹的所属坦克方便子弹与敌方坦克的碰撞检测，对友方坦克不作用；然后修改Vector类源码，增加自己的\_popBack方法，不然原来的popBack方法会删除子弹，无法再引用（不符合现实），然后removeFromParentAndCleanUP(false)，再把子弹的父节点设置为地图

8.子弹也有控制器BulletController，控制子弹与坦克、地图砖块的交互，子弹碰到敌人或者碰到地图可碰撞单元则消失。

**（二）针对用户体验的细节**.

1.对移动在90经典坦克大战上只能按一个键，加入“双键移动”，移动更加流畅；反复修改代码避免坦克嵌入地图动弹不得的bug。

2.加入玩家的ui控件，让玩家可以随意拖动和缩放控件在合适的位置。目前的血量控件以进度条的方式呈现，物品栏的加入让buff更加清晰明了。

3.音效方面则是使用经典90坦克大战的原声，使经典再现，勾起玩家的回忆。

4.在90坦克大战基础上添加了大量的物品buff，丰富了玩法和趣味性。

5. C++语言简单易懂，功能强大，我们利用C++语言编写程序，ui部分也使用C++，大大巩固了我们的编程能力。

**第二节 游戏的功能设计**

**一、游戏功能模块组成**

1.模块组成：



图2-2

1. **关键模块代码、效果图**

**一、关键模块代码及其效果图**

**1.游戏Logo**

**（1）代码**

**效果图：**

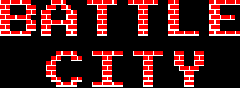
****

图3.1-1 Logo效果图

**（2）游戏关卡选择代码**

void LevelLayer::display\_levels()

{

Vec2 origin = Director::getInstance()->getVisibleOrigin();

Size winSize = Director::getInstance()->getWinSize();

PageView \*pageView = PageView::create();

pageView->setTouchEnabled(true);

pageView->setSize(Size(winSize.width, winSize.height));

pageView->setPosition(Point(0, 0));

//创建20个page

for (int i = 1; i <= 20; i++) {

Layout\* layout = Layout::create();

layout->setSize(Size(winSize.width, winSize.height));

ImageView\* imageView = ImageView::create(StringUtils::format("map%d.png", i));

imageView->setScale9Enabled(true);

//imageView->setContentSize(Size(600, 600));

imageView->setPosition(Point(layout->getContentSize().width / 2.0f, layout->getContentSize().height / 2.0f));

imageView->setTouchEnabled(true);

imageView->addTouchEventListener([this, i](Ref\* pSender, Widget::TouchEventType type) {

switch (type)

{

case Widget::TouchEventType::BEGAN:

log("touch page %d", i);

break;

case Widget::TouchEventType::ENDED:

tsm->goGameSceneWithMap(i);

break;

default:

break;

}

});

layout->addChild(imageView);

Text\* label = Text::create(StringUtils::format("level %d", i), "fonts/Marker Felt.ttf", 24);

label->setColor(Color3B(255, 0, 0));

label->setPosition(Point(layout->getContentSize().width / 2.0f, layout->getContentSize().height / 2.0f));

layout->addChild(label);

pageView->addPage(layout);

pageView->setSwallowTouches(false);

}

pageView->addEventListenerPageView(this, pagevieweventselector(LevelLayer::pageViewEvent));

this->addChild(pageView);

}

**效果图：**

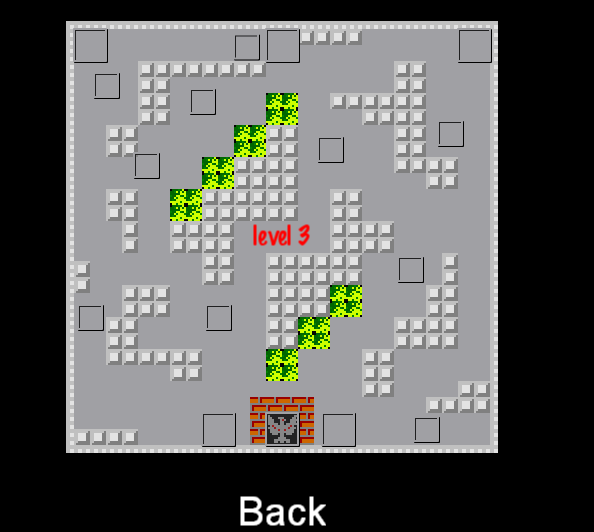


图3.2-1 关卡选择效果图

**（3）玩家坦克控制器的职责：**

bool TankController::init(Tank \*tank)

{

if (!Node::init())

return false;

this->tank = tank;

tank->bulletIce = false;

m\_moveListener = EventListenerKeyboard::create();

m\_fireListener = EventListenerKeyboard::create();

m\_buildListener = EventListenerKeyboard::create();

tankBorn();

listenMove();//Include collide

listenFire();

listenBuild();

scheduleUpdate();

return true;

}

void TankController::update(float dt)

{

if (tank->HP <= 0)

{

AudioEngine::play2d("sounds/explosion-player.mp3");

tank->gameLayer->m\_map->tankSet.eraseObject(tank);

auto animation = AnimationCache::getInstance()->getAnimation("tankboom");

auto action = Animate::create(animation);

tank->runAction(Sequence::create(action, CCRemoveSelf::create(), NULL));

unscheduleUpdate();

}

if (playerHP != nullptr && tank->HP >= 0 && tank->HP <= 100)

playerHP->getProgressBar()->setPercentage(tank->HP);

}

**（4）发射子弹代码：**

void TankController::listenFire()

{

/\*if (tank == nullptr)

return;\*/

m\_fireListener->onKeyPressed = [&](EventKeyboard::KeyCode keycode, Event \*event) {

if (keycode == key\_fire)

{

//tank->fire(); //经典的一按一发

//tank->fire(); //改进：按键连发

//先声夺人

if (tank->shootSpeed < 7){

AudioEngine::play2d("sounds/shoot.mp3");

}

else {

AudioEngine::play2d("sounds/fire-powerful.mp3");

}

m\_bullet = tank->shootOneBullet();//打出来的子弹

tank->gameLayer->m\_map->addChild(m\_bullet, -1, "bullet");

m\_bullet->setPosition(tank->getPosition());//其实可以这样粗略处理子弹位置

m\_bullet->gameLayer = tank->gameLayer;

m\_bullet->m\_direction = tank->m\_direction;//获取按下键时的方向

m\_bullet->tankShootSpeed = tank->shootSpeed;

m\_bullet->ice = tank->bulletIce;//ice

m\_bullet->addController();

}

};

m\_fireListener->onKeyReleased = [&](EventKeyboard::KeyCode keycode, Event \*event) {

if (keycode == key\_fire)

{

//log("space release");

//tank->stopFire(); //松开熄火

}

};

\_eventDispatcher->addEventListenerWithSceneGraphPriority(m\_fireListener, this);

}

Bullet\* Tank::shootOneBullet()

{

//"自动步枪"，自动装弹，外界不用担心打完

if (!m\_cartridgeClip.empty())

{

m\_cartridgeClip.back()->removeFromParent();//子弹脱离坦克但不删除

return m\_cartridgeClip.\_popBack();//手改源码，自己的popBack

//这里先不忙popBack,即删除这颗子弹,而是射向外界，所以修改了源码

}

loadBullets();

m\_cartridgeClip.back()->removeFromParent();

return m\_cartridgeClip.\_popBack();

}

void Tank::loadBullets()//默认子弹

{

for (int i = 0; i < 10; i++)

{

Bullet\* bullet = Bullet::create("bullet.png", bulletPower);

this->addChild(bullet);

bullet->p\_tank = this;

bullet->p\_tankName = this->getName();

m\_cartridgeClip.pushBack(bullet);

}

fixBulletPos();

}

bool BulletController::init(Bullet \*bullet)

{

if (!Node::init())

return false;

m\_bullet = bullet;

m\_bullet->fly();

listenCollideMap();

listenCollideTank();

listenCollideEnemy();

return true;

}

**效果图：**

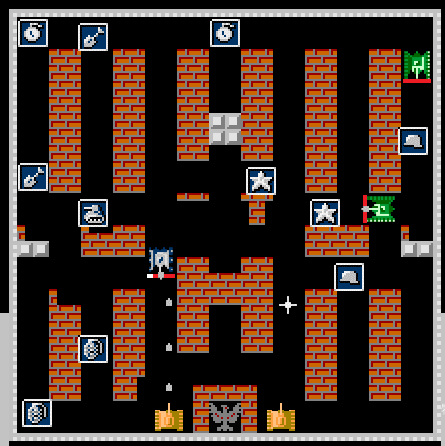


图3.4-1 绘画敌机(子弹)效果图

**（5）敌方坦克死亡效果**

void Enemy::update(float dt)

{

if (HP <= 0)

{

AudioEngine::play2d("sounds/explosion-enemy.mp3");

auto animation = AnimationCache::getInstance()->getAnimation("enemyboom");

auto action = Animate::create(animation);

//this->removeAllChildrenWithCleanup(true);

this->runAction(Sequence::create(action, CallFunc::create([&]() {

gameLayer->totalScore += 10;

gameLayer->m\_map->genRandomProp();

gameLayer->genEnemyRandom();

gameLayer->m\_map->tankSet.eraseObject(this);

}), CCRemoveSelf::create(), NULL));

unscheduleUpdate();

}

}

**效果图：**



1. **道具栏相关代码**
2. bool Inventory::init()//智能控件的话， 要引用一个东西（有物体的指针），当东西为nullptr时自动变为grid
3. {
4. focusGrid = nullptr;
5. //物品说明
6. label = Label::createWithTTF("物品说明", "fonts/SIMYOU.TTF", 12);
7. label->setAnchorPoint(Vec2(0.5, 0.5));
8. label->setPosition(Vec2(this->getContentSize().width / 2, this->getContentSize().height \* 0.25));
10. label->setDimensions(this->getContentSize().width, this->getContentSize().height \* 0.25);
11. //label->setColor(Color3B::RED);
12. this->addChild(label);
13. //RichText::create();
14. //物品栏
15. isTouched = false;
16. auto grid\_a = Sprite::create("grid.png");
17. auto grid\_b = Sprite::create("grid.png");
18. auto grid\_c = Sprite::create("grid.png");
19. auto grid\_d = Sprite::create("grid.png");
20. auto grid\_e = Sprite::create("grid.png");
21. auto grid\_f = Sprite::create("grid.png");
22. grid\_a->setName("grid.png");
23. grid\_b->setName("grid.png");
24. grid\_c->setName("grid.png");
25. grid\_d->setName("grid.png");
26. grid\_e->setName("grid.png");
27. grid\_f->setName("grid.png");
28. grid\_a->setUserData("null");
29. grid\_b->setUserData("null");
30. grid\_c->setUserData("null");
31. grid\_d->setUserData("null");
32. grid\_e->setUserData("null");
33. grid\_f->setUserData("null");
35. a = grid::create(grid\_a, grid\_a, [this](Ref \*pSender) {
36. /\*if (a->isEmpty){
37. label->setString("空");
38. }else{
39. label->setString(a->description);
40. }
41. printDialog();
42. clearLabel(a);\*/
43. focusGrid = a;
44. showDetails(a);
45. });
46. b = grid::create(grid\_b, Node::create(), [this](Ref \*pSender) {
47. /\*if (b->isEmpty){
48. label->setString("空");
49. }else{
50. label->setString(b->description);
51. }
52. printDialog();
53. clearLabel(b);\*/
54. showDetails(b);
55. focusGrid = b;
56. });
57. c = grid::create(grid\_c, Node::create(), [this](Ref \*pSender) {
58. /\*if (c->isEmpty){
59. label->setString("空");
60. }else{
61. label->setString(c->description);
62. }
63. printDialog();
64. clearLabel(c);\*/
65. showDetails(c);
66. focusGrid = c;
67. });
68. d = grid::create(grid\_d, Node::create(), [this](Ref \*pSender) {
69. /\*if (d->isEmpty){
70. label->setString("空");
71. }else{
72. label->setString(d->description);
73. }
74. printDialog();
75. clearLabel(d);\*/
76. showDetails(d);
77. focusGrid = d;
78. });
79. e = grid::create(grid\_e, Node::create(), [this](Ref \*pSender) {
80. /\*if (e->isEmpty){
81. label->setString("空");
82. }else{
83. label->setString(e->description);
84. }
85. printDialog();
86. clearLabel(e);\*/
87. showDetails(e);
88. focusGrid = e;
89. });
90. f = grid::create(grid\_f, Node::create(), [this](Ref \*pSender) {
91. /\*if (f->isEmpty){
92. label->setString("空");
93. }else{
94. label->setString(f->description);
95. }
96. printDialog();
97. clearLabel(f);\*/
98. showDetails(f);
99. focusGrid = f;
100. });
101. a->scheduleString = "aa";
102. b->scheduleString = "bb";
103. c->scheduleString = "cc";
104. d->scheduleString = "dd";
105. e->scheduleString = "ee";
106. f->scheduleString = "ff";
107. grids.pushBack(a);
108. grids.pushBack(b);
109. grids.pushBack(c);
110. grids.pushBack(d);
111. grids.pushBack(e);
112. grids.pushBack(f);
113. itemsMenu = Menu::create(a, b, c, d, e, f, NULL);
114. itemsMenu->setContentSize(Size(this->getContentSize().width, this->getContentSize().height \* 0.75));
115. itemsMenu->alignItemsInColumns(3, 3, NULL);
116. //itemsMenu->alignItemsInRows(2, 2, 2, NULL);
117. itemsMenu->setPosition(Vec2(this->getContentSize().width / 2, this->getContentSize().height - itemsMenu->getContentSize().height / 2));
118. this->addChild(itemsMenu);
119. //scheduleUpdate();
120. return true;
121. }

void Inventory::addItem(Props\* p)

{

for (auto &i : grids)

{

if (!i->isEmpty && i->p->file == p->file)

{

i->getNormalImage()->runAction(Sequence::create(ScaleTo::create(0.25f, 1.25f), RotateBy::create(0.5f, 360), ScaleTo::create(0.25f, 1.0f), NULL));

return;

}

}

for (auto &i : grids)

{

/\*if (i->p->file == p->file)

{

break;

}\*/

if (i->isEmpty)

{

i->addImage(p);

showDetails(i);

focusGrid = i;

//p->setPosition(Point::ZERO);//累死我了，终于发现。2019-9-5 morning

//i->isEmpty = false;

//i->setNormalImage(p);

//i->setName(p->getName());

//i->description = p->getDescription();

////自动显示

//label->setString(i->description);

//printDialog();

//clearLabel(i);

break;

}

}

}

void Inventory::showDetails(grid\* g)

{

std::string details;

if (g->isEmpty)

{

details = "null";

}

else

{

details = g->p->getDescription();

}

label->setString(details);

int index = 0;

while (label->getLetter(index) != nullptr)

{

label->getLetter(index)->stopAllActions();

label->getLetter(index)->setVisible(false);

index++;

}

index = 0;

while (label->getLetter(index) != nullptr)

{

label->getLetter(index)->runAction(

Sequence::create(

DelayTime::create(index \* 0.1f),

Show::create(), nullptr)

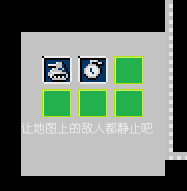
);

index++;

}

}

**效果图：**



**（7）血量控件：**

bool PlayerHP::init(const std::string& source)

{

isTouched = false;

m\_bg = Sprite::create(source);

m\_bg->setOpacity(20);

m\_bg->setPosition(this->getContentSize().width / 2, this->getContentSize().height \* 2/3);

addChild(m\_bg);

auto blood = Sprite::create(source);

blood->setColor(Color3B::YELLOW);

m\_progressBar = ProgressTimer::create(blood);

m\_progressBar->setType(ProgressTimer::Type::BAR);//设置进程条的类型

m\_progressBar->setBarChangeRate(Point(0, 1));//

m\_progressBar->setMidpoint(Point(0, 0));//设置进度的运动方向

m\_progressBar->setPosition(this->getContentSize().width / 2, this->getContentSize().height \* 2/3);

addChild(m\_progressBar);

description = "PlayerHP";

m\_label = Label::createWithTTF(description, "fonts/SIMYOU.TTF", 12);

m\_label->setPosition(this->getContentSize().width / 2, this->getContentSize().height \* 1/4);

addChild(m\_label);

//listenScaling();

return true;

}

**效果图：**



**（8）造路功能相关代码：**

void TankController::listenBuild()

{

keyShift = false;

m\_buildListener->onKeyPressed = [&](EventKeyboard::KeyCode keycode, Event \*event) {

if (keycode == key\_build)

{

if (tank->enableBuildFunction)

{

if (tank->buildType != "")

{

log("build type: %s", tank->buildType.c\_str());

building();

}

else

{

tileSelect();

}

/\*auto listenr = EventListenerKeyboard::create();

listenr->onKeyPressed = [&](EventKeyboard::KeyCode keycode, Event \*event) {\*/

if (keyShift)

{

tileSelect();

}

}

else

{

log("not get");

//弹幕弹出未获得的效果

}

}

if (keycode == EventKeyboard::KeyCode::KEY\_SHIFT)

{

keyShift = true;

}

};

m\_buildListener->onKeyReleased = [&](EventKeyboard::KeyCode keycode, Event \*event) {

if (keycode == key\_build)

{

unschedule("build");

}

if (keycode == EventKeyboard::KeyCode::KEY\_SHIFT)

{

keyShift = false;

}

};

\_eventDispatcher->addEventListenerWithSceneGraphPriority(m\_buildListener, this);

}

void TankController::building()

{

schedule([&](float dt) {

int a = real\_rand\_0\_1() \* 4;

int g1, g2, g3, g4;//每一排的起始gid值

if (tank->buildType == "block.png")

{

g1 = 1, g2 = 29, g3 = 57, g4 = 85;

}

else if (tank->buildType == "steel.png")

{

g1 = 5, g2 = 33, g3 = 61, g4 = 89;

}

else if (tank->buildType == "grass.png")

{

g1 = 9, g2 = 37, g3 = 65, g4 = 93;

}

else if (tank->buildType == "snow.png" || tank->buildType == "river.png")

{

g1 = 13, g2 = 41, g3 = 69, g4 = 97;

}

//选行

int row;

switch (a)

{

case 0:

row = g1;

case 1:

row = g2;

case 2:

row = g3;

case 3:

row = g4;

}

auto p1 = Point(tank->getBoundingBox().getMinX() + 0.000003, tank->getBoundingBox().getMaxY());//左上

auto p2 = Point(tank->getBoundingBox().getMaxX(), tank->getBoundingBox().getMaxY());//右上

auto p3 = Point(tank->getBoundingBox().getMinX() + 0.000003, tank->getBoundingBox().getMinY());//左下

auto p4 = Point(tank->getBoundingBox().getMaxX(), tank->getBoundingBox().getMinY());//右下

//参数0.000003是为了防止左边界超出地图，移动的时候也加了这个参数(bug)

if (tank->m\_direction == DIRECTION::UP)

{

auto position1 = tank->gameLayer->m\_map->tileCoordForPosition(p3);

auto position2 = tank->gameLayer->m\_map->tileCoordForPosition(p4);

for (auto i = position1.x; i <= position2.x; i++)

{

tank->gameLayer->m\_map->layer1->setTileGID(row + a, Vec2(i, position1.y));

}

}

else if (tank->m\_direction == DIRECTION::DOWN)

{

auto position1 = tank->gameLayer->m\_map->tileCoordForPosition(p1);

auto position2 = tank->gameLayer->m\_map->tileCoordForPosition(p2);

for (auto i = position1.x; i <= position2.x; i++)

{

tank->gameLayer->m\_map->layer1->setTileGID(row + a, Vec2(i, position1.y));

}

}

else if (tank->m\_direction == DIRECTION::LEFT)

{

auto position1 = tank->gameLayer->m\_map->tileCoordForPosition(p4);

auto position2 = tank->gameLayer->m\_map->tileCoordForPosition(p2);

for (auto i = position2.y; i <= position1.y; i++)

{

tank->gameLayer->m\_map->layer1->setTileGID(row + a, Vec2(position1.x, i));

}

}

else if (tank->m\_direction == DIRECTION::RIGHT)

{

auto position1 = tank->gameLayer->m\_map->tileCoordForPosition(p3);

auto position2 = tank->gameLayer->m\_map->tileCoordForPosition(p1);

for (auto i = position2.y; i <= position1.y; i++)

{

tank->gameLayer->m\_map->layer1->setTileGID(row + a, Point(position1.x, i));

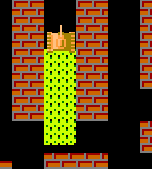
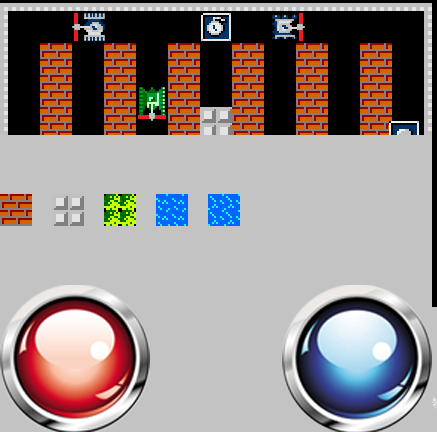
}

}

}, 0.0f, kRepeatForever, 0.0f, "build");

}

**效果图：**



**第三章 游戏的功能测试**

**第一节 测试计划及方案**

本游戏测试以黑盒测试(功能测试)为主，白盒测试在程序代码编写过程中已由3人中辅助编写的另外2人共同完成。

**一、方案计划**

1.本游戏按照功能模块逐一测试。

（1）在vs2017中，测试玩家坦克的移动效率，按键是否灵动，是否有坦克卡出地图的情况；坦克与坦克间的碰撞是否正确。

（2）道具的效果是否得到验证。

（3）敌坦生命<=0时能否引发爆炸效果，音效是否良好。

（4）一直发射子弹是否有卡顿现象，是否没有将其内存释放；子弹与地图、坦克的碰撞是否会产生效果。

2.寻找游戏测试志愿同学，让其在自己VS调试并真实体验，反馈体现建议或意见。

（1）主要方面：对手感和音效是否满意？

（2）对游戏难易程度是否接受？

3.根据问题，制定修改方案，修改原代码。进一步测试。

**第二节 游戏功能测试**

**一、游戏模块测试、集成测试**

1.随机改变敌坦的属性，观察敌坦移动和开火是否正确。观察每个敌坦的随机转向是否会走死胡同。

2.让玩家坦克一直加物品buff，测试buff效果和物品栏状态。

3.当前阶段，各种功能比较简陋。击毁敌机，观察敌机是否自我毁灭及和爆炸效果衔接，爆炸音效播放是否得当。

4.被敌坦击中，我机是否掉血，是否提示游戏结束。记录观察结果。

5.积分积累是否正确，积分达到要求是否正确过关。

**二、游戏整体风格、体验反馈**

1.让参与感受测试的志愿者同学下载试玩此坦克大战游戏，并就测试计划中的问题进行分析。记录分析及感受、建议意见的内容

2.对意见或者建议，进行代码优化修正。回归测试。

**第四章 总结分析**

**第一节 游戏制作总结分析**

**一、写此报告时，游戏存在的问题**

1.没有存档功能

（1）打算后续探究。

2.物品栏最多只能容纳6个，如果连续加6个物品，则后续物品没有效果。

写这份报告时，大部分作业要求完成的地方都已完成，不过存在的问题还在进一步解决。

3.修改后的游戏画面效果粘贴在上面内容里了。

**第二节 我的工作及感想收获**

**一、组内分工说名**

1.我们的信息：

表4.2-1

|  |  |  |
| --- | --- | --- |
| 班级 | 学号 | 姓名 |
| 04051702 | 2017212007 | 韦鑫桥 |
| 04051702 | 2017211990 | 白皓月 |
| 04051702 |  | 王淇锐 |

2.我们的分工说明

组员们一起进行了游戏主题的构想，讨论出此坦克大战单机游戏主题。

我的具体工作：

①主题构想；

②参与讨论游戏板块(此游戏可以完成哪些功能)构想；

③坦克基本行为的编写（移动、开火）；

④编写测试计划，代码编写过程中效果的测试及游戏存在问题的反馈；

⑤报告文档及PPT的编写；

**二、感想及收获**

1.游戏的制作是一个需要细心与经验的工作，比如判断坦克是否与地图上的砖块碰撞，坦克前方要多检测自身速度大小的距离，不然可能下一帧还没来得及检测坦克是否碰撞，坦克已经卡在砖块里了

2.游戏需要根据玩家不同的体验反馈，反复修改，微调。

3.制作过程要求你了解基本C++语法的使用，使我们自发搜集资料更深入学习C++语言，进一步巩固代码编写。我在游戏开发中编写的代码不多，但通过视频资料的学习认识到代码编写能力的薄弱，正在努力弥补中。

4.以前很少玩游戏，现在发现这么一个小游戏的开发也不是很简单的事。有必要在以后玩游戏的同时，思考别人是如何制作这些功能，游戏中又有哪些可以优化完善的地方。

5.熟悉了简单小游戏的大致制作过程。

**结　论**

完成本游戏前，游戏功能的设想（存档），由于时间、期末到来的各种原因，没有来得及全部实现。我们将会在假期的时间里，继续完善，这对于考研或者就业的同学都是一份难得的经验。在开发游戏的工程中，我们渐渐意识到一些知识的重要性，比如关于C++语言知识，也感觉到现有知识应用起来掌握的远远不够。

针对这些，我们也都进行了反思，愿意继续学习，为了使用而学习。以前玩一些游戏会随意评价：怎么界面不好看，这里那里设置不合理之类，自己真正做起来，整个游戏规则设定、界面细节考虑还是有很多讲究，对没有经验的我们，做出来的效果也可想而知，也要求我们掌握一些美学知识。总结部分一些内容与心得类似，不再赘述。

致谢

课程设计完成了，我们每个队员从中都学到了一些东西，比如一些游戏开发的理论知识方面的东西，是课本上涉及或者讲解了的。通过自己动手，我们更了解了课本尚无或者讲解不是很深入的细节问题，所以我们首先感谢自己的能够比较认真地搜集资料，对于之前对做游戏一无所知的队员，了解了游戏开发的的大致流程。其次，我们要感谢我们的队友任振东同学，他在整个过程中最认真，能够一个星期都不打游戏地投入到作业中来。再次是我们的任课老师，他是个温和耐心的人，在课堂上的讲解也容易理解、深入浅出。

本组所有成员

2019.12.22

**参考文献、资料**

1. 慕课网.《Cocos2d-x基础教程二-会动的时钟》、
2. 官方文档、
3. CSDN博客《利用WinRAR的自解压格式将Cocos2d-x项目代码打包成.exe文件》