1. Cocos2d-X引擎及存在的问题

[2.1 Cocos2D-x游戏引擎 5](#_Toc7937)

[2.1.1 Coscos2d-x游戏引擎介绍 5](#_Toc31447)

[2.1.2 Coscos2d-x的结构 6](#_Toc10406)

[2.1.3 Cocos2d-x工作流 7](#_Toc32436)

2.1.4 Cocos2d-X U UI系统

2.2 引擎存在的问题

2.3 相关需求分析和优化目标

[第3章 引擎的优化与实现 5](#_Toc5892)

[3.1 渲染模块的优化 8](#_Toc13259)

[2.3.1 游戏循环时间同步方式 9](#_Toc11693)

[2.3.2 几种多线程渲染方式 11](#_Toc30659)

[2.3.3 Cocos2D-x的渲染模块 13](#_Toc10432)

[2.3.4 改良后渲染模块 14](#_Toc1210)

[3.2 场景更新模块的优化 17](#_Toc3559)

[2.4.1 分层状态机介绍 19](#_Toc7489)

[2.4.2 Cocos2D-x的场景更新模块 20](#_Toc10245)

[2.4.3 改良后场景更新模块 20](#_Toc28918)

[3.3 事件分发模块的优化 21](#_Toc24413)

[2.5.1 Cocos2D-x的事件分发模块 21](#_Toc27047)

[2.5.2 事件分发模块优化 22](#_Toc30391)

[第4章 针对卡牌类游戏的UI系统的设计与实现 23](#_Toc1644)

[4.1 UI系统总体设计 28](#_Toc27382)

[4.2 配置文件管理模块设计与实现 29](#_Toc5784)

[4.3 核心模块的设计与实现 32](#_Toc6099)

[4.4 系统级控件的设计与实现 34](#_Toc26092)

[3.6.1 Image 34](#_Toc11186)

[3.6.2 Button 35](#_Toc6622)

[3.6.3 Label 36](#_Toc25036)

[3.6.4 Toggle 37](#_Toc8158)

[3.6.5 ScaleImage 38](#_Toc31877)

[3.6.6 InputEdit 38](#_Toc22695)

[4.5 游戏级控件的设计与实现 39](#_Toc2542)

[3.7.1 菜单控件（ListEntity) 39](#_Toc4417)

[3.7.2 面板跳转实体(LayerSelecter) 42](#_Toc9536)

[3.7.3 列表切换按钮(ListButton) 42](#_Toc14604)

[3.7.4 倒计时器(TimerEntity) 44](#_Toc25917)

[4.6 UI编辑器 44](#_Toc25882)

[3.8.1 UI编辑器的需求 45](#_Toc26273)

[3.8.2 UI编辑器的完成效果 45](#_Toc29541)

在分析了引擎原版UI系统的局限性和手游UI系统需求之后，我们为新版UI系统提出了以下的需求:

1. UI系统对游戏UI配置文件的管理

在开发过程中，UI编辑器和游戏属于两个不同的程序，需要设计一套配置文件以便两个程序都可以自由的加载或编辑。客户端读取配置文件以显示特定界面的UI控件。UI编译器读取该文件后，可以显示该配置文件在客户端中加载后的效果，并且在修改配置文件后可以实时显示在编辑器中。使用配置文件配置UI的方法可以减少我们重编译客户端代码的次数，如果我们修改了UI界面配置文件，只需要重新加载该界面即可。

在游戏的不同平台中，我们同时需要构建一套配置文件加载系统，可以在游戏开始时加载所有的配置文件，在读取到某一配置文件时，再去特定位置加载该配置文件所指定的资源。

2.UI系统可以更好的和屏幕适配

在手游的UI系统中，屏幕适配是一个很重要的问题。不同厂家的机型有着不同的分辨率和屏幕尺寸。在游戏一类的应用中，图片资源的大小占整个包体很大的比重，如果针对不同分辨率配置多套资源，势必增加客户端包体大小以及游戏整体开发难度。所以对于游戏一类的应用，通常采用只维护一套资源，并在UI系统中附加适当的屏幕适配控制的方法。

3.UI控件的管理

一个UI系统最重要的就是对控件的管理。其中包括UI加载，UI显示，UI释放等功能。针对不同UI控件，还需要对他们的属性进行配置。比如说调整位置，显示和隐藏，配置响应函数，比例等信息。甚至可以预先根据控件想要表现的功能挂载一些逻辑代码甚至shader。在UI编辑器中通过修改配置文件的方式设置是否在客户端中触发这段逻辑代码或者shader。

在UI管理系统中，我们也需要重新构建个UI树用来管理这些UI控件，每一个UI控件都是一个节点，与Cocos2D-x原先的UI渲染树同步更新，保证UI控件的遍历不会在中途出现问题。UI控件加载操作流程需要做到最简单，涉及到配置文件的地方，只需要一个API统一加载，其他工作全由UI系统自己完成。

4.系统级UI控件的实现

针对手机游戏，我们在Cocos2D-x原有控件的基础上，重新开发一套新的UI控件。这套控件为游戏底层控件，需要满足Cocos2D-x所能实现的各方面功能。

为此，我们设计了如表3.1所示的几种基本的系统控件。

|  |  |  |
| --- | --- | --- |
| 类名 | 控件名 | 功能描述 |
| Image | 静态图片 | 以一张静态图片方式显示 |
| Button | 按钮 | 游戏中的按钮实体，点击按钮可以触发相应的工作，包括三种状态。未触摸，触摸开始，触摸结束，针对这三种状态可以显示不同的静态图片或文字 |
| Label | 文字 | 游戏中的字体控件，封装了Cocos2D-X的字体控件，在配置文件中可以调整不同的配置，达到显示不同信息的目的 |
| Toggle | 按钮触发器 | 由多个按钮组成，在不同按钮状态之间可以进行切换 |
| ScaleImage | 九宫格图片 | 九宫格图片控件，减少因图片过渡拉伸而造成毛边的可能性。适合按钮所挂载的控件 |
| InputEdit | 文本输入框 | 文字信息输入 |

表3.1 系统级控件介绍

1. 游戏级UI控件的实现

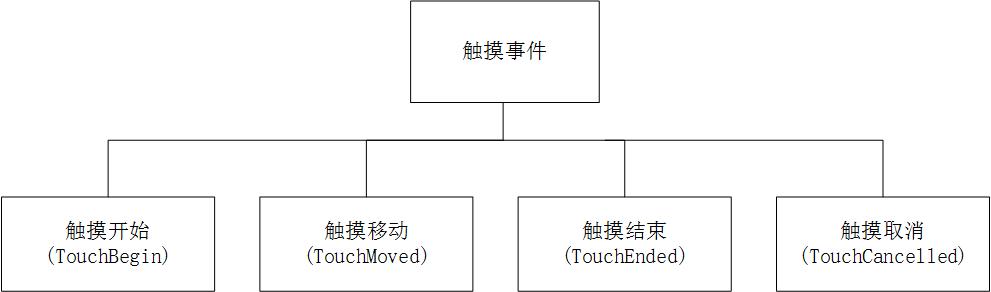
随着游戏的开发，我们需要越来越多的UI控件，大量的游戏UI是卡牌类型游戏所必须的，如在配置文件中通过系统级UI控件的方式去配置他们，势必需要大量的时间。所以为了增加UI系统的扩展性，减少开发难度，我们对所开发的游戏的UI控件进行总结，统计出常用的UI控件，比如菜单实体，倒计时器等，如表3.2所示。在UI系统和UI编辑器中添加对应的代码，使用系统级控件去构建。使用时可以直接通过对应的配置文件信息去管理。

|  |  |  |
| --- | --- | --- |
| 类名 | 控件名 | 功能描述 |
| 菜单实体 | MenuEntity | 游戏中的菜单界面，负责显示游戏中物品列表 |
| 面板跳转实体 | LayerSelecter | 不同面板之间的跳转按钮 |
| 列表切换按钮 | ListButton | 多个按钮拍成一排的切换控件，同时只有一个按钮可以被选中 |
| 倒计时器 | TimerEntity | 可以提供定时功能，与规定时间的时间间隔 |

表3.2 游戏级控件介绍

早期的手机不含有触摸功能，多为带键盘手机，玩家通过键盘的操作来控制游戏。玩家直接监听键盘按下所产生的消息，并调用特殊的逻辑对相应的UI进行修改。但近十年，随着智能手机的发展，电容屏手机已基本实现了普及，手机游戏多通过触摸屏幕和虚拟摇杆进行控制，触摸几乎成为了操作手机的唯一方式[27]。

但触摸功能的引用，造成了UI系统设计更复杂。触摸控制分为单点触控和多点触控，并且游戏中UI控件很有可能出现重合，在重合区域需要提供相应的分发事件方法以完成正确的操作。触摸功能需要采用一种完全不同于PC平台的输入事件接收方式，我们通常把一条触摸事件分为四条处理TouchBegin，TouchMoved，TouchEnded,TouchCancelled处理，并且配合适当的架构，才能解决触摸事件分发问题。

图3.2 手机游戏的触摸事件分解

以当前流行的手机游戏，UI系统已经成为不可缺少的一部分，尤其对于2D游戏来说，本质上view层全部是由UI控件组成。在实际生产过程中，我们需要提供一套完整和快速的工作流程，这套工作流程中可以通过可视化编辑器的方式调整UI布局，并且在游戏中以相同的方式显示。即使是在不同分辨率的设备上，也需要以规定的布局显示UI控件。可以快速为特定游戏设计出指定的UI控件，并且在游戏的另外一个地方重用这个UI控件。在编译器和游戏之间都需要UI配置文件，系统会统一管理UI配置过程中的配置文件，配置文件不会占用过大空间，适当优化配置文件体积大小。系统会以合适的方法加载UI控件对应的纹理等资源。

1

2

3

4

5

6

7

8

9

26 10

10 11

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

28

29

30

31

32