**《介绍》**

c++学习笔记

**《环境》**

**vscode**

使用vscode的优势在于vscode体量小、c++和python代码可以一起开发

vscode下载：https://code.visualstudio.com/

环境配置：下载mingw，在vscode插件中下载c++插件，搜教程配置编译环境。配置时注意使用g++替代gcc

MinGW：c++开发工具，包含GCC编译器，安装c++环境需要下载MinGW

MinGW下载地址：https://github.com/niXman/mingw-builds-binaries/releases

c\_cpp\_properties.json：配置好c++插件后会自动在.vscode下生成的环境配置文件

tasks.json：编译时的配置文件。自动生成：[终端菜单]->[配置文件]->[选择MinGW环境]。文件中的args参数的-g为debug编译(默认)，改为-02为release编译

编译：[终端菜单]->[运行生成任务]->[选择MinGW环境]。会生成.exe文件

运行：在下方终端中输入[.\文件名.exe]运行

vscode运行：点击[运行和调试]后会自动编译和运行，并且如果有断点会直接调试。快捷键[ctrl+f5]

新项目：创建新项目时，可以复制原来的.vscode配置文件

多文件编译：多文件编译(不同文件间函数调用)时要改造tasks.json

格式化：c++代码按ctrl+s保存时会自动格式化

Chinese插件：中文翻译插件

Remote-SSH插件：ssh远程连接服务器插件。安装后需要配置一下服务器的信息

PyCharm Theme 插件：将vscode风格换成pycharm

JetBrains IDE Keymap插件：将vscode快捷键换成pycharm

**其他**

cmake：开源、跨平台的项目构建系统，支持多种操作系统和编译器。比如从github下载和安装库，可以通过cmake的方式构建来避免环境影响

**《c++基础》**

c++标准：c++语言规范，国际标准化组织制定，c++标准和编译器都在不断更新

c++库安装：由于c++的编译器没有统一的标准，导致很多库不通用，没有像python的conda一样的库管理工具，要手动下载和管理c++的库。通常将库文件下载后，要在配置文件c\_cpp\_properties.json的includePath中添加库的"bin文件路径"，同时在tasks.json的args中添加"-I"和"bin文件路径"

编译器：c++语言需要编译器转换为对应系统的可执行文件，比如windows上编译后为.exe文件。常见的编译器有GCC、Clang、MSVS等。常用的编译器为GCC，支持多种系统和最新c++标准。通常会下载MinGW开发工具(包含GCC编译器)来配置c++环境

GCC：GCC编译器有gcc、g++两种，gcc针对c，g++针对c++

编译模式：编译器会将代码编译为可执行语言，有debug编译和release编译两种方式。前者包含调试信息，后者是经过大量优化的最终版本

字符串：c++中要使用双引号

静态库：.a或.lib。程序使用静态库时，会复制静态库的内容到系统上。优点是效率高，跨系统时不依赖环境；缺点是占用空间大，每次调试代码时要重新编译静态库

动态库：.so或.dll。动态库存在于程序外部，供程序调用

**导入库**

#include <A.h>：导入头文件A.h。A.h中存放源文件A.cpp中的函数声明，相当于导入A库，会将从系统目录、配置文件(c\_cpp\_properties.json和)中搜寻A.h文件。导入库后可以使用其中的函数和命名空间

namespace 命名空间{各类函数}：将各类函数放到命名空间下，可以嵌套。比如std::cout为使用std中的cout。命名空间相当于一个不需要实例化、简化版的类，方便维护

namespace name=命名空间：给命名空间起别名。通常为了避免命名空间冲突，命名空间名称会很长，使用时再用别名来简化

**数据类型**

不同编译器下的数据大小可能有区别，这里以常见的为例

字节：1个字节有8位二进制空间

short：短整型。2字节，-32768到32767

unsigned short：无符号短整型。2字节，0到65535

int 整型。4字节，-20多亿到20多亿

unsigned short：无符号整型。4字节，0到40多亿

long：长整型。4/8字节，64位linux为8，其他为4

unsigned long：无符号长整型。4/8字节，64位linux为8，其他为4

long long：长长整型。8字节

unsigned long long：无符号长长整型。8字节

float：单精度浮点数。4字节，7位有效数字

double：双精度浮点数。8字节，15位有效数字

long double：扩展精度浮点数。16字节，19位有效数字

Void：0字节

Bool：1字节

指针(\*)：4/8字节。32位系统为4，64位系统为8

枚举：变量默认4字节，可以定义数据类型

联合体(union)：所有成员共用一个内存

引用(&b=a)：引用成员的数值和内存地址都是同一个

**创建**

int a=1;：定义整型。也可以int a;//a=1;

char A="1";：定义字符串

&a：a的地址。所有的数据都有值和地址两个属性

int \*p=&a;：定义指针。注意定义后的指针是p不是\*p，指针p有自己的地址，值为其他数据的地址，\*p为a的值。即：&p=未知，p=&a，\*p=a。p+1指向a地址傍的下一个地址

const int a=1;：定义常量，之后a的值不能改变

constexpr int a=1：编译时常量。在编译时会直接将a替换为1，性能比const高，但由于在编译时优化，不能包含函数调用和对象构造

int array[a];：定义整型数组。由a个连续int数据组成。数组名array等效&array[0]

**基本操作**

#include <iostream> // using namespace std;：导入c++基础库

a++：自加1

a--：自减1

b-swap(A,B)：交换A,B两个数的值。A和B要为同一数据类型

floor(a)：向下取整，a可以为整数或浮点数

ceil(a)：向下取整，a可以为整数或浮点数

**输入输出**

cin >> x;：从键盘输入数据。x为提前定义的数据类型

cout << x << endl;：打印输出x，<< endl为换行符可不加

**条件语句**

if (条件){内容}else if (条件){内容}else{内容}：条件判断语句

for (int i=0;i<a;i++){内容}：for循环

while (条件){内容}：while循环

**字符串**

string定义的字符串是一个封装的对象，可以执行更高级的操作，比char更好用

string str\_;：定义空字符串

str2=str1+"...";：合并字符串

str\_.insert(a,"A");：在下标为a的字符前插入字符A

str\_.replace(a,b,"A");：从下标a开始往后共b个字符替换为A

str\_.substr(a,b);：选取下标a开始往后共b个字符

str\_.erase(a,b);：删除下标a开始往后共b个字符

str\_.find("A");：查找是否含有A，存在返回第一个字符的下标，不存在返回-1

**位运算符**

优先级：算术运算符>关系运算符>位运算符>逻辑运算符

a<<n：二进制左位移n次。等于乘以2的n次方，运算效率更高

a>>n：二进制右位移n次。等于除以2的n次方，运算效率更高

&：二进制与

|：二进制或

^：二进制异或

~：二进制非

**逻辑运算符**

&&：与

||：或

!：异或

**函数**

类型 函数名(类型 参数=默认值){...}：函数定义

void 函数名(int x){...}：定义形式参数

static void ...：定义静态函数。静态函数不能被其他文件使用

extern int a：在导入其他c++文件后，使用其中定义的变量a(函数可以直接使用)

inline void ...：定义展开函数。在执行代码时会将函数中的内容作为一个单独的模块，如果展开后与调用代码合在一起编译会更简洁、提高运行速度，但有for循环等会导致代码体积膨胀，导致编译后文件变大，因此对于复杂的函数不使用。相比类似展开的宏定义，inline有类型检查更安全

函数重载：允许有相同名称的函数，但它们的参数类型要不同。比如void 函数(int a)和void 函数(float a)，当传入参数不同时会调用不同的函数

**类**

class 类名{public:...}：定义类。类中有公有函数public、私有函数private、保护函数protected，不写时默认私有。public就是正常函数，private只供函数内部或指定类使用，protected和私有差不多

**《c++常用库》**

**<stdio.h>**

c语言的标准输入输出库，有printf、scanf,、puts,、gets等函数，c++也可以使用，但不支持面向对象编程、不提供类型安全。c++有自己的<iostream>库，提供更丰富的功能

**<iostream>**

#include <iostream> // using namespace std;

c++的标准输入输出库，早期版本使用<iostream.h>

**<string>**

处理字符串的库，不用关心内存分配和\0

string str1;：定义空字符串

getline(cin,str1);：输出带空格的字符串

str3=str1+“ “+str2;——字符串合并，中间可加空格

str1.insert(a,”A”);——在str1下标为a的字符前插入字符A

str1.replace(a,b,”A”);——从下标a开始往后共b个字符替换为A

str1.substr(a,b);——选取下标a开始往后共b个字符

str1.erase(a,b);——删除下标a开始往后共b个字符

str1.find(“A”);——查找str1中是否含有A，存在返回A第一个字符的下标，不存在返回-1

**<Eigen/Dense>**

#include <Eigen/Dense> // using namespace Eigen;

安装教程：<https://blog.csdn.net/qq_45948208/article/details/132434560>

Eigen是数学计算库。c++的eigen、numcpp和python的numpy库计算效率相差不大，单纯矩阵计算eigen>numcpp>numpy，动态矩阵计算(矩阵大小经常改变)numpy>numcpp>eigen

**<opencv2>**

c++版本的opencv函数效率和python版本的基本一致，但环境安装等比较麻烦，因此可以用python写相关函数，然后用c++调用python代码

**<python>**

#include <Python.h>

vscode从c++调用python代码要先安装anaconda或miniconda环境，下载python插件

配置：找到python环境的include位置，在配置文件c\_cpp\_properties.json的includePath中添加"D:\\anaconda3\\include"，然后在tasks.json的args中添加"-I"和"D:\\anaconda3\\include"、"-L"和"D:\\anaconda3\\libs\\\*"

Py\_SetPythonHome(L"D:\\anaconda3");：设置python库根目录

putenv("PYTHONIOENCODING=utf-8");：执行python程序用utf-8编码，放在初始化前

Py\_Initialize();：初始化

PyRun\_SimpleString("import py库");：执行python命令

**《ue5》**

**基础知识**

虚幻引擎5(unreal engine5)：

虚拟微多边形几何体(nanite)：可以让美术师创建出精致的几何体细节，无需考虑多边形数量、内存、绘制次数预算，不用将细节烘焙到法线贴图中也可以使用

全动态全局光照解决方案(lumen)：

安装：先下载epic games平台，在epic games平台中可以创建和下载不同版本的虚幻5引擎，初次安装时不需要选目标平台(文件很大)

游戏组成：虚幻5中一个游戏由多个关卡组成，开发者先制作每个关卡然后拼到一起

关卡：游戏世界的基本单元，独立的场景容器，每个关卡可以包含地形、光照、人物、UI、逻辑等元素。比如主菜单为一个纯UI关卡，对战地图包含地形、人物、逻辑等

文件名：不同平台的编码方式不一定是支持中文的utf-8，因此要全部使用英文命名

坐标轴：X轴是红色，Y轴是绿色，Z轴是蓝色

坐标单位：坐标单位是cm

蓝图：封装好的功能模块，由蓝图虚拟机读取并转为字节码执行。蓝图虚拟机本身由c++语言编写，蓝图的功能最终也会转化为c++函数执行。但是由于虚拟机要逐条解释字节码指令、动态检查数据类型、传递结构体、内存管理不够高效、无法进行c++中代码优化等，导致每帧都进行的高频计算、寻路算法等可能慢10倍以上

Blender：如果要对模型进行编辑、简化等，最好学会blender等软件

**制作流程**

游戏设计：确定游戏玩法规则

资源规划：制作或购买单位模型、材质、音效、动画

技术选型：用蓝图制作、c++编写寻路等高性能模块、mass ai框架管理大规模单位

玩家控制：玩家镜头视角、选中单位、下达指令

UI系统：主菜单、资源面板、单位面板、小地图。通过UMG控件蓝图和逻辑制作

地图设计：绘制地形、植被、遮挡碰撞体

单位：导航网格控制行走区域，行为树和黑板实现行为逻辑

关卡连接：将主菜单、关卡之间连接起来，保存玩家进度，跨关卡传递数据

优化测试：nanite处理高模地形和建筑，lumen光照设置，HLOD(分层细节优化)优化远距离单位渲染，测试单位寻路和多人同步

打包发布：打包游戏并发布到目标平台(手机或电脑)

**快捷键**

视角移动：按住鼠标左/右键进行前后和旋转，此时使用wasd和eq可以快速移动，按住鼠标的同时滑动滚轮可以修改移动速度，视角移动速度可以在视口的右上角看到

物体调整：q选择；W移动；e旋转；r缩放。移动物体时按住shift保持视角跟随

F：视角跳转到选择的物体

Delete：删除

FN+END：物体对齐到地面上。地面不平整或有其他物体时，会对齐到最高点

Ctrl+D：在旁边复制一个相同的物体

Alt：按住Alt拖动物体时会复制并拖动新的物体

鼠标中键：单个或多个物体进行操作时，对坐标轴按住鼠标中键可以临时移动坐标轴点

**性能**

蓝图与c++：每帧都进行的高频计算、寻路算法用c++重写可能会提高10倍以上速度

显示帧率和内存：[编辑]->[编辑器偏好设置]->[性能]

启用垂直同步(VSync)：[编辑]->[编辑器偏好设置]->[性能]

垂直同步(VSync)：使游戏画面与显示器刷新率一致，限制GPU性能，防止在复杂场景高速移动时画面撕裂。可能会增加输入延迟，在竞技类游戏中不使用

绘制调用(Draw Call)：ue5中每个物体渲染时CPU会调用图像程序(可能为GPU)，程序会进行切换渲染状态(材质、着色器、贴图)、加载顶点数据(位置、UV、法线)、顶点着色器和像素着色器计算、输出像素到屏幕。将多个物体合并为同一个物体会提高效率，但如果各物体的材质不同且复杂、单个物体顶点太多(30-50万以上)则反而会降低效率

物体合并：若要进一步提高效率，可以去除重叠看不到的面、简化所有面的顶点数

**基本页面**

创建项目：创建项目时可以选蓝图或者c++，c++需要安装编程软件和环境。创建项目后会生成一张空的关卡。每次进入项目后会生成空的关卡，点击项目中保存的关卡进行切换

项目文件：左下角[内容侧滑菜单]。可以查看项目中所有保存的关卡和资产

顶部选项：[文件]为保存相关操作；[编辑]对整个项目进行设置；[窗口]打开窗口

工具栏：对当前的关卡进行修改

视口：屏幕中间的画面窗口为一个视口。在上方[窗口]->[视口]中打开。关卡画面

大纲：右上窗口。在上方[窗口]->[大纲]中打开。关卡中所有的内容

细节：右下窗口。在上方[窗口]->[细节]中打开。关卡中选中对象的属性编辑

世界场景设置：默认没有，可以从右上[设置]中添加。设置关卡全局光照

**项目文件**

新创建或复制的内容会带上\*号，此时未保存，[保存所有]后会消除\*号

**材质**

材质(material)：右键空白处选择[材质]。双击材质进入材质图表，鼠标右键+中间移动。在图表中右键可以新建节点，基础着色器为最终的输出节点。基础着色器有输入时，可以在左侧[使用材质属性]，会将输入自动拆解到相应的通常上

材质实例：如果需要将材质应用到多个物体但只改变颜色等参数时，复制多份会占内存，创建材质实例的方式可以共用大部分相同的内容、更高效。如果在材质节点中定义了变量，则材质实例中可以修改相关的变量

官方文档：创建节点后，右键[查看文档]可以查看详细用法

MakeMaterialAttributes：材质属性，用于连接纹理贴图、法线贴图等

TextureSample：贴图节点，连在材质属性上。拖入纹理贴图、法线贴图可以直接创建

TextureCoordinate：控制贴图效果。连在所有贴图节点的UV上，通常会乘一个比例

Multiply：乘法。将两个输入相乘后输出

ScalarParameter：数值变量。定义变量后，在材质实例中可以修改

LandscapeLayerBlend：地形材质混合，创建自由绘制的图层。将多个材质属性连接到混合器，在左侧的[图层]中可以定义连接数量，注意新建的层需要命名。地形材质混合后的材质颜色是黑色的，需要通过地形绘制使用：[细节]->[地形]中导入材质，[地形模式]->[绘制]->[层]中点击[从指定的材质创建层]，对每个图层创建[权重混合层(法线)]，保存地形图层到本地，此时选择图层可以自由绘制地形的颜色，绘制后会改变本地的地形图层文件

**工具栏**

保存：保存关卡到项目文件中

选项模式：修改地形、植被等内容

添加图标：添加内容到关卡中

蓝图图标：设置蓝图

镜头图标：设置镜头

运行关卡：点击[三角形]运行关卡，ESC键退出并复原

平台：项目打包

**建模**

在[选项模式]中选择[建模]。修改和创建模型后会直接覆盖原模型

修改坐标轴点：[XForm]->[编辑枢轴点]中可以精确修改位置

**地形**

管理：创建新的区块

雕刻：绘制地形的形状。平整时设置扁平目标为0可以压平地面

材质：在项目文件中可以创建材质，在物体细节中可以应用

自由绘制图层：使用材质中的LandscapeLayerBlend节点制作材质。<参考该节点资料>

**光源**

在[添加图标]中选择[光源]。环境光使用[环境光源混合器]

添加光源后可以在[大纲]中选择，并在[细节]的[光源]中修改效果

环境光源混合器：[窗口]中选择，在混合器中可以同时编辑光照、大气、云等各类环境效果，初始项目大纲中的Lighting文件夹中为默认的环境效果

**视口**

帧数：FPS为每秒画面数，ms为FPS的倒数。FPS以30为基础(单机游戏)、60为流畅(3A游戏)、90以上为VR游戏、120以上为电竞级别，通常单机游戏不到30帧也能玩，会受电脑性能、复杂场景、优化和渲染技术(nanite,lumen)、垂直同步(VSync)

距离测量：[透视]切换为[顶视图]后，按住鼠标中键测量距离(cm)

对齐网格：右上角[自动对齐网格]开启后，会以固定距离(cm)移动物体。还有固定角度旋转、固定缩放距离等

多视图：右上角开启多视角窗口，可以看到不同视角或维度的窗口

**大纲**

HLOD0文件夹：包含HLOD(分层细节优化)的配置

Lighting文件夹：包含基本的光线、天气等。在[环境光源混合器]中统一编辑环境效果

Landspace：包含所有的地形区块

PlayerStart：开始关卡后玩家的起始点

WorldDataLayers-1：世界场景内容管理的配置

WorldPartitionMiniMap：世界分区缩略图的配置

**细节**

变化：改变物体位置、方向、大小。以物体坐标轴点为原点变化

静态网格体：用其他网格替换当前网格