浅谈网络游戏设计

•学生：201806010108项建航

•单位：浙江工业大学计算机科学技术与软件学院/健行学院

•联系方式：[Jh\_zjut@163.com](mailto:Jh_zjut@163.com)

•文章目录

.浅谈网络游戏设计

.关键词

. 1.♦言

. 2.网络游戏介绍及策划构思

* 2.1网络游戏的出现

. 2.2网络游戏的策划构思

. 2.3服务器的设计

* 3.基于cocos2d-x和android的植物大战僵尸开爰
* 3.1功能分析

. 3.2开发环境

. 3.3游戏设计

* 3.4关键代码实现与解析
* 3.4.1图层创建

. 3.4.2游戏，菜单

* 3.4.3战斗场景
* 3.4.4战斗逻辑
* 3.4.5战斗场景的讲席条
* 346 MainActivity

. 4.基于DBN的用户体验模型

. 4.1 DBNUX简单介绍

* 4.2用户体验因子的选择
* 4.3 DBNUX构■流程

. 4.4模型讨论

. 5.关干cocos2d引擎的优化

. 5.1游戏幵爰中的ECS模式（以cocos2d为例）

* 5.1.1问题引入
* 5.1.2 ECS介绍

. 5.1.3 cocos2d-x的缺陷和解决思路

. 5.2摹干ECS的cocos2d-x弓I擎优化方案

* 6,参考资料

摘要

随着人民生活水平的不断提高，大众对精神文化方面的需求越来越大。游戏程序的岀现满足了人们对精神娱乐的需求，此外随着互联网、计算机、移动设备的高速发展，游戏的出现也呈现着渐进性和多样性的特点。本文主要讲述网 络游戏编程的设计思路、策划构思并结合一个基于cocos2d和android的植物大战僵尸案例进行分析研究。对案例的研究主要从游戏策划、游戏设计、代码实现解析和引擎优化等方面展开。

关键词

网络游戏编程、策划构思、COCOS2d. DBNUX用户体验模型、引擎优化

1. 引言

近期阅读了大量关于游戏程序开发的文章和博客，对游戏开发的流程有了更深的了解。通过阅读游戏开发策划和设计的相关文章，本人意识到一个成功的游戏必须要有一个好的策划，好的策划是拥有市场的必要条件，因此本文将谈 论一定的游戏策划和设计。此外，在学了游戏设计开发课程和移动开发课程之后，本人选择植物大战僵尸作为一个实战案例来进行分析，并在文末给出一个用户体验评价模型和一个基于ECS模式的COCOS2d-X引擎优化方案。

（以下仅为本人非正式的体会感想：在我生活的娱乐组成部分中，体验过许多游戏，比如单机游戏、端游、手游、网游等，当然游戏可以划分得更丰富，此处简单介绍，我对游戏的感觉就是玩腻了就换，从以前各种网游到后来 steam.战网这种平台上的游戏，再到现在由于学习更加忙碌而选择偶尔打会简单方便的手游来消息娱乐，大多游戏都有一个特点--时效性，一般游戏很难拥有长期骨灰级玩家，又比如说腾讯通过一定的软件垄断，打造QQ-微信- 支付-娱乐等一体化产品圈，在一定程度上增加了游戏的吸引力，因为我本人就会由于QQ和微信的方便玩腾讯游戏..・）

（MARKDOWN格式会由于编辑软件的不同，显示会有一定差异，因此学生另外上传一份PDF版本文章）

1. 网络游戏介绍及策划构思

2.1网络游戏的出现

近年来，游戏产业正从传统的2D游戏、单机游戏向新兴的3D游戏和在线网络游戏发展.而3D游戏和网络游戏的兴起就伴随着技术和算法的改变.传统的游戏设计模式也需要及时更新，从而适应大众对游戏娱乐的需求.目前看来， 游戏中的界面系统在游戏中占据了重要的地位，一个优秀的UI界面设计可以吸引玩家的兴趣，占据市场.前段时间，本人在许多平台上看到《黑神话：悟空》的宣传视频，该游戏以其高质量的画质渲染和独特风格的界面设计吸引了 大量玩家的关注。一个好的游戏不仅需要考虑其界面等服务对用户的吸引力，也要考虑其技术上的要点，比如和用户的交互性和体验感。对于网络游戏而言，最重要的就是玩家场景的一致性和实时性.一致性就需要建立对应的网 络通信协议，而实时性就要减少网络传书中的延迟误差。网络游戏以互联网为载体，可以通过多种形式来提高用户的游戏体验感，比如说《王者荣耀》以多人团队合作对抗为特点，又比如说《植物大战僵尸》这类游戏虽然几乎没有 直接形式上的团队对抗合作，但是通过互联网设定一个排名榜单来建设一个竞争机制来吸引玩家的兴趣。

2.2网络游戏的策划构思

网络游戏的开发需要好的创意和构思。需求推动生产，创意应该源自于我们日常生活，比如在平时和身边的朋友交流游戏的时候学会捕捉他们思想的闪光点。一个网络游戏是否能够成功，就看它是否能引入一种游戏规则，让玩家一 上手就觉得自己是在做一件有意义有意思的事情。这个规则有很多办法实现。最常用的办法就是引入一个竞争机制，因为只有竞争，才能激起人的奋斗欲望，才能让人把金钱和时间花在上面。这个机制的竞争规则，应该是相对一致 的，即不同的玩家之间竞争的东西应该是一样的，比如，目前的网络游戏大多引入的竞争基本上是等级竞争，装备竞争，为了更高的级别，为了更好的装备，玩家可以不停的作一些枯燥的事。

网络游戏的开发元素有很多，比如说NPC、任务、剧情、交易、装备、宠物等，我们可以通过组合不同的元素来提高玩家的兴奋刺激点和游戏冲动，如NPC+剧情、宠物+装备+交易..・

因此，做一个成功的网络游戏，最重要的是采取合适的竞争机制并通过合适的开发元素组合来提高用户玩游戏的欲望。

2.3服务器的设计

•网络连接

网络游戏服务器需要处理大量的连接，大多数的网络游戏的服务器都会选择^阻塞socket这种结构。对于网络连接，需要用一个结构来储存，其中需要包含一个向客户端写消息的缓冲，还需要一个从客户端读消息的缓冲，具体 的大小根据具体的消息结构来定。

•服务器同步

对于网络游戏来说，如何做好游戏在服务器上的同步是极其重要的。一个简单的做法是把每个玩家的行为向其他玩家广播一遍，为了提高同步的准确性我们可以辅助以不同的解决方案。比如说加入验证同步的机制，简单的说就 是每条指令在经过服务器验证通过了以后再执行动作，并将服务器的反馈信息广播给一定范围内的玩家，当然这个范围是经过逻辑计算得出的。

• NPC智能

NPC智能分为两种，一种是被动触发的事件，一种是主动触发的事件。对于被动触发的事件，处理起来相对来说简单一些，可以由事件本身来呼叫NPC身上的函数，比如说NPC的死亡，实际上是在NPC的HP小于一定值的时 候，来主动呼叫NPC身上的OnDieO函数，这种由事件来触发NPC行为的NPC智能，可称为被动触发。

显然仅有被动触发的NPC并不智能，那我们该怎么提高NPC的智能性呢？以我本人的认知深度，随机函数和计时器可以帮助我们提高NPC的智能性，通过设定一定的条件让对象在一定的时间主动去进行特定事件。

♦服务器内存管理

对于服务器内存管理我们将采用内存池的方法，也称为静态内存管理。其概念为在服务器初始化的时候，申请一块非常大的内存，称为内存池，同时也申请一小块内存空间，称为垃圾回收站。其大体思路如下：当程序需要申请 内存的时候，首先检查垃圾回收站是否为空，如果不为空的话，则从垃圾回收站中找一块可用的内存地址，在内存池中根据地址找到相应的空间，分配给程序用，如果垃圾回收站是空的话，则直接从内存池的当前指针位置申请 一块内存；当程序释放空间的时候，给那块内存打上已经释放掉的标记，然后把那块内存的地址放入垃圾回收站。

1. 基于cocos2d-x和android的植物大战僵尸开发

植物大战僵尸这款游戏有许多版本，其中市场主要存在的是电脑端、手机端的《植物大战僵尸1》和《植物大战僵尸2》，恰巧本人对上述版本都有过游戏体验和了解，因此下文将结合手机端的《植物大战僵尸1》进行开发和分析。 植物大战僵尸在游戏策划上主要由植物角色、僵尸角色、植物购买机制、闯关成就机制和世界排名机制等元素组成，在一定程度上吸引了玩家的兴趣，具有一定的市场。

3.1功能分析

本游戏项目大概有如下功能需求：游戏角色、菜单页面、地图移动、展现僵尸/植物、选择植物框、已选植物框、选择/取消选择、开始战斗、'’准备-好-开始“动画、加载僵尸、安放植物、僵尸攻击植物、植物攻击僵尸、游戏的进度 展示、太阳花产生太阳、声音播放、收集太阳…

3-2开发环境

•系统层面

型号名称：

处理器名称： 处理器速度： 处理器数目： 核总数：

L2缓存（每个核）:

L3缓存： 超线程技术： 内存：32 GB

MacBook Pro 八核工ntel Core i9 2.3 GHz

256 KB

16 MB 已启用

•应用层面

Android Studio 4.1.2 cocos2d-x 相关jar■包

3.3游戏设计

•设计思路

游戏的设计运用了cocos2d-android的CCGL-SurfaceView的主体功能实现，先建立一个场景，再在场景上添加图层，随后在图层上添加精灵，最后由导演指挥、运行。

本游戏事先会准备很多图片资源存储在asset文件夹中，植物坐标以二维数组的存储方式保存在特定集合中，僵尸坐标也保存于特定集合中，当游戏开始时，僵尸会从保存的坐标点中随机产生出来；而植物的产生则由玩家安放。 游戏的主体界面上有类似的植物安放位置地图，是以屏幕位置划分的，用户只要单击了植物，再点击屏幕的设置范围、即可做出相应的晌应。

游戏主要包括3个界面，游戏开始界面、游戏选择界面、游戏主体界面。游戏的界面流程图如下所示。所有非游戏界面釆用了贴图或者菜单选项等显示机制，并对这些贴图或者菜单设有按钮点击事件，当用户点击时，会有相应 的提示。游戏主体界面是游戏设计中游戏对象最多，需要处理的游戏线程处理最复杂的界面。

•界面流程图

欢迎界面

点击开始游戏

主菜单界面

退出游戏

选择按钮

END



3.4关键代码实现与解析

3.4.1图层创建

游戏界面

选择关卡

模式选择界面

游戏帮助

返回

帮助界面

暂停游戏

暂停游戏

暂停游戏

牌界面

•实现思路：

这里新建一个名为layer的包，然后在该包里新建一个BaseLayer,作为所有图层的基类，在该类中通过CCDirector.sharedDirector（）.winSize（）«获取屏幕的宽高。

在Layer包下新建一个WelcomeLayer,用于作为欢迎界面的图层，继承BaseLayero然后编写intitLogoO方法、showLogoO方法、showWelcome。方法，分别实现相应的逻辑。 •代码实现：

\*基类图层

public class BaseLayer extends CCLayer {

/\*\*

\*屏幕的宽高

public CGSize size = CCDirector.sharedDirector().winSize();

public BaseLayer() {

/\*\*

\*欢迎页面图层

public class WelcomeLayer extends BaseLayer {

/\*\*

\* Log。对象

private CCSprite logo;

public WelcomeLayer() { initLogo (); / / 初始化丄。go showLogo(); // 展示logo

/\*\*

\* ].logo?®始化

private void initLogo(){

logo = CCSprite ・ sprite (" image/popcap\_\_logo • png")；

size.height / 2); // 屏幕居中

logo.setPosition(size.width / 2 , this.addChild(logo);

/\*\*

\* 2 .展示 Logo

private void showLogo()(

CCHide hide = CCHide.action(); // 表示隐藏动作

CCDelayTime delay = CCDelayTime. action(1); // 延时一秒钟

CCShow show = CCShow.action(); // 表示显示动作

CCSequence sequence = CCSequence.actions (hide,delay,show,delay,hiderdelay/CCCallFunc.action(thisz nshowWelcome")); // —上来先隐藏Logo,过一秒后显示logo,再过一秒钟后继续 隐藏logo

logo・ runAction(sequence);

/\*\*

\* 3 .显示欢迎界面。

public void showWelcome()(

logo.removeSelf (); // 删除log。

CCSprite welcome = CCSprite. sprite (" image/welcome - jpg"); // 初始化欢迎界面 welcome・setAnchorPoint(0.0); // 设置锚点为左下角 this.addChild(welcome);

3.4.2游戏主菜单

•实现思路：

在Layer包下新建游戏主菜单与层MenuLayer,添加showMainMenuO方法，后续只需在图层相应按钮上添加响应事件。

•代码实现：

\*菜单图层

public class MenuLayer extends BaseLayer{ public MenuLayer() { showMainMenu();

/\*\*

\* 1 .初始化主菜单页面

\*/

private void sho^lainMenuf){

CCSprite background = CCSprite.sprite("image/menu/main^\_menu\_bg. jpg"); background・ setAnchorPoint(0,0);

this.addChild(background);

3.4.3战斗场景

•实现思路：

在layer包下新建FightLayer,用于表示战斗场景的图层，并在loadMapO方法中加载地图；该图层中实现了加载地图、加载僵尸、移动地图、加载植物和植物选择的功能。

•代码实现：

\*战斗图层

\*/

public class FightLayer extends BaseLayer{ public FightLayer() {

loadMap();

/\*\*

\* 1 .加载地图

private void loadMap(){

CCTMXTiledMap map = CCTMXTiledMap.tiledMap("image/fight/map\_\_day.tmx"); this .addChild(map);

/\*\*

* 2 .加载僵尸的坐标点集合
* @return

private 2UrrayList<CGPoint> loadZombiePoint(){

ArrayList<CGPoint> tempPoints = new ArrayListo();

CCTMXObj ectGroup zombies = map.obj ectGroupNamed("zombies"); ArrayList<HashMap<String, String» zonbiesPoint = zraibies.objects; for (HashMap<String, String> map : zonbiesPoint) {

Integer x = Integer.parselnt(map.get("x"));

= Integer.parselnt(map.get("y"));

Integer y

ten?>Points - add (ccp (x, y));

retum tempPoints;

3 .移动地图

\*/

private void moveMap() (

=size.width - map.getContentSize().width; // 地图移动的偏移■

float offset

CCDelayTime delay = CCDelayTime. action(1); // 延时 1 秒

CCMoveBy move = CCMoveBy.action(*2,*ccp(offsetz0));

CCSequence sequence = CCSequence.actions(delay,move);

map.runAction(sequence);

\* 4 .展示植物框

public void showPlantBox(){ showSelectedBox(); showChooseBox();

\* 5 .展示植物选择框（已选）

private void showSelectedBox(){

mSelectedBox = CCSprite.sprite(" image/fight/chose/fight\_\_chose.png");

mSelectedBox.setAnchorPoint(0,1); // 设置锚点为左上角

mSelectedBox.setPosition(0,size.height);

this.addChild(mSelectedBox);

/\*\*

\* 6 .展示植物选择框（未选）

private void showChooseBox(){ mChooseBox =

CCSprite. sprite("image/fight/chose/fight\_\_choose.png"); mChooseBox・setAnchorPoint(0,0); // 设置锚点为左下角 this.addChild(mChooseBox);

String format = "image/fight/chose/choose\_default%02d.pngw; for (int i = 1; i <= 9 ; i++) {

CCSprite plant = CCSprite・sprite(String.format(formatfi)); float x = (i - 1) % 4 \* 54 + 16; // 计算x坐标

float y = 175 - (i - 1) / 4 \* 59; // 计算y坐标 plant.setAnchorPoint(0z 0); // 设置锚点为左下角 plant.setPosition(x,y);

mChooseBox.addChild(plant);

3.4.4战斗逻辑

•实现思路：

新建一个engine包在包下创建GameEngine类，井且将该类设置为单例模式.

修改FightLineEngine,添加attackPlantO方法表示僵尸和植物处在同一个x轴上时僵尸开始攻击植物. •代码实现：

\*处理战斗逻辑的引擎

\*单例类

public class GameEngine {

\*单例对象

private static GameEngine mlnstance = new GameEngine ();

/\*\*

\*游戏地图

\*/

private CCTMXTiledMap map;

/\*\*

\*已选植物

private CopyOnWriteArrayList<Plant> mSelectedPlants;

/\*\*

\*标记游戏是否正式开始

public static boolean isStart;

public GameEngine() {

}

public static GameEngine getlnstance(){ return mlnstance;

/\*\*

* 1 .游戏开始的一些处理
* @ param map游戏地图
* @param selectedPlants

已选植物

public void gamestart(CCTMXTiledMap map,CopyOnWriteArrayList<Plant> selectedPlants)( isStart = true;

this .map = map;

this.mSelectedPlants = selectedPlants;

/\*\*

* 2 .专门处理点击事件
* 0 param event

public void handleTouch(MotionEvent event){

*\*\**

\*封装战线的引擎类

public class FightLineEngine {

/\*\*

\*保存植物对象的Map

\*/

private HashMap<IntegerzPlant> mPlants = new HashMap<>(); // key表示植物在第几歹U

/\*\*

\*保存僵尸对象的集合

private CopyOnWriteArrayList<ZOTibie> mZombies = new CopyOnWriteArrayList<>();

public FightLineEngine(int i) {

CCScheduler scheduler = CCScheduler.sharedScheduler()；

scheduler. schedule (" attackplant",thisz 0.2f, false); // 每隔0.2秒检测僵尸是否可以攻击植物

/\*\*

* 1 .添加植物
* @ param plant植物对象

\*/

public void addPlant(Plant plant){ mPlants-put(plant.getColumn()rplant);

/\*\*

* 2 .判断战线上是否已经有植物，有的话就不能再安放了
* @return

public boolean contaionsPlant(Plant plant){

return mPlants.keyset().contains(plant・ getColumn());

* 3 .添加僵尸
* @ param zombie 僵尸对象

public void addZombie(final Zombie zraibie){

//僵尸的死亡回调

zombie・ setDieListener(new BaseElement-DieListener() { ^Override

public void die() {

mZombies.remove(zombie); //僵尸死亡后从集合中移除

})； mZombies. add (zcanbie);

/\*\*

* 4 .僵尸攻击植物
* @param f

public void attackplant(float f)(

if (ImPlants.isEn^ty() && !mZombies.isErrpty ()) { for (Zombie zombie : mZombies) {

int column = (int) (zombie.getPosition().x / 46 - 1);

if (mPlants. key Set () .contains (column)) { //僵尸当前所在的列上有植物存在 if (!zombie.isAttacking()){

zombie・attack(mPlants.get(column)); //表示僵尸开始攻击该歹!J的植物 zombie.setAttacking( true); // 标记正在攻击

3.4.5战斗场景的进度条

•实现思路：

修改GameEngine,添加progress。方法，代表进度器的逻辑，并修改gameStart（）方法，在游戏开始时显示进度条，再修改loadZombiesO方法，让每加载一只僵尸就让进度+5.

・代码实现：

/\*\*

\*定义进度器

private CCProgressTimer progressTimer; /\*\*

\*僵尸加载的进度

private int progress;

/\*\*

* 1.游戏开始的一些处理
* @ param map游戏地图
* @param selectedPlants 已选植物

public void gameStart(CCTMXTiledMap map,CopyOnWriteArrayList<Plant> selectedPlants){ isStart = true;

this.map = map;

this.mSelectedPlants = selectedPlants;

mZombiePoints = ConmonUtil.loadPoint(map, "road");

//定时器

CCScheduler scheduler =

CCScheduler.sharedScheduler();

scheduler• schedule("loadzombie",this,*2,*false); 〃每隔两秒执行一次lo&dZombie()方法 loadPlant();

progress (); //在游戏开始时显示进度条

/\*\*

* 3.加载僵尸
* gparam f必须有的参数，不然CCScheduler无法通过反射调用

public void loadZombie(float f) (

Randcxn random = new Random ();

int line = random.nextlnt(5); // 随机数为0,1,2,3,4

CGPoint startPoint = mZombiePoints.get(line \* 2); // 起点坐标 CGPoint endPoint = mZombiePoints.get(line \*2 + 1); // 终点坐标 PrimaryZcanbie zombie = new PrimaryZombie(startPoint,endPoint); map.addChild(zombie,1);

mFightLines.get(line) • addZombie(zombie); // 把僵尸添加到战线中 progress +=

5; //每加载一个僵尸，让加载进度+5

progressTimer. setPercentage(progress); // 更条

}

/\*\*

• 6 .进度条

private void progress() {

progressTimer = CCProgressTimer.progressWithFile("image/fight/progress.png");

progressTimer・setPosition(CCDirector.sharedDirector()・getWinSize().width - 70, 13); map, getParent(). addChild(progressTimer);

progre s s Timer.setScale(0-6f);

// 0-100 progressTimer. setPercentage(0); //每增加一个僵尸需要调整进度，增加5 0-100 //设置样式

progressTimer.setType(CCProgressTimer.kCCProgressTiraerTypeHorizontalBarLR);

CCSprite sprite = CCSprite. sprite("image/fight/flagmeter・ png");

sprite.setPosition(CCDirector.sharedDirector().getWinSize().width - 70, 13); map, getParent(). addChild(sprite);

sprite・ setScale(0.6f);

CCSprite name = CCSprite.sprite(Mimage/fight/FlagMeterLevelProgress.png"); name.setPosition(CCDirector・sharedDirector()•getWinSize().width 一 70, 5); map.getParent().addChild(name);

name.setScale(0.6f);

3.4.6 MainActivity

* ssasss：

SBRSfiS5anclraicllSBMaa«faEWActivitvB0ot. »®?cocos2d-xl»«»AlIS«J=»iSffi«HS«BSil®«±.

* ft码夹现：

public class MainActivity extends AppCompatActivity {

/\*\*

\*导演

CCDirector director = CCDirector.sharedDirector();

@Override

protected void onCreate(Bundle savedlnstanceState) { super, onCreate(savedlnstanceState);

〃调用顺序：视图(cCGLSurfaceView) \_》导演(CCDirector) 场景(ccscene) \_》图层(CCLayer) ■》精灵(CCSprite) \_》动作(cCMove)

//获取视图

CCGLSurfaceview view = new CCGLSurfaceView(this); // 创建—^SurfaceView,类似导演眼前的小屏幕，必须传this,底层要强转成Actvity setcontentview(view);

//获取导演的单例对象

director.attachlnview（view）; // 开启绘制线程的方法

director.setDisplayFPS（true）; //显示帧率，表示每秒刷新页面的次数。一般当帧率大于30帧时，基本上人眼看起来比较流畅，帧率和手机性能与程序性能有关 director.setAnimationinterval（l/60f）; // 设置最高帧率位60

director.setDeviceOrientation（CCDirector.kCCDeviceOrientationLandscapeLeft）; // 设置屏幕方式为显示

director. setScreenSize（480,320）; //设置分辨率，用于屏幕适配，会基于不同大小的屏幕等比例缩放，设置我们开发时候的分辨率

//获取场景对象

CCScene scene = CCScene.node();

//获取图层对象

//FirstLayer layer = new FirstLayer(); //ActionLayer layer = new ActionLayer(); DemoLayer layer = new DemoLayer();

//配置环境

scene.addChild(layer); // 给场景添加图层 director.runWithScene(scene); // 导演运行场景

0Override protected void onResume() { super, onResume(); director .resume (); // 游戏继续 SoundEngine. sharedEngine (). resumeSound (); *II* 音乐

@Override protected void onPause() { super.onPause(); director . pause (); // 游戏暂停 SoundEngine. sharedEngine()・pauseSound(); // 音乐暂停

@Override protected void onDestroy() { super.onDestroy(); director.end(); // 游戏结束

4.基于DBN的用户体验模型

4.1 DBNUX简单介绍

在一个网络游戏代码开发结束之后，显然我们需要进行各种测试，此处我想介绍的是对用户游戏体验研究的一个模型。

为了更直观、有效地评估游戏产品的用户体验，消除单一评估标准的不确定性。方法从传统的MDA游戏设计的角度出发，引入用户的生理特征测量，构建基于动态贝叶斯网络的用户体验评估模型。该模型通过MDAUX框架提取用户体 验影响因子，作为贝叶斯网络的输入层节点，通过生理特征测量方法提取用户的脑电和眼动状态，作为贝叶斯网络输出层节点，以一阶隐马尔可夫模型表示两个相邻时间片上用户体验元素的影响关系，从而动态地展示用户体验状 态。

4.2用户体验因子的选择

为明确游戏设计流程，提取用户体验影响因子，基于游戏幵发的MDA框架理论，将游戏产品的技术要求（如数据交换、网络环境等）整合至设计要求中，即功能机制、动态调节和美学需求，建立基于MDAMMDAUX框架。MDAUX框架 大致分为3层：功能层、结构和表现层。功能层的游戏机制是指游戏本身的运行机制、奖励机制等；结构层指动态调节的游戏结构，包括交互设计和信息架构，交互设计即关于游戏的可操作部分的设计，信息架构指游戏中的角色行 为、战斗调节（游戏中的人工智能）等；表现层指游戏设计的美学需求，包括玩家直观感受到的视觉设计和听觉设计。

4.3 DBNUX构建流程

为了建立基于生理特征测量的离散DBN用户体3佥模型，第一步是指定离散DBN的节点，即指定用于构建离散DBN网络的输入层变量、隐藏变量和输出层变量；第二步是确定离散变量的数值类型；第三步是配置变量的初始状 态，即在时间t=l时计算SBN；最后一步是计算随时间变化的条件概率并推断游戏玩家的用户体验。一个DBN可以被认为是连续时间片相互连接的SBN的集合，并且两个相邻时间片之间的关系由HMM建模。

模型构建流程大致如下图：

MM i

:

:tWVMW

,ass,

»«•

I

4.4模型讨论

在本文引入DBNUX模型的主要目的是为了提供一个测试用户体验的模型工具，本人对该模型就不详细介绍了。模型可以以一阶HMM表示相邻时间片的影响关系，通过收集用户脑电和眼动信息，推测出用户当前使用状态，消除单一 指标预测的不稳定性，将用户体验研究数理模型化，对游戏产品用户体验的高效评估具有参考价值。设计者基于证据变量可推断出当前用户的体验状态，并根据概率分布情况来合理安排各设计模块的投入资源。

在一个游戏设计开发完成之后必然要进行用户体验的测试，通过这样的一个数学模型可以定量地分析用户的体验感，并可以在一定程度上预测其他玩家的游戏体验感。

5.关于cocos2d引擎的优化

5.1游戏开发中的ECS模式（以cocos2d为例）

5.1.1问题引入

通常在使用继承来进行游戏开发编程时会这样带来的问题：游戏对象的逻辑往往在超类中被定死了，而一旦需要对逻辑作出修改，要么重写实现，要么继承基类进行覆盖。此外，在C++中使用对象池优化时就会造成灾难性的后果: 一种类型一个池（尽管可以用通用的内存分配器，但是这样还要考虑内存碎片等杂七杂八的问题）。对于传统的设计思路，在游戏开发上就会导致〃类灾难"。于是ECS模式被提了出来，用于解决继承带来的问题。

5.1.2 ECS介绍

ECS模式类似与软件设计模式中的组件模式，为了解决继承导致的开发问题，我们可以采取组合的形式来替代。ECS,即Entity-Component-System （实体-组件-系统）的缩写，其模式遵循组合优于继承原则，游戏内的每一 个基本单元都是一个实体，每个实体又由一个或多个组件构成，每个组件仅仅包含代表其特性的数据（即在组件中没有任何方法），例如：移动相关的组件Movecomponent包含速度、位置、朝向等属性，一旦一个实体拥有了 网5^€：。叫。11611七组件便可以认为它拥有了移动的能力，系统便是来处理拥有一个或多个相同组件的实体集合的工具，其只拥有行为（即在系统中没有任何数据），在这个例子中，处理移动的系统仅仅关心拥有移动能力的实体，它 会遍历所有拥有Movecomponent组件的实体，并根据相关的数据（速度、位置、朝向等），更新实体的位置。实体与组件是一个一对多的关系，实体拥有怎样的能力，完全是取决于其拥有哪些组件，通过动态添加或删除组件，可 以在（游戏）运行时改变实体的行为。

5.1.3 cocos2d-x的缺陷和解决思路

1. aefS Cocos2d-X WSWBSS\*,改掘号避机之冋的相亘03|使得开发者SSM

St抿利行为ft■为 ECS SBffKSSSE,

1. 当开发交亘史时住蔓#枚高旳漩我时，CocosSd-X 北出现卡顿等性熊问息’Wilt,

m+w/i«s«<t>wis cocos2d-x 由改据族与政本，awMttse.

5.2基于ECS的cocos2d-x引擎优化方案

St用于模世Node 时象以及衣击其内卽各WtWSaaSHE现在都被视为saw国中分SI以 NodeComponent« DataComponent MS）, ECS ID）\*«a.

（2）Node ECS 中旳糸城（Systorrt\*!（执行.速車、LoglcSystem Kni）；第二美

MNodeComponent 中取待 Cocos2d-X 司孽中StBIS Node JB\*以売成国后胸象啾击更Sti工作（以 UpdatoSystem 采用#Mt方禀前后 CooosSd-X

游戏对象构建 游戏对象模拟 游戏对象状态更新

^^Node

Nog対家

*MS*

♦方救点：

以上星于ECS架的化方禀貝希以下仇点,

Si的星于外台旳时象杓建方\*储为代鰻带来更好85可扩展住.Cooos2d-X RllWJffl K：s架构棲丈,卖住!SSB件也集合旳！（》性，

件域台方夷，或呈定件美型，由于浙我时象木身只呈一个标符id,问且妙件wass有任何行为，极大液少了并多可象之前复未触域京成031关糸旳由现.

Node 可家内簞"移到了不同荚璽的纸件中.aMWjfflWRasBSWH,#., ss®SBftSitBi»is«s<>i\*n»«HSss,

（3>Cocos2d-X触濾■哥家更航现在由员责如理漩■世界中不同遮机旳糸SKSystom,完成，它”郭只该兮指是荚型“如件，S-i'SaTeSSSKKlSiaBflffl,減少了为卖规S8行时务表而付世的姓熊ft忻.另久，

若熊使sb件在内存中鹵分布符合楊窟扃wit旳蔓#.便能以据升源戏還行si和中如计算■。命中\*如方式\*-步据升流浄性熊.

Cocos2d-X KMHS Scheduler,刘波戏世界中85亲袖避絞注册一个更如$时家相关旳布刘道極、而不是SHMBBSS楽个漱戏对家注册更标回诃.开发者只需关，|MS崖宙机行优

6.参考资料

[1W!K».»?oooos2d-androldWa\*\*aair"»W»[J]«»,2018,39(08):174-177.

(2)SISi^,«i\*,SSfria,IMTCocos2d-x5l«ai|BaWSi+«BffltJlSftlg?a\*,2018,41(24):106-109\*113.

[3BUHt»,Me,M.Unity?IBTSA6aRi«SWm®i+\*5f8[J].i+W»l«Mffl.2020,29(05):103-109.

W莎可，统\*庆JlMSiStW 星于。BN 旳)B 户俐a>KS 樫 SWf 瓠n.包装:M,2020,41(02\*231-236+248.

[5)##tt.MTECSKCooos2d-X?l»ft<t®i+\*«MD],S»\*\*,2019.

[Sllovoo. cocos2dttW^«r^fi?«T[Z/OL]. CSDN.2016.

[41DXT00.瘀开发,“BCS槿就Z/0L1. CSON, 2019.

[SieSEwn. \*WS>ti«<IECS«»[Z/OlJ. CSDN, 2020.