1. 游戏循环 每次循环是1桢 60FPS 循环很多依赖于平台或者语言，来源于语言机制，或者是浏览器机制

传统的循环：处理输入（遍历事件队列），更新世界，渲染

力回馈

多线程的游戏循环 用主线程更新对象数据 用分线程渲染 让渲染比主线程慢1桢

游戏时间 ： 真实时间 桢时间 使用时间增量

游戏对象：绘制又更新数据 绘制不更新数据（比如背景图，只用显示一次，**不需要循环中**） 更新不绘制（不显示的，比如触发陷阱，暗雷怪，未必要用对象实现，但是可以碰撞检测）

接口：契约 共同

1. 渲染

屏幕上所有的像素点，颜色，亮度 把所有像素颜色保存到像素缓冲区

不应该不断输入像素，这会导致中途更新，屏幕撕裂 应该一次输入一个屏幕所有的像素，然后一次显示出来

双缓冲技术 读写在不同缓冲区 然后缓冲区交换

精灵绘制： 一层层地画 每桢擦除重画

可以抽象出对象的数据，位置，宽高，引用的图片，缩放 然后根据数据来**自动**地绘制

对象这些数据最后会被某种算法被转换为像素点

动画精灵，每桢图片

纹理图：精灵图 是为了用位置切换纹理

滚屏： 滚屏不仅是图片的卷动 同时也包含图片的数据，比如静态块类别（地面，砖块，梯子，随着背景卷动而卷动），方便碰撞 比如图片中包含的动态对象 这种数据如何识别是个问题，靠世界坐标（主角所在的坐标） 或者专门有个数组来存储这些？

自动滚屏 人物行走滚屏 ，无论如何其实都只是背景卷动 人物的世界坐标，屏幕坐标，地图上的块的世界坐标，屏幕坐标

碰撞： （地图碰撞） 由人物位置算出当前地图块的世界坐标，然后在某个数组中找到当前块是什么类型

（对象碰撞）把所有对象存到数组，每次遍历所有对象

摄像机是什么？

单轴滚屏 图片分割，

无限滚动（重复背景，随机背景）

平行滚屏 背景多层 不等速移动

四向滚屏：rpg

砖块地图tiled 砖块二维数组 用数组索引表示块位置，用数组内的数据表示块类型（数据可以是单个数字，或者是复杂的对象类型 ）

图块可以是共享对象吗？这样的话需要抽离出位置信息，使用同一份对象数据（属性） ，不同的位置渲染 避免了空间占用 这样或许需要把每个图块形成新对象，包含位置和一个共享对象

粒子也可以用同样的原理实现：粒子其他信息一样 ，只有位置不同，其他信息只需要一份（新对象只有两个属性，一个位置属性，一个**指针**）

但是这样一来，**访问属性的方式就要改变** : monster->share->hp 不可以给monster直接设置属性，不然占用的空间大小没区别 ，这样一来渲染方法的实现就要改变， 可以考虑使用代理，修改访问行为

斜视角

1. 线性代数

向量 带方向 加法 减法 长度 和标量相乘

点乘（算夹角）投影 叉乘，法线 手系 右手法则

三角形的法线

旋转（向量夹角）

线性插值

坐标系

矩阵 加法 减法 乘法（常用） 单位矩阵 逆矩阵 转置

4、3D GPU可以画点线三角 不需要程序员画 由三维直接转为2d平面

3D对象由多边形组成 三角形 3个顶点（三维顶点x,y,z） 模型网格有多个多边形（三角组成） 成千上万

模型坐标系，相对自身原点 世界坐标系

矩阵变换（修改各顶点坐标） 变成世界坐标 缩放矩阵 等比矩阵 平移矩阵 旋转矩阵欧拉角 所有变换叉乘，就可以共同作用

摄像机位置 观察位置 变换矩阵

投影坐标系:平铺到2D平面之后的坐标

光照 着色 rgb颜色 色深（即颜色位数，种类数，24位真彩） 不透明度（不透明通道）

顶点属性 纹理映射：贴图到三角上 UV坐标系 纹理上x u y v

光照：顶点法线 顶点序

环境光（每个物体固定光照） 方向光（照亮正面） 点光源 聚光灯

双向反射分布函数：光照模型phong

着色

5、游戏输入 数字输入：按下没按下 模拟输入：数字范围如摇杆

同时按键和顺序按键

模拟输入 （摇杆） 输入的范围 回弹 出界 按住

轮询和事件 轮询：定时查找状态

事件，发送带状态的事件到循环队列，系统有个循环逐个取出队列的事件，然后遍历所有可接收事件的监听方，触发所有监听方注册的回调

语言原生的事件系统 也可以用函数对象，函数指针，匿名函数实现 触发注册的函数，

**根据事件类型判断触发的注册函数，当一个监听者注册了多个函数 ，一个事件触发多个函数**

字典：键值对 键是字符串 数组空间

触屏 手势Rubine算法 手势识别库

加速器 陀螺仪

6、声音 随机播放 循环播放 同时播放的频道

声音资源映射 json

3D音效 发射和监听者 衰减 环绕

数字信号处理 修改声音信息，放大缩小 音高偏移 多普勒偏移 低通滤波器 冲量驱动（回车）

7、物理 平面 射线 线段

碰撞几何 包围球 包围盒 AABB（轴对齐） 朝向包围盒 ，可以随着对象的朝向旋转改变宽高 胶囊包围 凸多边形 组合几何体（比像素快，但是比简单几何检测，点检测慢）

碰撞检测：球之间 AABB之间 （旋转和缩放后的AABB？） 线段和平面交叉（检查线）

球与平台

子弹穿出墙面，连续碰撞检测

如何响应碰撞：减血，死亡，反弹，减速

多个对象的碰撞检测（遍历所有存在的对象）， 之间是否碰撞

分区：**只检查主角是否和分区发生碰撞 四叉树 分成越来越小的矩形，最后叶子是**

**每个对象 需要把对象按位置情况加入到树中**

**如果和某个区碰撞，再检测是否和分支的区碰撞**

二进制空间分割BSP 八叉树

基于物理的移动 F=ma 速度 加速度 冲量（在桢上起作用的力）

8摄像机 固定摄像机 根据玩家移动 摆放在固定位置

第一人称摄像机 跟随摄像机（地铁跑酷，跟在后面）

场景切换摄像机

9 人工智能 AI遗传算法 神经网络 效率不高

状态机规则

决策树：棋类

寻路 搜索空间，节点 ：把世界分成格子 邻近的方形就是邻近节点

主流游戏：不是一格一格走，每次移动是1步

路点（寻路节点）： 在地图按情况放节点（不是每个格都放），不在节点的区域不可走（可能导致节点太多或者太少）

导航网格：设置多边形区域，多边形内任意位置可走，其他的不能

寻路算法，启发式 ：估算路径开销

曼哈顿距离 欧几里得距离

贪婪最优先算法 临时存储节点的集合

A星算法

迪

状态机：行为就是不同的状态 走 站 跑 跳

不同状态 状态切换的原因 哪些可以切换 画状态转换图

根据当前状态 条件判断 进行切换

策略：微观策略用状态机就行了 宏观全局策略呢？ 策略可以认为是一系列的特定目标，一个策略有很多目标，哪个更重要

目标需要计划， 根据步骤的成功失败来调整步骤

10、用户界面

菜单系统

Hud 当前界面显示信息的 hp 得分 路点箭头 准心 雷达

菜单栈

按钮

打字

11、脚本语言和数据格式 引擎主体用c++等 游戏逻辑用脚本实现，然后解释

数据格式：二进制:图片 文本：配置 关卡数据 ini xml json

12网络游戏

数据包 tcp udp ip

Icmp 回声包

服务器 验证客户端行为

点对点模型（桢同步） 所有玩家同步 ，不能先于其他玩家行动