1. 架构性能、游戏

特定领域 完整引擎

时间顺序

快速构建和迭代 不污染代码库 性能 C++

测试 需要研究问题 多态，泛型解决动态传参的问题（传入不同对象）

解耦 抽象化

观察者 消息机制

写代码前先规划：写中文计划，写中文变量和函数 最后实现

对于代码实现：用中文写逻辑推理

1. 设计模式
2. 责任链 处理不了就交给后面 用对象实现比较简单 各对象有处理方法 a.solve(){ if(false)b.solve() } b.solve() { if(不能处理) c.solve() } 这样链式调用

另一种考虑 类似状态模式 当前处理者 currentsolver=a currentsolver.solve( if(无法处理) currentsolver=currentsolver.next )

可以写出通用属性 currentsolver next等，然后递归实现链式调用

1、命令模式 就是对回调的面向对象实现

按键命令 按键-行为

把行为（函数）封装到命令对象中 方法:执行行为，撤销行为 buttonA->execute()

可以在执行方法中，把接受命令的对象传进去， kill->execute( npc){npc.die()

AI和角色间使用命令对象

也可以给玩家加上Ai,把命令按照队列来顺序执行

命令对象是可以动态传入的， 接受命令者也可以动态传入 ，接受命令行为可以多态（重写 ） 所以可以完全解耦

策略游戏撤销操作 需要把属性回滚 或者倒着对受命令者发出命令（部分适用）。

也可以用持久化数据结构（对象，数组的持久化存档，文件等）

多重撤销：记下指令列表，每条指令都undo

回放： 记录每桢状态（属性，太耗空间） 记录所有的命令（或许只需要记命令名，或者代号）

完全可以用高阶函数实现，回调

直接返回命令对象return kill:{ execute() } 返回闭包形式 return execute:function(){ }

命令对象的多态 对于命令由谁处理（接受命令的是自己还是别人）可以写成职责链模式，处理不了就转发给下个对象

2、享元模式

过多的对象挺浪费的，享元模式是把对象的不同转化成属性（状态）的不同，一个对象，多种属性

简单来说就是共享属性，各对象用指针或者对象 来获取共享的属性，避免了内存的占用和维护的困难？

显示无数的树，很多多边形属性，大量的数组参数 但是树大部分属性是一样的

把共同属性抽离（仅一份数据/可以是对象，数组任何形式） 然后所有其他对象共享该数据 这样似乎只会减少空间吧

指针指向数据 指针追逐 缓存不命中

对象池 状态对象

3、观察者模式 mvc架构 event 发送通知 发送者-观察者 比如html的点击事件等

Emit addEventLisener()

观察者 需要注册一个事件处理函数 onNotify(event){ 回调 }

被观察者发送通知 向谁发送 如何冒泡 通知传递到所有可接受通知的对象，然后引发所有人的处理函数 或许需要一个属性保存所有观察者列表（数组），然后遍历所有观察者，一一尝试传入event对象参数，触发其事件处理函数

可以有方法增加列表中的观察者，移除观察者 发送消息的顺序

事件对象，包含了事件的信息，比如点击事件坐标，键位，目标对象

事件、消息、观察者、 数据绑定， 观察者是同步的，那么异步是如何实现的

异步模型: 考虑硬件结构，汇编cpu原理， 不断地交换进程线程时间片 保存和切换寄存器堆栈环境 ，一步步地执行代码

异步情形，定时器 ,io 网络通信 回调机制 浏览器有定时器线程,http请求线程，完成后触发事件，事件进入事件循环

Js执行方法，会有context,类似于windows线程context 在执行栈中执行，而事件则会被挂起，等到（执行栈里的）同步代码执行完才会取事件队列？？？（那么无限循环不会阻碍他的异步吗？）

Js的堆栈分配

关于定时器实现:需要专门的线程，while(now()-time<1000){回调函数 }

事件队列? 无限循环 事件队列需要传入什么 event对象，event目标,注册函数?还是从一个推出其他？ Js每次点击都会层层冒泡，说明观察者列表里很多，他又是如何识别回调函数的名字的，需要知道吗？

动态分配，垃圾回收，担心分页 分配和回收都要时间

链式观察者，所有的观察者可以保存成链，方便遍历，增加删除

链表节点池

如何回收观察者，失效监听者

基于 高阶函数的观察者 数据绑定，双向绑定

3、原型模式 产生相近对象 克隆 深拷贝（嵌套对象拷贝 对于对象中包含的对象type属性 ）

模板泛型

不用类实现oop a={ }

数据的组织 monster={ hp:100;mp:50:atk:10;def:10 }

4、单例模式 应该避免运行 单例目的 （1）不重复创建（2）全局任何作用域随时唤出同一对象（作用域穿透） 更方便地访问（3）多个对象共享同一个对象 (4)管理器 （5）多线程同时操作 （6）网络并发 数据的顺序

配置信息 管理 控制 代理

不请求就不创建，运行时实例

java框架的bean对象是自动创建的，有些应该只有一个，比如控制器，有些应该同步（?）

全局变量可以代替单例吗？ 只要不再new不就不会产生第二个单例了吗？

静态类可以代替单例吗？

创建方法 获取单例 getsingleton() 运行时阻止生成更多实例 ，编译时

应该减少全局，不适合并行 耦合

避免碎片，严格管理堆上的内存

管理器：怪物、粒子、声音、系统、引擎 可以用静态类：完全不需要对象

如何传递对象 作为参数 / 从全局（不太好）/从其他全局（比如stage）/

子类沙箱 服务定位器

5、状态模式 FSM有限状态机 层次状态机 下推自动机

状态转化的函数 受当前状态影响 走 跑 倒地 滞空 空中跳

状态图 状态，和影响状态的操作

所有状态集合;同时只能是一种状态;输入或者事件导致状态变化;状态转移和另一状态有关

(1)、switch 每种状态 进行不同输入后执行的操作 如站立状态下 按左走 按上跳 按下无反应 列举各种状态接受**所有输入**的反应

(2)面向对象 多个状态对象 输入的同时 执行方法 并 改变当前状态对象

当前状态.执行（输入） 试图在当前对象和执行的输入上都使用多态，解耦

用动态对象参数代替switch,直接调用不同对象方法 需要，不同对象，同名方法

并发状态机：状态越多，关联越复杂 拿枪跳和空手跳 用多个状态机代替

平行状态可以分成两个状态机

分层状态机 状态行为类似 用父子状态，继承，如果子状态没处理给父状态

或者状态栈

下推自动机 栈指针

状态机用于：玩家输入，导航菜单，分析文字，网络协议，异步行为

1. 序列模式

1、双缓冲模式 需要同时进行 在各自循环进行（cpu可以看成无限循环执行？）

像素点数组 桢缓冲被存入内存，被显卡读取显示 显示需要一次显示出来，不能读取一半的缓冲

我们用两个缓冲

原子操作，改变完成才进行交换 外部请求，没完成改变的时候拒绝 交换时必须锁住两个区

更新方法模式 update()

2、游戏循环

定时执行的代码 ，操作着所有图形对象 数据对象的变化，行动，互相的关系 和碰撞检测，其他检测 对象遍历，属性遍历 等

每个循环都需要把所有行动对象都遍历，并且使用对象方法或者 其他方法来操控对象属性 或者执行比较、判断、检测等

事件循环:无限读取事件列表（队列或者数组），等待动态添加进来的事件

在浏览器中这个循环已经存在，在很多语言也实现了现成的事件循环机制，所以无需自己写

循环：读取输入， 更新，渲染 滴答tick 桢Frame 固定速率运行游戏

停止循环来停止游戏 减速

多个循环，可以实现一些差别时间的效果，但是需要维护

变化时间的循环，动态桢数

循环步长，每次循环时间是否需要同步（算上代码执行时间）

**尽量不使用自己的循环，使用现成的循环封装settimer之类的，要么用原生的循环包装一下。**

**While(nowtime-time<1000) 利用goto之类的 如何实现每秒执行一次？**

**While(true){getfromeventqueue(); } 不断从事件循环中取事件**

3、更新方法

每个对象都封装自己的行为以免代码混杂

Move()

Jump

Update 游戏世界 管理 对象集合 每个对象提供一个更新方法update（一桢内所有行动） 该方法是行动的封装 建议把每一桢的每个行动也封装成方法

这里把代码用桢（或者其他时间片，如秒） 为单位行动 定义每一桢的各种行为，控制属性的改变 增加删除 遍历数据结构（数组链表等）

行为的执行顺序， 行为的条件判断是否执行

存储状态

关于停止运行的对象：超出凭屏幕，停用， 遍历浪费时间，应该设法从循环移除掉

遍历对象需要一个集合来存储对象，一般是数组，对象池，链表之类的

不活跃对象用单独集合保存，或者保存所有活跃对象

4、多用组件模式，少用类继承， 让其他对象成为对象的部件 { {} } 实现复杂结构

组件 可以实现解耦

行为委托给其他对象： 中间转一层，过滤或者修改， 然后转发给接收的对象

发送消息（） 什么是消息（传递数据或者获取数据，本质是对接受者的方法调用 ,get set） 接受者是谁 接受者的方法获得发送者的参数，然后调用

Boy.send(girl,flower){ girl.receive(flower) } boy 对girl发送消息 调用girl的方法（也可以认为是事件处理，回调等）

1. 行为模式
2. 每个实体做什么

字节码模式 编译很耗时，调bug很难 数据>代码，如何让数据易于加载调试，和程序其他部分隔离

解释器模式，中缀表达式转表达式树 创建对象，连接，进行树递归算

编成字节码，游戏中运行个小虚拟机 jit

基于栈的虚拟机（容易实现） 基于寄存器的虚拟机（也有栈，还可以随意调用）

指令集 语法转换工具yacc

思考如何简化指令集 ，简化实现 如何最简的脚本语言 没必要的实现可以取消

静态语言也不错

语法定义

解析器

错误处理

1. 子类沙箱

父类提供部分操作供子类共同调用，protected 沙箱方法则由子类继承重写，重写时可能需要用到父类的操作

类似装饰模式，把方法包装成新的方法

1. 类型对象（模仿类型系统）

类型对象 和具有类型的对象 不同类型对象的属性变化，代表类型的变化

有类型的对象，其行为根据类型的不同而不同，此时的行为要自己设计算法，来计算类型对象对其的影响

简单的想法是设计多种行为，然后根据类型对象值不同switch一种

用类型对象的构造函数或者是工厂函数，new一个拥有类型的对象 ，使行为和类型系统类似

封装还是暴露

类型是否可改变，能否继承

和享元很像

1. 解耦模式

1、组件模式 不同领域的代码 放在不同组件中 音乐系统 碰撞系统

对象中的对象 {{} } a={b,c}

不同功能分到不同对象 不同数据分离

内部创建的组件 外部传入的组件对象

组件通信:修改父容器的状态（共享数据保存到容器） 互相引用 作为参数，作为外部变量传入到函数 作为属性 发送消息

2、事件队列 消息队列 事件循环 消息泵

Gui 事件驱动编程 鼠标点击事件是系统生成的，添加到win程序的消息队列，底层是cpu中断例程

游戏不是事件驱动，而是主循环

不过有

事件队列是把请求放进队列就结束 ，然后另一边循环取出事件 两边是解耦的

合并请求

分离线程

事件，通知和消息/请求/命令的区别 一个是告知事件已发生，一个是发出行为请求，区别不大，都要调动处理程序

单播队列 只一个监听

广播队列：很多监听

发布提交

3、服务定位器 出现在全局的东西 对象或系统 日志 内存分配 随机数-服务

静态、单例

服务类 抽象接口 服务提供者实现接口（切换提供者就能实现多态，不同的接口实现）服务定位器查询服务

Js就用动态对象呗 重写方法调用就是了

1. 优化模式
2. 数据局部性 ram从任何一个块读取数据速度都一样，和光盘硬盘不同

但是CPU缓存技术 读入一整块内存 cache line 假如下一个要读的在，缓存命中，如果不命中，会重新读一块 而不是继续执行

让要处理的数据 让 要执行的数据 相邻

冷热分割：每桢要用的数据

1. 脏标识

位置依赖于父对象属性 如何转换为世界坐标，从父层层下推相对位置

需要每桢都把所有对象从父算起吗？

并不是所有对象都移动了，是否只用改变移动的呢？

世界变换

需要先缓存所有对象的世界信息：世界坐标等，避免重算

给每个对象增加标识：是否改变了？ 自身发生变化后-设置该位 –更新世界数据-渲染-清除该位

用在一系列对象需要用复杂过程导出某个数据的时候 ，比如是否碰撞，周围是否有人？ 对象当前位置

未保存的修改

之前的推导数据需要保存在内存中（已发生碰撞等） 对于不变的对象不需要重新算，每次变化都需要重算

1. 对象池 对象重用

粒子，零散对象移除时产生内存碎片 不方便连续存储 对象池可以不释放

可重用对象具有状态：使用中 isuse（）

遍历池子

池：频繁创建销毁 对象相仿 堆分配会碎片化 而且缓慢 封装的数据库连接非常复杂昂贵的对象

池内部初始化对象。还是外部

1. 空间分区 对于对象需要知道彼此位置 需要从位置获取周围对象

遍历所有对象，判断彼此的位置关系 需要两层循环

使用一个空间数据结构，存储对象的空间位置，对象的id等

数据可以按照空间位置、距离排序 比如二维数组，可以用索引表示距离，顺序

对象改变位置，空间结构中的信息也改变

固定网格

网格为基础的碰撞检测 ，检查整格，半格，格边缘

网格为基础的移动（可以加入缓动，但实际还是以格为基础）

网格组织成双向链表

平面数据结构 划分和对象集合有关吗？ 可以把对象们全放在网格数组中，或者另外用个结构（比如数组）来存对象（**可以完全无序**），网格数组只存位置和对象名，id,索引等方便找对象

网格

四叉树

Bsp

Kd树

游戏组件 实体