ICS 35.080
CCS L 77

T/CASME

中 国 中 小 商 企 业 协 会 团 体 标 准

T/CASME XXXX—2023

手机端游戏搭建技术指南

Technical guide to mobile game building

（征求意见稿）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

2023－XX－XX 发布　　2023－XX－XX 实施

中国中小商企业协会　　发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由××××提出。

本文件由中国中小商企业协会归口。

本文件起草单位：××××

本文件主要起草人：××××

手机端游戏搭建技术指南

1 范围

本文件规定了手机端游戏搭建的术语和定义、总体要求、开发环境搭建、游戏设计要求、技术框架、游戏功能模块要求、性能优化与测试、发布与上线要求、维护与更新。

本文件适用于手机端游戏搭建。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 22080 信息技术 安全技术 信息安全管理体系 要求

3 术语和定义

下列术语和定义适用于本文件。

3.1

手机端游戏 mobile game

指运行于手机上的游戏软件，简称“手游”。

4 总体要求

4.1 安全性要求

手机端游戏在设计和开发过程中应注重安全性，包括用户信息保护、防止恶意攻击、防止外挂等方面的要求。

4.2 稳定性要求

手机端游戏应具备良好的稳定性，不应频繁崩溃或出现严重错误，要求游戏在不同设备上运行稳定，并能够处理异常情况。

4.3 兼容性要求

手机端游戏应具备较广泛的设备兼容性，适配主流的操作系统版本和各类移动设备。

4.4 性能要求

手机端游戏应在保证良好体验的前提下，在资源占用、加载速度、图形渲染等方面具有良好的性能表现。

4.5 用户体验要求

手机端游戏应注重用户体验，包括界面友好、操作简单、游戏流畅、反馈及时等方面。

4.6 法律法规要求

手机端游戏应遵守中国的法律法规，不得含有违法、淫秽、恶俗、暴力、煽动性等内容，不得侵犯他人权益和隐私。

4.7 数据管理要求

手机端游戏开发过程中应合理管理和使用用户数据，符合GB/T 22080有关要求并遵循相关的数据保护法律法规。

4.8 更新与支持要求

手机端游戏在发布后应及时提供更新和技术支持，修复漏洞和bug，同时及时响应用户反馈和问题。

5 开发环境搭建

5.1 选择开发平台

应选择合适的开发工具和集成开发环境，如Android Studio、Unity、Cocos Creator等，并按照官方文档指引下载并安装。

5.2 安装集成开发环境（IDE）

应根据所选开发工具和集成开发环境的要求，配置相关环境变量和路径。例如，Android开发需要配置JAVA_HOME、ANDROID_HOME等环境变量。

5.3 安装编程语言和框架

游戏开发常用的编程语言包括C++、C#、Python等，应根据开发平台和个人喜好选择合适的编程语言。

5.4 下载和安装 SDK

应根据所选择的开发平台，下载并安装相应的软件开发套件（Software Development Kit，SDK），如Android SDK、iOS SDK等。

5.5 安装调试工具

为调试和测试游戏的运行情况，应安装虚拟机或真机调试工具。如Android开发可使用Android模拟器或连接一台Android设备。

5.6 创建项目

应根据开发工具的指引，导入已有的游戏项目或创建新的项目。根据游戏类型和需求进行相应的项目设置和配置。

5.7 配置编译环境

应根据项目需求，配置编译环境的选项，包括目标平台、编译版本、编译选项等。

5.8 导入依赖库和资源

应根据游戏需求，导入所需的依赖库（如游戏引擎、第三方库等）和相关资源（如图片、音频、模型等）。

5.9 创建代码框架和逻辑

应根据游戏设计，创建游戏的代码框架并实现相应的逻辑。包括处理用户输入、游戏物体的行为、游戏规则等方面。

5.10 调试和测试

应使用集成开发环境提供的调试工具，结合虚拟机或真机调试工具，进行游戏的调试和测试，修复bug和错误。

5.11 打包发布

完成开发、调试和测试后，应根据所选平台和发布要求，对游戏进行打包和发布，生成安装包或上传至应用市场。

6 游戏设计要求

6.1 游戏概念和目标

应明确游戏的核心概念和目标，包括游戏类型（如动作、冒险、益智等）、游戏背景故事、玩法理念等。

6.2 游戏玩法

应定义游戏的基本玩法规则和机制，包括用户输入方式、角色控制、游戏进程、任务目标、奖励机制等。玩法应易于理解和上手。

6.3 游戏界面和交互

应设计游戏的界面布局、按钮位置和大小、配色方案等，以及用户与游戏进行交互的方式，如触摸、滑动、拖拽等操作。界面应简洁清晰，交互应直观顺畅。

6.4 角色和道具

6.4.1 应定义游戏中的主要角色和次要角色，包括形象设计、属性设定、技能特点等。
6.4.2 应确定角色与道具之间的关系，道具的种类和功能，以及获得和使用道具的方式。

6.5 关卡设计

应规划游戏的关卡设计，包括关卡难度递增、关卡目标设定、可变化的地图布局等。

6.6 游戏音效和音乐

应选择合适的音效和音乐来增强游戏的氛围和体验。可根据场景、角色行为等设定相应的音效或选择背景音乐。

6.7 游戏进度和存档

应确定游戏的进度管理和存档方式，包括关卡进度的保存、用户数据的管理与存储等。

6.8 社交互动和多人模式

如果游戏需要社交互动或多人对战模式，应设计相应的功能和机制，如好友系统、排行榜、联机对战等。

6.9 用户反馈和调优

应设置用户的反馈机制，根据用户反馈改善游戏的体验，修复bug和错误。

6.10 版权和法律事项

应确保游戏内容不侵犯他人的知识产权，遵守相关的法律法规。如有需要，应进行版权申请和合规审查。

7 技术框架

7.1 应选择适合游戏类型和需求的游戏引擎，如 Unity、Unreal Engine、Cocos2d 等。
7.2 应采用客户端-服务器架构，将游戏逻辑分为客户端和服务器端。客户端应负责渲染和用户交互，服务器端应负责处理游戏逻辑和数据存储。可使用 C++、C#、Java 等语言进行开发。
7.3 应选择适合游戏需要的数据存储方案，如关系型数据库（MySQL、SQLite）、非关系型数据库（MongoDB、Redis）或者云服务（AWS、Azure）。
7.4 应使用网络通信库实现客户端与服务器之间的通信，如 Socket、WebSocket、HTTP 等。
7.5 应根据游戏引擎选择相应的客户端框架，如 Unity 可选择使用 uGUI 或新 UI 系统，Cocos2d 可选择使用 Cocos Creator 等。
7.6 应选择适合服务器开发的框架，如 Node.js、Django、Spring 等。

7.7 应根据游戏的特点选择合适的设计模式，如状态模式、观察者模式、命令模式等，优化游戏逻辑的组织和可维护性。
7.8 应考虑游戏的安全性和防作弊策略，如数据加密、用户身份验证、反作弊技术等。
7.9 应根据目标平台选择适合的技术方案，如使用跨平台引擎和开发工具，编写可在不同平台上运行的代码。
7.10 应使用版本控制工具（如 Git）管理代码的版本，结合持续集成工具（如 Jenkins）进行自动化构建、测试和发布。

8 游戏功能模块要求

8.1 游戏场景管理

8.1.1 应设计游戏场景的加载、切换、销毁等功能。
8.1.2 场景管理模块应具备灵活性和高效性，以便于游戏中各个场景的切换和流程控制。

8.2 用户界面(UI)

8.2.1 游戏中的用户界面应包括主菜单、游戏设置、角色属性、道具商店等。
8.2.2 UI 模块应具备可定制性和交互性。

8.3 角色控制

角色控制模块应具有处理用户输入、碰撞检测、动画控制等功能，确保角色在游戏中的自然表现和流畅操作。

8.4 物理引擎

8.4.1 应使用集成适用的物理引擎模拟游戏中的物理效果，如重力、碰撞、摩擦力等。
8.4.2 物理引擎模块应处理物体之间的物理交互和碰撞检测，提供真实的物理效果。

8.5 AI 模块

AI模块应考虑到游戏的类型和需求，采用合适的算法和策略来实现敌对角色、队友协作等功能。

8.6 存档与进度管理

8.6.1 应具备游戏存档和进度管理模块，记录用户的游戏进展和选择。
8.6.2 应支持保存和加载游戏进度、实现自动存档和回放功能等。

8.7 多人游戏

如果游戏具备多人游戏功能，应设计和实现多人对战或合作模式。多人游戏模块包括网络通信、玩家匹配、房间管理等功能。

8.8 数据统计与分析

8.8.1 数据统计和分析模块应收集游戏中的关键数据，如用户行为、游戏进度、付费情况等。
8.8.2 应通过数据统计和分析，了解玩家行为和游戏性能，并做出相应的优化和改进。

8.9 其他功能模块

应根据游戏需求可能还需要开发其他功能模块，如社交分享、广告展示、成就系统等。

9 性能优化与测试

9.1 目标设定

在进行性能优化前应首先明确性能优化的目标。例如提高帧率、降低加载时间、减少内存占用等。

9.2 性能分析工具

应使用合适的性能分析工具来收集游戏运行时的数据，如帧率、内存占用、CPU使用率等。常用的性能分析工具包括Profiler（如Unity Profiler）、GPU监视器和内存分析器等。

9.3 代码优化

应根据性能分析结果，对游戏代码进行优化。优化方式包括算法优化、内存管理优化、渲染优化等。

9.4 图形优化

应使用合适的渲染技术、减少多余的渲染操作、使用合理的纹理压缩等方式，提升游戏的渲染性能。应优化Shader代码，避免过多的计算和纹理采样。

9.5 内存管理

应及时释放不再使用的资源、合理使用对象池并优化内存分配和释放的频率。

9.6 资源优化

应对游戏中的资源进行优化，包括纹理、音频、模型等。通过合理压缩纹理、降低音频质量、优化模型顶点数等方式，减少资源加载时间和内存占用，提高游戏性能。

9.7 手机性能适配

进行性能优化时，应考虑不同设备的性能差异。移动设备应注意处理器、内存和GPU的限制。根据设备的性能，进行适当的调整和优化。

9.8 游戏流程测试

在进行性能测试之前，应确保游戏的流程测试已经完成。包括测试游戏的各个场景、功能是否正常。

9.9 性能测试

9.9.1 应使用合适的性能测试工具对游戏进行全面的性能测试，包括帧率测试、加载时间测试、内存占用测试等。

9.9.2 应通过模拟不同的使用场景和负载，验证游戏在各种情况下的性能表现。

9.10 优化迭代

应根据性能测试结果进行进一步的优化迭代。重复之前的步骤，收集新的性能数据并进行分析，然后根据分析结果进行优化。

9.11 兼容性测试

在性能优化完成后，应进行兼容性测试，确保游戏能够在目标平台上正常运行。

10 发布与上线要求

10.1 在游戏发布之前应进行全面的测试和验证，包括游戏性能、稳定性、玩家平衡、UI 设计等方面的检查。

10.2 应根据游戏类型和目标用户，选择合适的发布平台。

10.3 应根据发布平台的要求提交的发布材料。材料包括但不限于游戏介绍、游戏图标、游戏截图、视频预览、用户协议、隐私政策等。

10.4 将游戏上传到发布平台并提交审核，应根据审核结果进行修改和改进。

10.5 审核通过后，应在发布平台上配置游戏信息，如定价、语言支持、加密方式等。可根据实际情况选择不同的配置选项。

10.6 应确保游戏信息配置无误后正式发布游戏，发布时应进行必要的说明和描述。

10.7 游戏开发者应持续对游戏进行迭代和改进，解决 bug、添加新功能、调整平衡等。

10.8 应向玩家提供及时的技术支持，处理玩家反馈和投诉。

11 维护与更新

11.1 监测游戏运行

应持续监测游戏的运行情况，通过收集和分析游戏数据、用户反馈、bug报告等，了解游戏存在的问题和优化方向。

11.2 修复 bug 和漏洞

11.2.1 应根据用户反馈和 bug 报告，及时修复游戏中的 bug 和漏洞。
11.2.2 修复 bug 的过程中，应进行详尽的测试，验证修复结果，避免引入新的问题。

11.3 添加新功能

11.3.1 应根据玩家需求和市场趋势，不断添加新的功能和内容。
11.3.2 在添加新功能之前，应进行充分的设计和测试，确保新功能能够与现有游戏逻辑和系统良好地结合。

11.4 调整游戏平衡

11.4.1 应根据玩家的反馈和数据分析，进行游戏平衡的调整。例如调整角色技能的强弱、平衡多人对战的匹配机制等。
11.4.2 平衡调整应有针对性地进行，不得对游戏整体造成负面影响。

11.5 添加内容和扩展包

11.5.1 应定期添加新的内容或推出扩展包。内容可包括新的关卡、任务、装备、角色等。
11.5.2 在添加内容之前，应进行充分的设计、测试和平衡调整。

11.6 与社区互动

11.6.1 应建立和维护游戏社区，与玩家直接互动。
11.6.2 应通过回应玩家的问题、收集反馈、举办活动等方式，从社区中获取宝意见和建议。

11.7 优化性能

11.7.1 应持续优化游戏的性能，包括帧率优化、加载时间优化、内存占用优化等。
11.7.2 应进行针对性代码优化和资源管理，提升游戏的流畅度和稳定性。

11.8 发布补丁和更新

应及时发布游戏补丁和更新，修复已知问题、添加新功能或者进行调整和平衡。

11.9 进行市场推广

维护和更新的过程中，应进行适当的市场推广，市场推广方式包括但不限于社交媒体宣传、与相关媒体合作、举办活动等。