SuperNES: 基于安卓平台的 NES 模拟器

李嘉睿

10175101250

2021.6

主要内容

- NES emulator
- 安卓平台游戏框架
- 业务和 ui 相关
- 演示

什么是模拟器

模拟器

通过软件或硬件，使得一台计算机系统在行为上类似于另一台计算机系统，从而能够运行为另一台设计的软件或外部设备。

- SuperNES：在安卓平台上运行为 NES 系统设计的游戏软件。
- 实现原理：基于解释执行的二进制翻译
  - 二进制翻译：将客户机二进制文件翻译到主机执行
  - 解释执行：对每条指令实时解释执行
- 需要模拟的硬件
  - CPU
  - PPU
  - BUS
  - JoyPads

NES emulator - CPU 模拟

- 体系结构
  - 硬件：6502CPU，累加器架构，pc、sp、状态寄存器等，2KB 内存
  - ISA：CISC 架构，1-3 字节，256 条指令，13 种寻址模式
- 如何模拟?
  - 内存 -> 无符号字节数组
  - 寄存器 -> 寄存器对象或实例变量
  - 指令 -> 按语义用 kotlin 对应语法翻译
- 难点：如何保证正确性？测试!

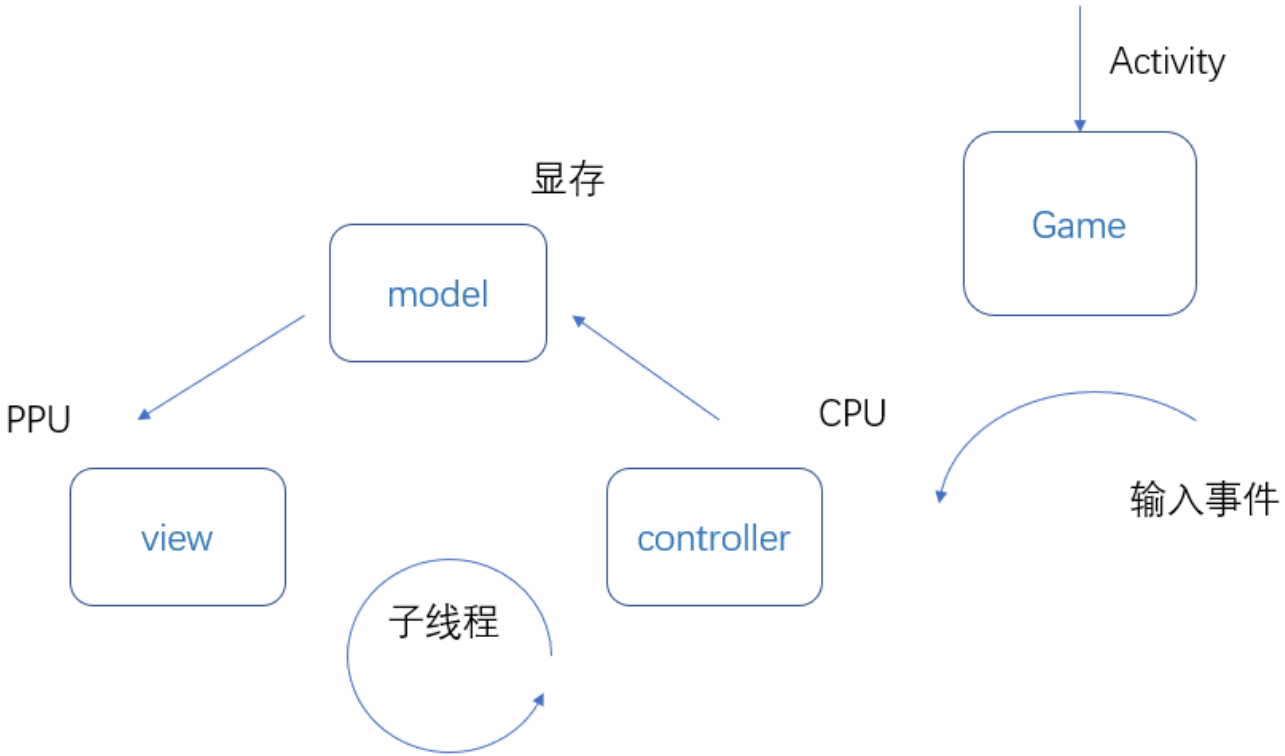
```
while (true) {
    val opcode = nextCode();
    when (opcode) {
        0xA9u -> lda() # 从内存装载数据到寄存器
        0x69u -> adc() # 加法指令
        ...
    }
}
```

NES emulator - PPU 模拟

- 什么是 PPU：独立的渲染处理器，有独立的内存、寄存器，类似于 GPU
- 工作原理：根据自己的内存（即显存）持续渲染显示器，该处内存如何与 CPU 同步？
- 交互
  - CPU 通过 IO 映射寄存器（CPU 写总线，总线转发给 PPU 寄存器）
  - PPU 进入 V-BLANK 周期向 CPU 发送中断信号
    - 固定扫描线共 262 行，每行渲染完成固定需要 341 个时钟周期，从第 241 行开始是 V-BLANK 区域。
- 如何模拟？
  - CPU 和 PPU 硬件并行 -> 软件串行模拟并行，how?
    - CPU 每执行一条指令，更新时钟周期，同时通知 PPU，PPU 维护当前扫描线，当进入 VBLANK 区域，PPU 设置中断状态，CPU 暂停执行交出控制权进行渲染
    - 这种模拟方式，每 341 * 241 个时钟周期渲染一次，也即刷新率
  - PPU 渲染如何模拟 -> 传递给 PPU 一个 Bitmap 对象，PPU 使用 Canvas API 根据显存的值渲染该对象

安卓游戏框架

- 通过子线程进行渲染的视图：SurfaceView
- MVC 设计模式
- 优化：外部点击事件的实例池，重复利用对象，减少垃圾回收

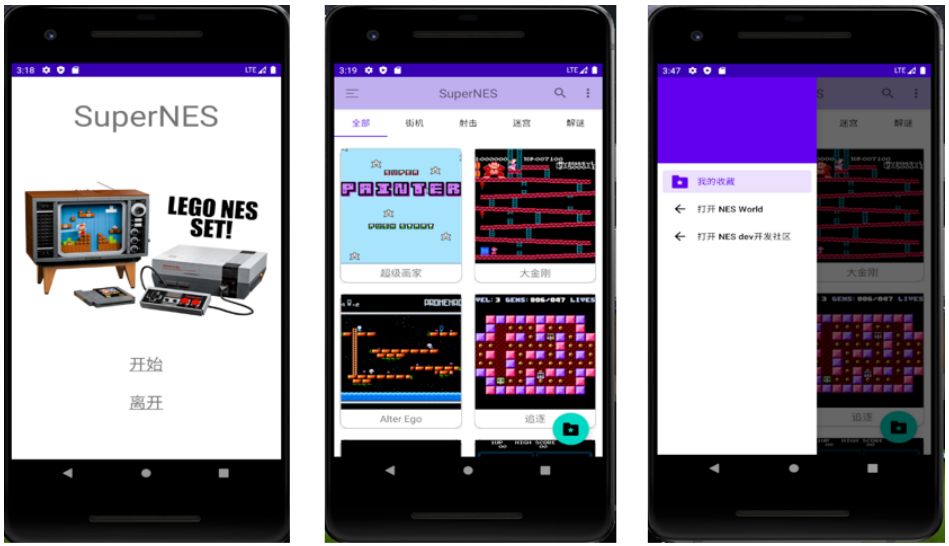


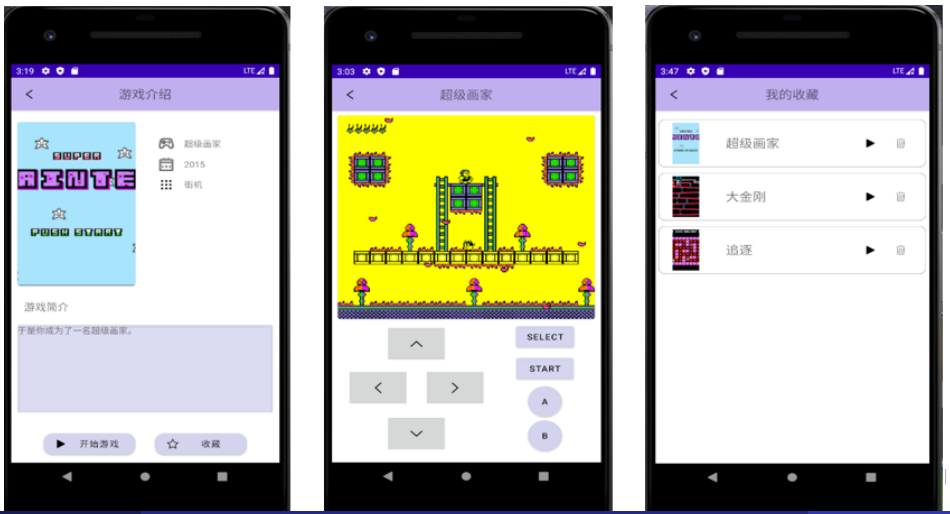
业务和 ui 相关

- 业务：收藏游戏，试玩游戏，搜索游戏
- 数据：游戏二进制 rom 文件采用文件存储，元数据使用游戏表、用户收藏表存储
- UI：力求简洁美观，material design
  - constraintLayout 布局
  - viewPager + fragment
  - toolBar + searchView
  - 大量使用 recyclerView

演示

- 目前已经移植完成 13 个游戏
- 包括街机、射击、解谜、迷宫等类别。





谢谢

https://github.com/Willendless/SuperNES/
v1.0 版已经发布。