/\*

红

\*/色：王荐宁

绿色：夏童奇

黄色：施杨

蓝色：亚梦溪

#include <queue>

#include <string>

#include <vector>

#include <thread>

#include <chrono>

#include "define\_classes.h"

using namespace std;

#define ROUND\_0 0

int now; //当前时间

//60帧/s -> 17ms/帧

#define PERIOD 17

//TODO: 改成单例模式

class ResModule{

public:

vector<Event> LoadMap(ObjType type, int typeId);//加载地图事件

void Load(BaseObject &obj, ObjType type, int typeId);//加载属性、贴图等

void Init();//初始化，检查文件存在与否

}mRes;

class RenderModule{

public:

void ShowIamge(BaseObject \*obj);

}mRender;

class SceneModule{

public:

vector<BaseObject \*> objList;

priority\_queue<Event> heap;

SceneModule();

void Init()

{

vector<Event> eventsList = mRes.LoadMap(ENUM\_MAP, ROUND\_0);

for(auto i : eventsList)

heap.push(i);

}

void Run()

{

while(true)

{

//延时

std::this\_thread::sleep\_for(std::chrono::milliseconds(PERIOD));

while(heap.top().occurTime == now)

{

Event &t = heap.top();

objList.push\_back(t.Occur());

heap.pop();

}

CollisionDetect();

//显示地图背景

//mRender.show\_img(wall, background)

//遍历更新物体并显示

for(auto obj : objList)

{

obj->Update();

mRender.ShowIamge(obj);

}

}

}

void CollisionDetect()

{

for(auto i : objList)

for(auto j : objList)

if(Detect(i, j))

i->OnCollision(j);

}

bool Detect(BaseObject \*i, BaseObject \*j);// 判断

}mScene;

void Welcome();// 界面、欢迎信息等

int main()

{

Welcome();

mRes.Init();

mScene.Init();

//开个新线程后台运行

thread backgoundThread(mScene.Run, &mScene);

backgoundThread.detach();

}

#pragma once

enum ObjType {

ENUM\_MAP, ENUM\_MONSTOR, ENUM\_PLAYER, ENUM\_WALL, ENUM\_BULLET

};

class BaseObject

{

public:

virtual void Delete() const;//删除对象

virtual void Update() const;//常规更新

virtual void OnCollision(BaseObject\* obj) const;//处理碰撞

protected:

int positionX, positionY;//横、纵坐标

int velocityX, velocityY;//横向、纵向速度

ObjType type;

int typeId;//物体种类ID

int uniqueId;//每个物体独一无二的ID

int shapeType;//几何形状的种类：0为圆形，1为矩形

int radius;//圆形半径

int width;//矩形宽度

int height;//矩形高度

};

class Charater : public BaseObject

{

public:

virtual void Update() const;//常规更新

virtual void OnCollision(BaseObject\* obj) const;//处理碰撞

virtual void Fire() const;//开火

virtual void Move() const;//移动

protected://先假设弹匣子弹无限

int bulletType, bulletPeriod, lastFireTime;//子弹类型、子弹发射周期、上次开火时间

float bulletPeriodRate;//子弹发射周期速度倍率(控制子弹发射速度)

int HP, HPIncrement, lastTime;//血量、血量增速(可正可负)、血量增速倍率、buff持续时间

float velocityRate;//速度倍率(控制人物的移动速度大小)

};

class Player : public Charater

{

public:

void Update() const override;//常规更新

void OnCollision(BaseObject\* obj) const override;//处理碰撞

void Fire() const override;//开火

void Move() const override;//移动

};

class Monster : public Charater

{

public:

void Update() const override;//常规更新

void OnCollision(BaseObject\* obj) const override;//处理碰撞

void Fire() const override;//开火

void Move() const override;//移动

};

class Item : public BaseObject

{

public:

void Use(Player\* user) const;//使用物品

void Update() const override;//常规更新

void OnCollision(BaseObject\* obj) const override;//处理碰撞

};

class Bullet : BaseObject

{

public:

void Update() const override;//常规更新

void OnCollision(BaseObject\* obj) const override;//处理碰撞

protected:

int attack;//攻击力

};

//map Event

class Event{

public:

int occurTime; //出现时间

int typeId;

ObjType type;

int posX, posY;

bool operator < (const Event &t)

{

return occurTime < t.occurTime;

}

BaseObject \*Occur()

{

BaseObject \*ret;

switch (type)

{

case ENUM\_MAP:

break;

case ENUM\_MONSTOR:

break;

case ENUM\_PLAYER:

ret = new Player();

break;

case ENUM\_WALL:

break;

case ENUM\_BULLET:

break;

default:

break;

}

return ret;

}

};