概要设计说明书

文件状态: [ ] 草稿 [√] 正式发布 [ ] 正在修改	当前版本:	V1.0
	作　　者:	2187小组
	完成日期:	2016年7月14日

修订历史

日期	版本	描述	作者
2016-7-10	0.0.1	初稿	2187 小组
2016-7-10	0.0.2	绪言与项目背景	2187 小组
2016-7-10	0.0.3	硬软件、外部因素、技术技能	2187 小组
2016-7-11	0.0.4	系统体系结构	2187 小组
2016-7-11	0.0.5	模块视图	2187 小组
2016-7-12	0.0.6	技术技能（M）	2187 小组
2016-7-12	0.0.7	领域模型	2187 小组
2016-7-14	0.0.8	数据视图	2187 小组
2016-7-14	0.0.9	补充未鉴别因素和研发风险	2187 小组

目 录

1 引言

1.1 编写目的

编写概要设计说明书的目的主要是用抽象的语言对整个需求进行概括，确定系统的物理配置，确定系统的处理流程和系统的数据结构、接口设计、UI，以及实现对系统的初步设计。另外通过建立目标系统的逻辑模型，也使得软件编程人员对目标系统有一致的认识。

1.2 背景

待开发的软件系统：2187 游戏
项目任务提出者：2187 小组
项目开发者：李祖兴，林小敏，龙俐伶，刘畅，袁晓晖
用户：TA 等项目评定人员、其他用户
实现该软件系统的环境：cocos2dx，vs2012， Android 开发环境

1.3 范围

本说明书适用于游戏的迭代开发阶段。

1.4 定义及缩写

缩写	定义
,2187	2187 是一个类似 2048 的消除类游戏，通过合并相邻的数字，从而得到更大的数字，以此来获得最高分。

1.5 参考文献

书名或文档名	作者
《我所理解的 Cocos2d-x》	秦春林
http://www.cocos.com/article/	Cocos2dx 官方文档
《android 开发从入门到精通》	扶松柏

2 总体设计

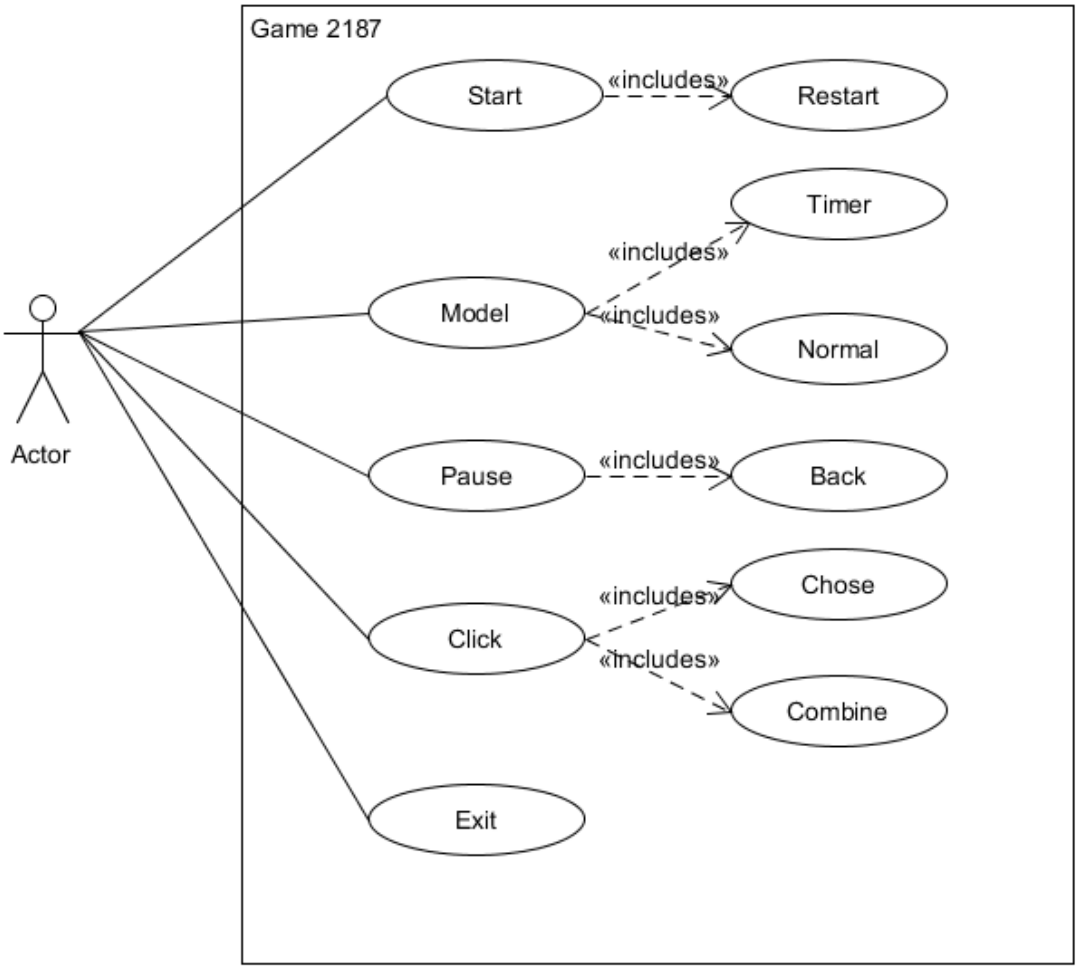
2.1 目标概要

本系统要实现一个基于正三十边形地图的 Cocos2d-x 消除类游戏。通过点击不同的数字，来组合与其相邻并相同的数字，从而获得一个更大值。最后以组合的最大数字为游戏最高分。

2.2 技术选择

我们的项目要实现一个小游戏，首先考虑用 cocosdx 实现。Cocos2d-x 是一个开源的移动 2D 游戏框架，与 vs2013 共同搭建环境，可以很容易地在 windows 和安卓操作系统中建立和运行项目。

2.3 系统功能



2.4 系统性能

没有额外需要加载的图片或数据，几乎不占用内存，可以流畅运行。

3 运行环境

由于系统只是一个简单消除类游戏，所以不存在服务器通信。另外数据也通过 cocos2d-x 的 userdefault 使用 xml 文件存取，不进行 mysql 操作。在平台上，暂定为 windows，不进行 Android 及其他平台的移植。

硬件要求：

主机一台

软件要求：

软/硬件	端	软件	备注
软件要求	客户端	Windows10/8/7	Windows 操作系统
		VS2013	开发软件
		Cocs2d-x	开发软件

配置要求：

主机：

类别	标准配置	最低配置
计算机硬件	CPU: Pentium 4 或以上 内存: 512 MB 或以上 硬盘：5GB 或以上	CPU: Pentium 3 内存: 256 MB 硬盘：1GB
软件	Windows10	Windows7
其他	无	无

4 外部因素

4.1 存在的应用软件

VS2013、Cocos2d-x-3.11.1、Python1.7、jdk1.8、ndk、apache

4.2 第三方应用软件

无

4.3 外部数据源

UserDefault 实现。

5 技术技能

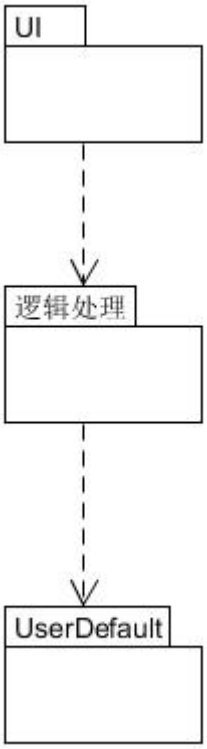
项目管理人员：配置管理、系统分析等相关技术
程序开发人员：C++、Cocos2d-x
测试人员：Junit 等测试技术

6 解决方案架构

6.1 系统体系结构

本开发系统为单用户的体系结构。该体系结构较为简单，整个系统运行在一台计算机上，由一个用户占用全部资源，不同用户之间不共享和交换数据。

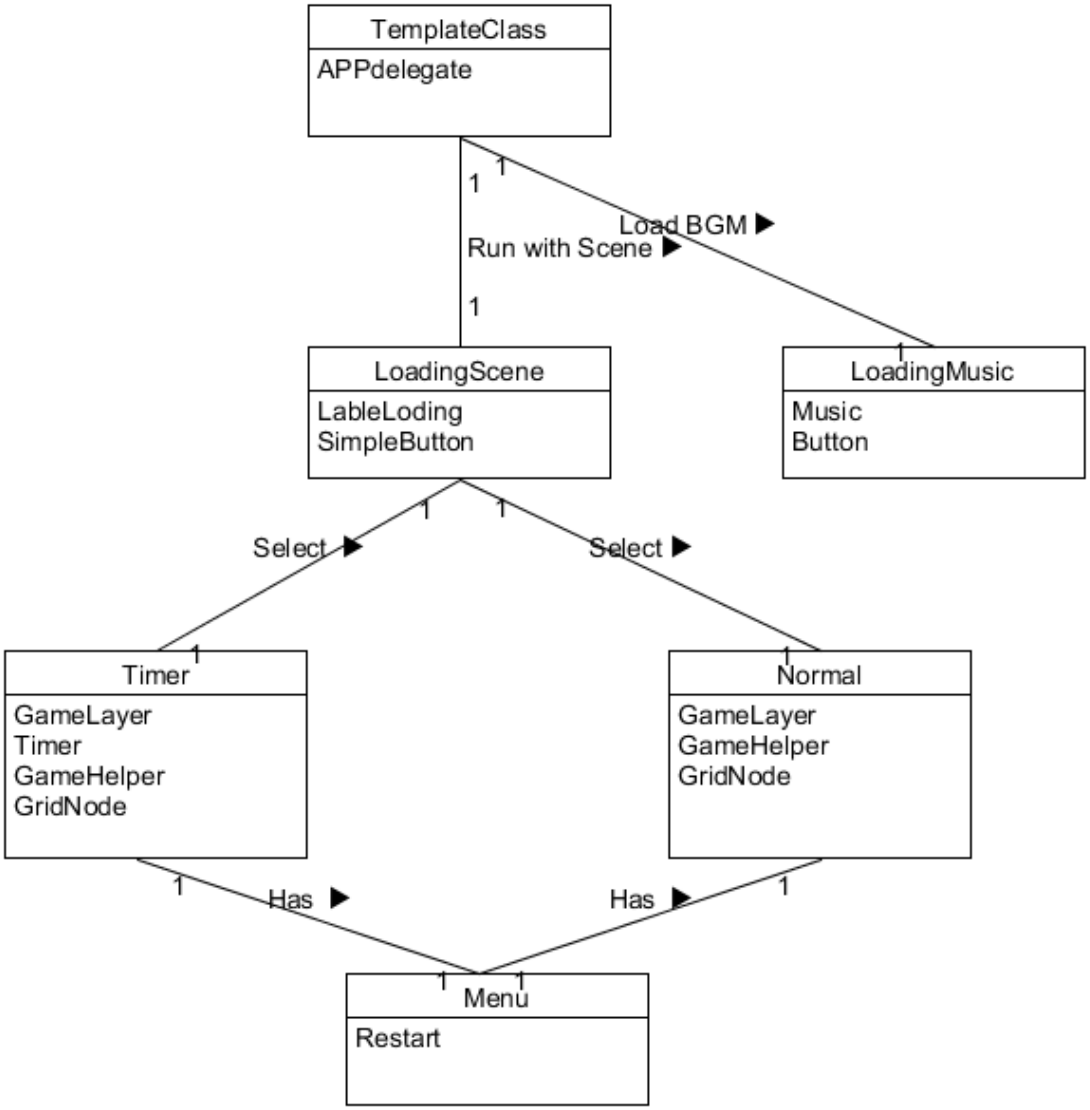
6.2 包图



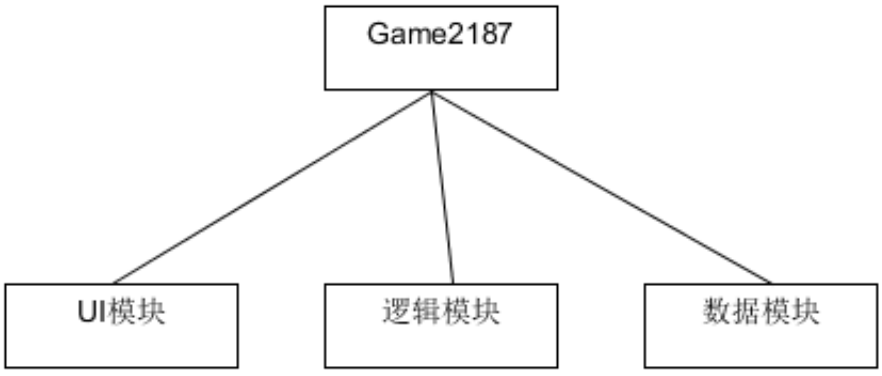
我们采用三层架构(3-tier architecture) 的软件设计方式。通常意义上的三层架构就是将整个业务应用划分为：界面层（User Interface layer）、业务逻辑层（Business Logic Layer）、数据访问层（Data access layer）。区分层次的目的即为了“高内聚低耦合”的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。

距离来说，用户的鼠标点击操作被 UI 层侦听到，随后通过逻辑处理层调用相应的响应方法进行响应，并在 UI 层做出相应反馈，另外，逻辑处理层可能访问 UserDefault 微型数据库进行数据存取。

6.3 领域模型



6.4 模块视图



UI 模块代码：（LoadingScene.cpp，GameScene.app）

```cpp
#include "LoadingScene.h"

bool LoadingScene::init(){
    if(!Layer::init()){
        return false;
    }

    //»-Ò» ö´¿É«µÄ±³¥°
    auto layerColor = LayerColor::create(Color4B(204, 204, 204, 255), VISIBLE_SIZE.width, VISIBLE_SIZE.height);
    layerColor->setPosition(Point::ZERO);
    this->addChild(layerColor);

    //ÆÁÃ»ÓÐ¾ãµÄ¡°loading¡±(FlashÖÐµÄTextField£©
    _lblLoading = Label::createWithSystemFont("Loading...", "ºÚÌå", 20);
    _lblLoading->setPosition(Point(VISIBLE_SIZE.width / 2, VISIBLE_SIZE.height / 2));
    _lblLoading->setTextColor(Color4B(0, 0, 0, 255));
    this->addChild(_lblLoading, 1);

    //Æô¶¯Ò»¸ö0.5ÃëµÄ¼ÆÊ±¨FlashÖÐ¿ÉÒÔÓÃTimerÀ´ÊµÏÖ£©
    this->scheduleOnce(CC_CALLBACK_0(LoadingScene::showStartGameButton, this), 0.5, "showStartGameButton");

    return true;
}

Scene * LoadingScene::createScene(){
    auto scene = Scene::create();
    auto layer = LoadingScene::create();
    scene->addChild(layer);
    return scene;
}

void LoadingScene::showStartGameButton(){
    this->removeChild(_lblLoading);

    auto button = SimpleButton::create(60, "Start", "ºÚÌå", 20, CC_CALLBACK_0(LoadingScene::gotoStartScene, this));
    button->setPosition(VISIBLE_SIZE.width / 2, VISIBLE_SIZE.height / 2);
    this->addChild(button);
}

void LoadingScene::gotoStartScene(){
    //ÇÐ¸ÄÁ£¬´ÓÖÐ±»¬¶¯¼øÈë
    Director::getInstance()->replaceScene(TransitionSlideInR::create(0.5, GameScene::createScene()));
}
```

```cpp
#include "GameScene.h"

GameNodeContainer * container;
GameScoreLayer * scoreLayer;

bool GameScene::init(){
    if(!Layer::init()){
        return false;
    }

    //´¿É«±³¥°
    auto layerColor = LayerColor::create(Color4B(204, 204, 204, 255), VISIBLE_SIZE.width, VISIBLE_SIZE.height);
    layerColor->setPosition(Point::ZERO);
    this->addChild(layerColor);

    //ÓÎÏ·µÄ¸ñ×ÓµÄ±³¥°
    auto bg = GameBackgroundLayer::create();
    bg->setPosition(VISIBLE_SIZE.width / 2, VISIBLE_SIZE.height / 2);
    bg->setScale(1.1f);
    this->addChild(bg);

    //·ÖÊý
    scoreLayer = GameScoreLayer::create();
    scoreLayer->setPosition(Point::ZERO);
    this->addChild(scoreLayer);

    //¸ñ×Ó
    container = GameNodeContainer::create();
    container->setPosition(VISIBLE_SIZE.width / 2, VISIBLE_SIZE.height / 2);
    container->setScoreLayer(scoreLayer);
    this->addChild(container);

    return true;
}

void GameScene::restart(){
    container->restart();
}

void GameScene::gameOver(){
    auto gameOverLayer = GameOverLayer::create();
    gameOverLayer->setPosition(VISIBLE_SIZE.width / 2, VISIBLE_SIZE.height / 2);
    this->addChild(gameOverLayer);
}
```

逻辑模块：（GameNodeContainer.cpp）

```
//¹ã¶ÈÉÑÉ÷ÏàÍ¬Êý×ÖµÄ¸ñ×Ó
Vector<GridNode *> GameNodeContainer::searchSameGrids(GridNode * node){
    resetSearchGrids();
    //³õÊ¼»¯ÉÑÉ÷ÁÐ±í¶ÎÃ¤¹û
    Vector<GridNode *> result;
    searchGrids.clear();
    result.pushBack(node);
    searchGrids.pushBack(node);
    while(!searchGrids.empty()){
        GridNode * checkNode = searchGrids.front();
        searchGrids.eraseObject(checkNode);

        searchedGrids[checkNode->getRow()][checkNode->getCol()] = 1;
        for(int i = 0; i < SIDE_COUNT; i++){
            int row, col;
            if(checkNode->getRow() % 2 == 0){
                row = checkNode->getRow() + extendedInEven[i][0];
                col = checkNode->getCol() + extendedInEven[i][1];
            }else{
                row = checkNode->getRow() + extendedInOdd[i][0];
                col = checkNode->getCol() + extendedInOdd[i][1];
            }

            if(row < 0 || col < 0 || row >= GRID_ROWS || col >= GRID_COLS){
                continue;
            }

            GridNode * extendNode = _listNode[row][col];
            if(!isExceptNode(row, col) && searchedGrids[row][col] == 0 && extendNode->getNum() == node->getNum() && !result.contains(extendNode)){
                result.pushBack(extendNode);
                searchGrids.pushBack(extendNode);
            }
        }
    }

    return result;
}

//ºÏ²¢ÏàÍ¬µÄ¸ñ×ÓÉú³É¸ü´óÒ»µãµÄ¸ñ×Ó¬²¢ÇÒÛ±»ª²¢µÄ¸ñ×Ó¨¦Éú³ÉÐÂµÄ¸ñ×Ó
void GameNodeContainer::composeBiggerGrid(GridNode * targetNode){
    if(_resultGrids.size() < MIN_COMPOSE_COUNT){
        return;
```

```
42  bool GameNodeContainer::onClickGridContainer(Touch * tou, Event * evt){
43      //±éÀúÃ¿Ò»¸ö¸ñ×ÓÉ¬ÅÐ¶ÏÊÇ·ñ»÷ÖÐ
44      for(int row = 0; row < GRID_ROWS; row++){
45          for(int col = 0; col < GRID_COLS; col++){
46              if(isExceptNode(row, col)){
47                  continue;
48              }
49
50              auto node = _listNode[row][col];
51              //ÐèÒª¾«×¼»»³ÉÏ¬Ò»¸ö±êÎµ²ÂÀÙ²ÎÖë¼ÆËã
52              Point locatioInNode = node->convertToNodeSpace(tou->getLocation());
53
54              if(node->hitCheckPoint(locatioInNode)){
55                  switch(node->getCurState()){
56                      case NORMAL:
57                          //µã»÷Ä³¸ö¸ñ×ÓÉ¬À©Õ¹ÏàÍ¬Êý×ÖµÄ¸ñ×Ó
58                          extendSameNumGrid(node);
59                          break;
60                      case SELECTED:
61                          //Ö¤Ç°ÒÑ¾µã»÷¹ýÁË£¬ÔÙ´Îµã»÷¾ÍÒª²¹Ï²¢¸ñ×ÓÁË
62                          composeBiggerGrid(node);
63                          break;
64                      case EXTENDED:
65                          //Ö¤Ç°ÊÇ±»À©Õ¹µÄ¸ñ×Ó£¬ÔÙ´Îµã»÷¾ÍÊÇÀ©Õ¹ÖÜÎ§µÄ¸ñ×Ó
66                          extendSameNumGrid(node);
67                          break;
68                      default:
69                          break;
70                  }
71              }
72          }
73      }
74      return true;
75  }
76
77  //Ñ¡ÖÐÒ»¸ö¸ñ×Ó£¬À©Õ¹µ¼ÖÜÎ§ËùÓÐÏàÍ¬ÊýÖµµÄ¸ñ×Ó
78  void GameNodeContainer::extendSameNumGrid(GridNode * node){
79      clearGridState();
80      node->changeState(SELECTED);
81      _resultGrids = searchSameGrids(node);
82      for(auto resultNode : _resultGrids){
83          if(resultNode != node){
84              resultNode->changeState(GridState::EXTENDED);
85          }
```

数据模块：（Gridnode.cpp,LocolScore.cpp）

```cpp
#include "GridNode.h"


GridNode::GridNode(void)
{
}


GridNode::~GridNode(void)
{
}

bool GridNode::init(){
    if(!Node::init()){
        return false;
    }

    //Á¾¸öÕýÁù±ßÐÎµþÕÚÒ»Æð×é³É±³¾°
    _bg1 = Hexagon::create(GRID_SIDE_LEN, BG_COLOR_1);
    _bg2 = Hexagon::create(GRID_SIDE_LEN, BG_COLOR_EXTENDED);
    this->addChild(_bg1, 2);
    this->addChild(_bg2, 1);

    //ÊÝ×Ö³¿·Ö
    _lblNum = Label::createWithSystemFont(StringUtils::format("%d", _num), "ºÚÌå", 40);
    _lblNum->setTextColor(Color4B(0, 0, 0, 255));
    this->addChild(_lblNum, 3);

    //³öÏÖ¶¯»-£¬ÅýÂý·Å´ó
    this->setScale(0.01f);
    auto scaleTo = ScaleTo::create(0.5f, 1.0f);
    this->runAction(scaleTo);

    changeState(GridState::NORMAL);

    return true;
}

void GridNode::updateNum(int num){
    _num = num;
    refreshView();
}
```

```cpp
#include "LocalStore.h"

static LocalStore * _instance = nullptr;

LocalStore * LocalStore::getInstance(){
    if(!_instance){
        _instance = new LocalStore();
    }

    return _instance;
}

int LocalStore::getHighScore(){
    return UserDefault::getInstance()->getIntegerForKey("highScore");
}

void LocalStore::setHighScore(int score){
    UserDefault::getInstance()->setIntegerForKey("highScore", score);
}
```

6.5 数据视图

6.6 用户界面

用户初始页面为一个 Start 按钮，点击 Start 按钮后切入游戏界面。游戏界面由 20 个六边形组成的三十边形构成，每个六边形为 button 类型，可点击。Button 上面显示数字。可显示当前分数和最高分数。

7 未鉴别因素

7.1 对外部系统的支持

本系统是一个简单的单机游戏系统，没有提供外部接口（如多人对战模块、用户登录模块），可扩展性不足。

7.2 系统安全

本系统不存在安全方面的设计。

8 研发风险

8.1 技术风险

本开发团队成员经验不足，对涉及的项目部分所需要的开发软件和开发技术不熟悉，在研发的过程中可能会遇到不少的开发问题。

8.2 团队管理风险

团队是初次合作，相互之间默契不够。各个团队成员所负责的内容不同，最终可能因接口不规范导致相互之间难以顺利关联。另外，管理人员缺乏相关经验，管理规范化程度较低。