**程序设计文档**

# I 目录结构：

CardGame/

├── 引用

├── 外部依赖项

├── Resources/

│ ├── number/ ->存放点数图片

│ ├── suits/ ->存放花色图片

│ ├── card\_general.png ->卡牌背景图片

│ ├── undo\_normal.png ->回退按钮图片

│ ├── undo\_selected.png ->选中回退按钮图片

│ ├── level\_1.json ->桌牌"Playfield"区、手牌"Stack"区初始化解析文件

│ ├── fonts/

│ └── res/

├── Source Files/

│ ├── Classes/

│ │ ├── views/

│ │ │ ├── CardDefs.h ->枚举花色、点数类型定义头文件

│ │ │ ├── CardView.cpp ->展示一张完整的扑克牌图像

│ │ │ ├── CardView.h

│ │ │ ├── GameView.cpp ->负责游戏主界面渲染与交互的场景

│ │ │ └── GameView.h

│ │ ├── AppDelegate.cpp ->应用程序入口,初始化与生命周期管理

│ │ └── AppDelegate.h

│ └── proj.win32/

└── CMakeLists.txt

# II 游戏界面：

# III AppDelegate 模块

一、模块概述  
AppDelegate 类是程序的入口类，继承自 cocos2d::Application。该模块负责应用程序的初始化、生命周期管理（进入后台与恢复前台）、OpenGL 上下文设置以及主场景的加载和运行。是整个游戏程序运行的起点。

二、类定义  
类名为 AppDelegate  
该类继承自 cocos2d::Application（使用私有继承）

三、主要职责  
职责一：设置 OpenGL 上下文参数  
职责二：初始化引擎与游戏主场景  
职责三：设置设计分辨率和适配策略  
职责四：设置帧率与是否显示 FPS  
职责五：添加资源搜索路径  
职责六：处理应用进入后台与回到前台时的动画与音频管理

四、关键成员函数说明

函数一：AppDelegate()  
这是构造函数，初始化 AppDelegate 实例。

函数二：~AppDelegate()  
这是析构函数，负责释放所使用的音频资源。根据宏定义判断使用 AudioEngine 或 SimpleAudioEngine，并调用对应的 end 函数清理资源。

函数三：void initGLContextAttrs()  
该函数设置 OpenGL 上下文属性，包括颜色缓冲区、深度缓冲区和模板缓冲区的位数等。

函数四：bool applicationDidFinishLaunching()  
该函数在应用程序启动时调用，负责以下操作：  
操作一：初始化 Director 和 OpenGL 视图  
操作二：设置设计分辨率和适配策略  
操作三：设置动画帧率为每秒 60 帧，并开启 FPS 显示  
操作四：添加资源目录到搜索路径中  
操作五：加载并运行主界面 GameView  
此函数返回 true 表示初始化成功，返回 false 表示初始化失败导致程序终止

函数五：void applicationDidEnterBackground()  
当应用进入后台时调用该函数，负责暂停动画以及音频播放，节省系统资源

函数六：void applicationWillEnterForeground()  
当应用重新进入前台时调用该函数，负责恢复动画和音频播放，使游戏继续运行

五、关键实现要点

要点一：设计分辨率  
使用固定宽度为 1080 的适配策略，适配不同设备的屏幕尺寸，采用 ResolutionPolicy::FIXED\_WIDTH 策略

要点二：多音频引擎兼容支持  
通过预编译宏 USE\_AUDIO\_ENGINE 和 USE\_SIMPLE\_AUDIO\_ENGINE 判断使用哪种音频引擎，并确保不同时启用两者

要点三：资源路径配置  
通过 FileUtils::getInstance()->addSearchPath 添加资源目录路径，确保程序能正确加载图像和音效资源

要点四：场景加载  
程序启动后创建 GameView 场景并运行，是用户看到的第一个游戏主界面

六、总结  
AppDelegate 模块是游戏程序的起点，承担了初始化渲染环境、加载主界面、设置资源路径和处理应用前后台切换的重要任务。其设计符合 cocos2d-x 框架标准，模块职责清晰，代码结构简洁，便于后续维护和扩展。

# IV CardView 模块

一、模块概述  
CardView 模块负责展示一张完整的扑克牌图像，并支持点击交互。该模块继承自 cocos2d::Sprite，提供设置花色和点数、更新卡面显示、响应用户点击等功能，是用户界面中牌面渲染的关键组件。

二、类定义  
类名为 CardView  
继承自 cocos2d::Sprite

三、主要职责  
职责一：根据指定的花色和点数生成卡牌图像  
职责二：动态更新卡牌上的图案与数字  
职责三：支持启用和禁用点击交互  
职责四：绑定点击事件回调函数以响应用户输入

四、关键成员变量说明  
成员变量一：CardSuitType \_suit  
说明：记录当前卡牌的花色类型

成员变量二：CardFaceType \_rank  
说明：记录当前卡牌的点数

成员变量三：EventListenerTouchOneByOne\* \_touchListener  
说明：用于处理卡牌点击事件的触摸监听器

成员变量四：std::function<void(CardView\*)> \_clickCallback  
说明：卡牌被点击后触发的回调函数，外部可自定义行为

五、关键成员函数说明

函数一：static CardView\* create(CardSuitType suit, CardFaceType rank)  
说明：静态工厂方法，根据花色和点数创建一张卡牌实例。函数内部自动加载卡牌底图，初始化牌面元素。

函数二：void updateView(CardSuitType suit, CardFaceType rank)  
说明：刷新卡牌的图像内容。重新加载点数、花色图标，并将它们布置在卡牌的左上角、右上角和下方中部区域。

函数三：CardSuitType getSuit() const  
说明：返回当前卡牌的花色类型

函数四：CardFaceType getRank() const  
说明：返回当前卡牌的点数类型

函数五：void setTouchEnabled(bool enabled)  
说明：启用或禁用卡牌的点击响应功能。若启用，则添加触摸监听器，并在点击时触发回调函数。

函数六：void addClickEventListener(const std::function<void(CardView\*)>& callback)  
说明：设置点击事件监听函数。当用户点击卡牌时，会调用设置的回调函数。

六、卡牌图像渲染逻辑说明

图像一：左上角小数字  
资源路径为 number/small\_颜色\_点数.png，显示当前卡牌点数，位置设置为左上角（偏移 9，-8）

图像二：右上角小花色  
资源路径为 suits/花色英文.png，显示当前卡牌花色图标，位置设置为右上角（偏移 -1，-8）

图像三：下方中心大数字  
资源路径为 number/big\_颜色\_点数.png，放置在卡牌中心略偏下的位置，用于突出显示卡牌点数

颜色命名规则：红色为 red，黑色为 black  
花色英文名称包括 club、diamond、heart、spade

七、辅助函数说明

辅助函数一：std::string rankToString(int rank)  
说明：将点数转换为字符串表示，如 1 转换为 "A"，11 转换为 "J" 等

辅助函数二：std::string suitColorPrefix(int suit)  
说明：根据花色编号返回颜色前缀，黑色为 black，红色为 red

辅助函数三：std::string suitName(int suit)  
说明：根据花色编号返回英文名称字符串，用于拼接花色图片路径

八、点击事件实现逻辑  
在 setTouchEnabled 函数中创建触摸监听器，拦截单点触摸事件。  
判断触摸点是否在卡牌区域内，若是则触发 onTouchEnded 中的回调函数。  
通过 addClickEventListener 可设置回调行为，如翻牌、匹配等逻辑。

九、总结  
CardView 模块是游戏牌面显示的核心组件，封装了花色与点数的可视化渲染逻辑，同时支持交互响应，便于游戏控制逻辑与界面展示解耦。其设计兼顾了功能完整性与易用性，代码结构清晰，便于扩展如添加动画、高亮等效果。

# V GameView 模块

一、模块概述  
GameView 模块是游戏的主视图组件，负责加载和展示游戏界面的所有可视元素，包括背景图、卡牌区域（Playfield）、备用堆区域（Stack）等。该模块继承自 cocos2d::Node，不涉及游戏逻辑处理，仅作为视觉展示载体，为控制器提供完整的场景渲染与访问接口。

二、类定义  
类名为 GameView  
继承自 cocos2d::Node

三、主要职责  
职责一：加载并显示游戏主界面背景图  
职责二：根据布局数据创建并显示所有卡牌视图  
职责三：管理当前场景中所有卡牌视图的容器  
职责四：提供接口供控制器访问或操作场景中的卡牌视图集合

四、关键成员变量说明  
成员变量一：std::vector<CardView\*> \_cardViews  
说明：记录当前视图中已添加的所有卡牌视图指针，用于后续动画、移动、匹配等操作

五、关键成员函数说明  
函数一：static GameView\* create()  
说明：静态工厂方法，用于创建 GameView 实例并执行初始化操作。该方法会调用 init 函数构建界面。

函数二：virtual bool init() override  
说明：初始化主界面，包括设置背景图、解析布局数据、生成卡牌视图等。函数中对 Playfield 和 Stack 区域卡牌进行了坐标偏移设置以防止重叠。

函数三：void addCardView(CardView\* cardView)  
说明：向场景中添加一个卡牌视图节点，并将其加入卡牌列表 \_cardViews 中进行统一管理。

函数四：std::vector<CardView\*>& getCardViews()  
说明：返回当前 GameView 中所有卡牌视图的引用，控制器可通过该接口获取卡牌对象进行操作。

六、场景布局与渲染逻辑说明  
逻辑一：背景图加载  
背景图使用资源路径为 "card\_general.png"，在 init() 中创建并设置位置为屏幕中心，作为游戏主背景。

逻辑二：Playfield 区域卡牌解析  
从 \_layout["Playfield"] 中获取卡牌数据列表，为每个卡牌创建 CardView，并通过 updateWith 设置花色与点数，然后加入 GameView。Playfield 区域整体向上偏移 STACK\_AREA\_HEIGHT。

逻辑三：Stack 区域卡牌解析  
从 \_layout["Stack"] 中获取备用堆卡牌数据列表，同样创建 CardView 并添加。整体区域向上偏移 STACK\_VERTICAL\_OFFSET，其中最顶层一张卡牌需额外向右偏移 TOP\_CARD\_X\_OFFSET，以实现视觉层叠效果。

逻辑四：卡牌添加统一入口  
所有创建好的卡牌都通过 addCardView 函数添加到视图中，并统一加入 \_cardViews 容器进行管理。

七、使用说明  
说明一：GameView 不直接控制游戏逻辑，仅负责显示牌面和布局  
说明二：应在控制器中解析好 \_layout 数据，再调用 GameView 添加卡牌视图  
说明三：可通过 getCardViews() 访问所有卡牌视图，用于位置动画、状态变更等操作  
说明四：GameView 需配合 CardView 使用，后者负责牌面具体渲染与交互响应  
说明五：游戏重启或重新布局时，应清空原有卡牌视图并重新创建（建议新增 clearAllCards() 方法支持）

八、总结  
GameView 模块是卡牌消除游戏的界面核心容器，承担了游戏牌面布局与显示的职责。该模块结构清晰、职责明确，充分解耦了界面渲染与游戏逻辑控制，有助于后期维护与功能拓展。通过集中管理卡牌视图并提供统一访问接口，GameView 为控制器提供了良好的操作基础，是构建完整 MVC 架构的重要一环。