《虚妄与瘟疫的弹幕游戏》游戏设计

1、问题描述

1.1 游戏介绍

《虚妄与瘟疫的弹幕游戏》（以下简称《游戏》）是延续传统东方 project 正作玩法的独立游戏。弹幕游戏发源于上个世纪，从 fc 上的第一款弹幕游戏《沙罗曼蛇》，到在同人界掀起轩然大波的《东方永夜抄》无论哪一部都是百玩不厌的经典之作。经典弹幕游戏的玩法主要是在躲避子弹的同时讨伐首领和他的爪牙，东方 project 系列游戏则是增加了“符卡”这一设定。

本作的设计思路是在有所创新的同时延续“东方”风的弹幕风格。在一对一的对战同时增加了体力条和耐力条（ui 设计参考 monster hunter）。在低速状态下的移动将不再是无偿，随着玩家的移动耐力会随之消耗。

1.2 游戏流程

（完整游戏周期约为 40 分钟）

（1） 启动游戏，在弹出的命令提示行界面会有关于游戏的一些信息。在阅读文字后根据提示设置游戏难度及分辨率，启动游戏。

(2) 在启动游戏显示的是游戏的启动动画，此时按下【space】键可以跳过动画。动画一共有三段，依次观赏或跳过后进入游戏开始界面，按下开始键正式开始游戏。

（3） 游戏开始，将进入剧情对话动画。按下空格键可加快动画速度。

（4） 之后进入第一关，第一关为教学关，一位剧情 npc 将为玩家详细讲解游戏玩法及注意事项，部分对话内容如下：

阿斯克勒皮俄斯：你应该记得，用【方向键】来控制你在空间中的投影

游戏中，玩家可以通过按下方向键操纵主角移动。

阿斯克勒皮俄斯：注意，在瘟疫反抗手术时，瘟疫的投影也会由于本体的波动而扩散出弹幕状的攻击，若被击中会对你和患者的生命安全造成危害，注意躲避

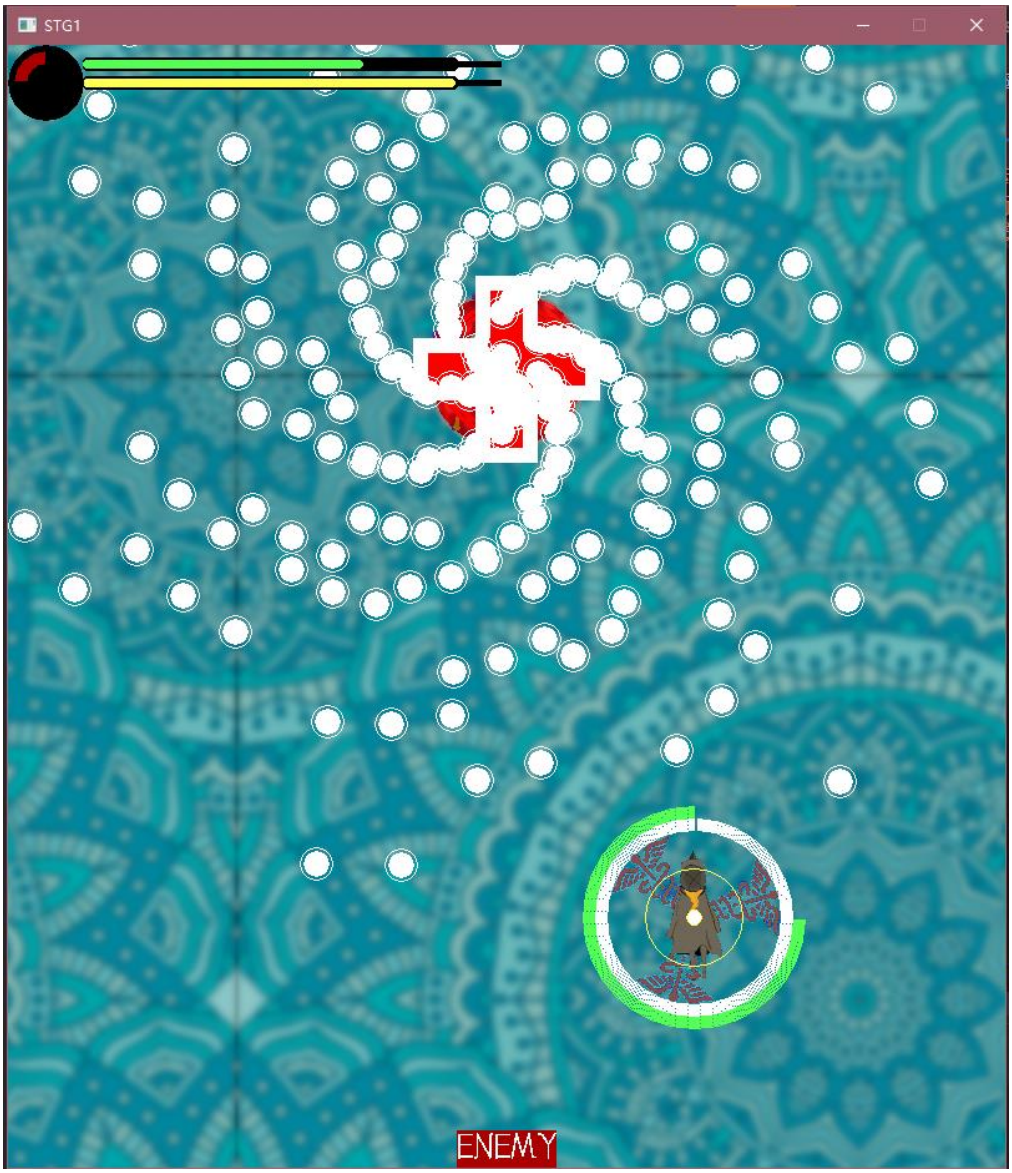
敌人会发射密集的子弹攻击玩家，若被击中会造成【hp（血量）】的减少，当血量减少至 0 后则会进入【死亡】判定。

阿斯克勒皮俄斯：对方可能会进行大密度的弹幕攻击，必要时刻不要吝啬耐力值，按下【shift】进入精细模式，是请留意耐力计量表，当耐力清空时将动弹不得，按下【X】键可以近距离观察体力与耐力计量表，但注意，观察的时候不可以移动和攻击。

普通的移动模式速度过快，可能会导致玩家不方便更细微的操纵，因此当按下【shift】的时候，主角的移动速度会大幅下降，并显示白色的命中判定圈，直到松

开【shift】键。该过程会消耗【mp（耐力）】，当耐力过低时，玩家将无法操纵主角。

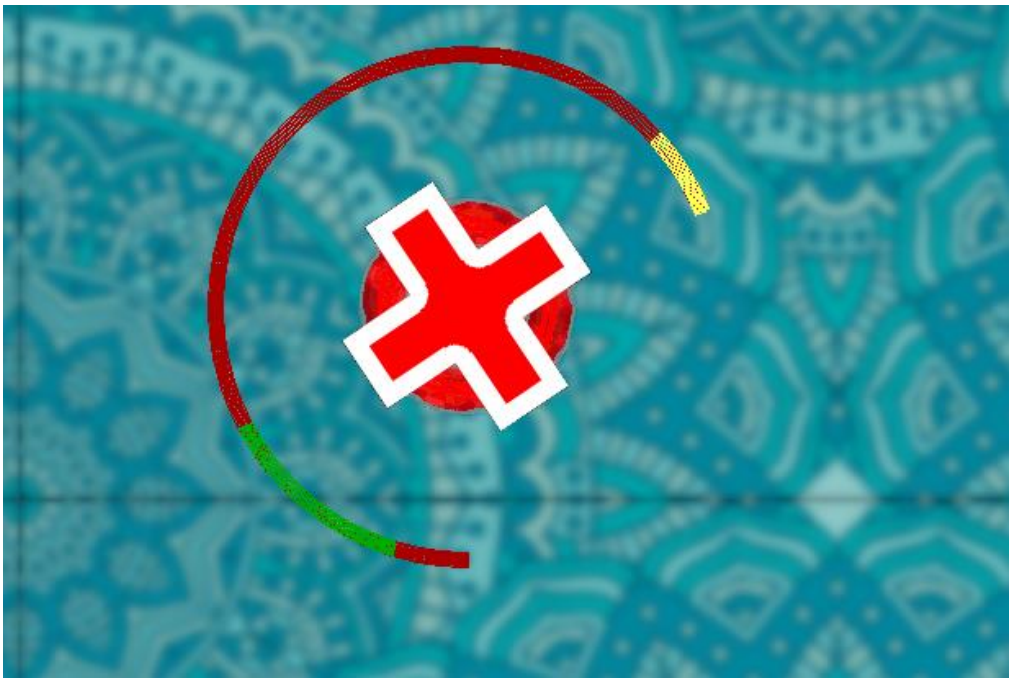
由于体力计量表与耐力计量表位于屏幕左上方，非常不利于玩家在高紧张度下监控，故按下【x】键可以在主角身边生成两个圆环，圆环的弧度代表对应的数值百分比。



阿斯克勒皮俄斯：是强化机会！把握住，不要错过！当圆弧走到黄色的位置时按下【空格】，你将获得【耐力加速回复】的强化，直到下一次强化机会，绿色的位置是【回复所有体力值】的强化。

关于【强化机会】：

在两波攻击之间，有概率会出现 Intensify Chance，此时会在敌方的周围出现表示时机的圆弧。在合适的时机按下空格会获得相应的 buff。当圆弧为黄色时按下会获得增强耐力恢复的 buff 并持续一波攻击；当圆弧为绿色时按下会缓慢恢复所有血量，直到血量为满或受到攻击。



阿斯克勒皮俄斯：【空格键】除了之前说的那种用途外，还可以让你在空间的边缘进行穿梭。当你在空间的最左边或最右边时，按下左键或右键的同时按下空格键，就可以穿梭到另一边的边界。这个举措会大量消耗耐力，要注意哦。

当玩家处在屏幕左右边界时，同时按下方向键与空格键可穿梭至屏幕另一边。

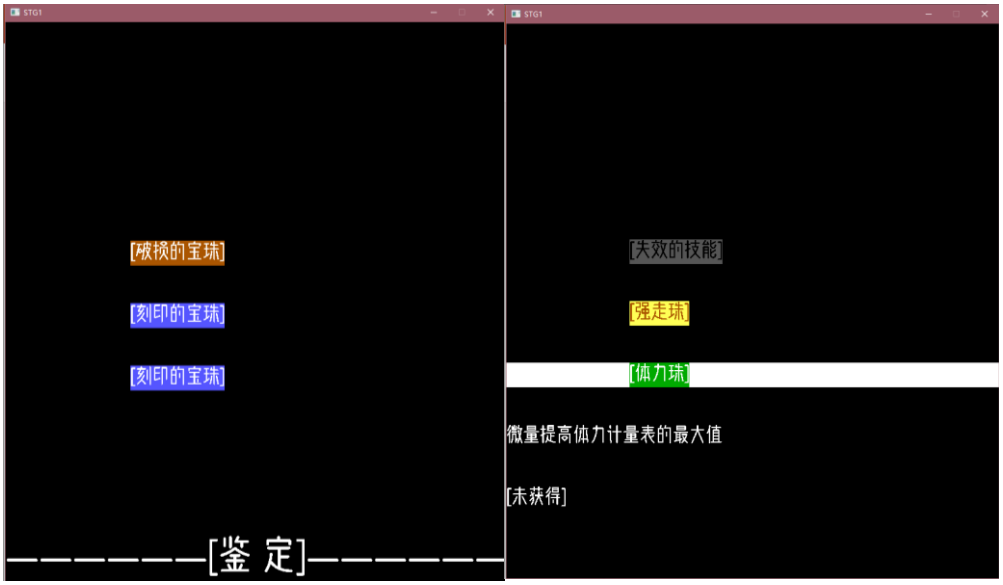
阿斯克勒皮俄斯：在躲避的同时，我们也要进行反击，按下【Z】键加速对瘟疫进行瓦解。瘟疫的瓦解度会反应在左上角的圆环中，随着瘟疫被逐步瓦解，圆环的红色部分将逐步收敛，当红色部分消失时，瘟疫完全瓦解，当然，主动加速瓦解是要消耗耐力的。因此，倘若耐力不足的话，就安心躲弹幕吧！

玩家按下【z】键可进行攻击。攻击命中敌人时会加快游戏进程

阿斯克勒皮俄斯：危险！你的体力计量表下降到零界点以下了！现在是你一场手术失误后 唯一一次 挽救机会！在被命中的一瞬间按下【空格键】，你将以一半体力值的状态返回，并且，你会处于耐力回复速度下降的异常状态，但总比力尽倒下好，不是吗？要在什么时机按下空格键？要具体说的话……就是现在！！！

当玩家体力值清空时，每一局有一次机会【DEAD KILL】。在死亡的一瞬间按下空格，玩家可以以一半体力值的状态返回，并且会处于耐力回复速度下降的异常状态。

一局游戏结束后，玩家可看到自己目前的分数。那之后进入剧情对话动画。在动画结束后玩家可随机获得三个技能，并从这三个技能中做出选择，选择一个技能强化主角。



以下为技能表：

[体力恢复量上升]　提高受到伤害后的生命回复数值
[耐力增强]　微量提高耐力计量表的最大值
[体力增强]　微量提高体力计量表的最大值
[无伤]　当体力值为满时，攻击获得的分数微量提升
[怨恨]　当体力计量表中存在红色部分时，子弹发射频率大幅上升
[超回复力]　体力值持续极微量回复
[加速再生]　攻击命中敌人后微量提升体力值
[回避性能]　微量缩小判定点大小
[精灵加护（伪）]　极微量降低受到的伤害

此外，玩家在游戏中途按下【esc】键或鼠标左键可暂停游戏。

暂停时可查看当前体力值与耐力值，以及所获得的技能

玩家在受到伤害后，减少的体力值中一小部分会存储在体力红值中。当一段时间没有受到伤害后，体力将缓慢恢复至红值阈值。

在长按【x】键出现蓄力条后，待蓄力条满值时按下【z】键，可发动时长约为 40 毫秒的“居合斩”。居合斩期间，人物受到伤害将大幅下降，并且居合斩期间若受到伤害则将恢复所有耐力值。

总游戏计划一共有 6 个关卡，目前实装至第四关（一半）；

1.3 游戏分析

（由于游戏机制过于复杂，于此赘述则显格外冗长，故笔者将主要描述界面的刷新，子弹飞行、人物及敌人的移动，子弹命中判定，背景图片运动，实体的生成与删除）

（1） 首先，本游戏采用持续刷新数值这一方法以保证变量的时效性。游戏的帧率，我选择为 60 帧，在适应人眼的同时保证游戏所占用的资源在一般电脑所能承受的范围之内。我写了一个高精度计时函数“blanktime（）”在游戏循环两次之间进行补帧。通过 while（true）来控制游戏流程的循环。

（2） 子弹的飞行、人物及敌人的运动方面，我使用了结构体以方便处理。在子弹 bullet 结构体、实体 object 结构体中我均添加了 vangle 与 velocity 变量表示该物体当前的运动方向与速度，并在 Vmove（）函数中参考这两个变量对子弹和实体结构体中的 x 与 y 变量加以加工处理。

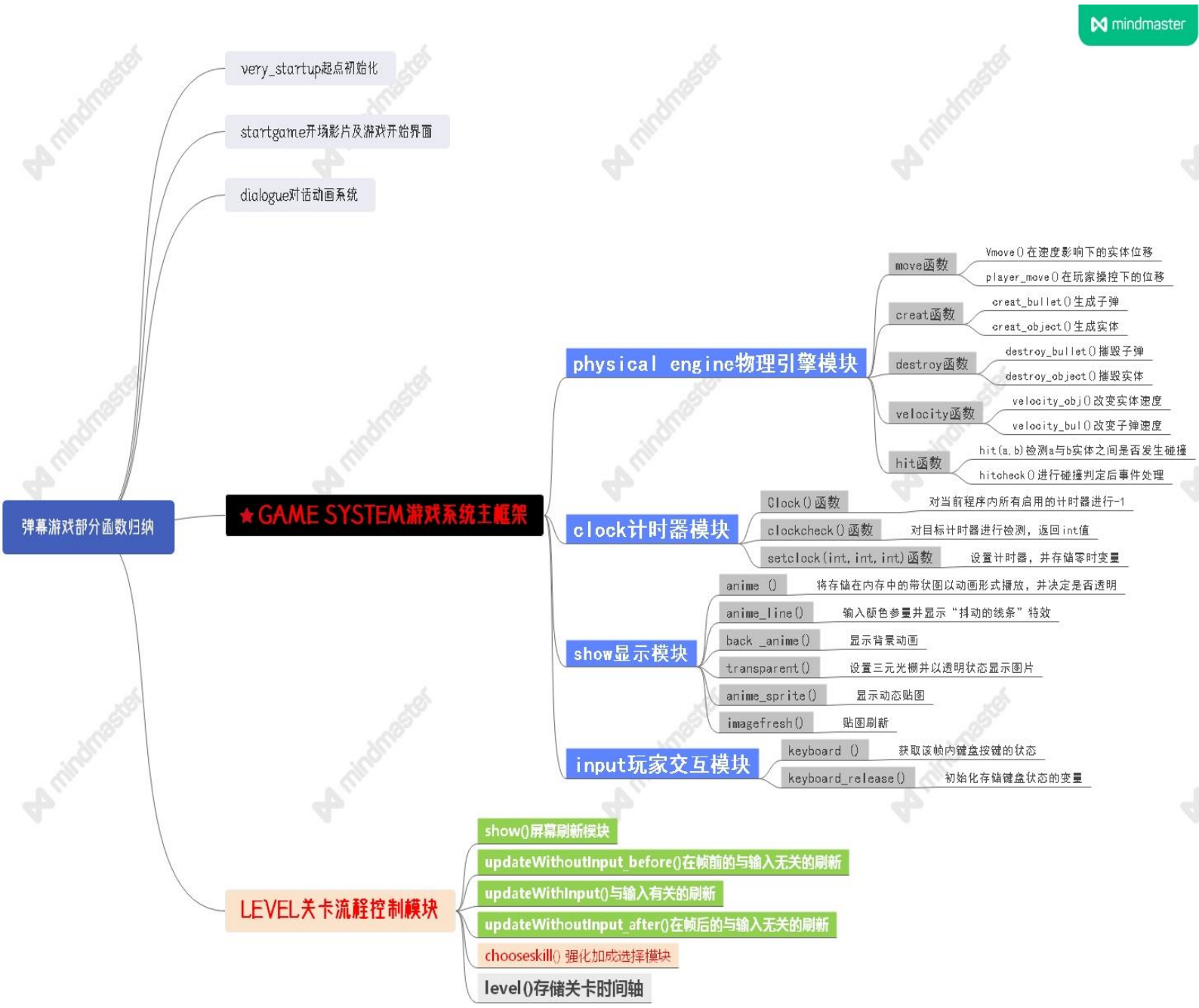
（3） 子弹命中判断，我在实体及子弹结构体中引用了 hitbox 变量作为一个简单的碰撞箱检测参数。当两个实体之间的距离小于这两个实体的 hitbox 之和时，则视为这两个实体发生碰撞。这个功能的实现我将它放在了 hitbox_check（）函数中。

（4） 关于背景图片，我添加了景深和相对运动效果。背景图片始终保持 y 轴循环运动状态，同时会随着玩家的移动而轻微位移。这样能够让画面更有动感，玩家也会感到更加舒适。

（5） 实体的生成与删除，我选择用结构体替换法。在游戏初始化阶段我声明了数组成员为 1000 的子弹结构体数组和成员为 100 的实体数组。在这些结构体中所包含的

isused 变量用来代表当前结构体的状态：isused=1 时该结构体启用，isused=0 时该结构体关闭。当结构体关闭时，位置刷新及贴图刷新将不再刷新该个体。

2．概要设计



模块划分与数据结构模型及物理引擎中部分功能函数

3.详细设计

（同理，此部分我只选择一小部分函数及模块的设计思路）

3.1：流程控制函数示例

3.1.1 高精度计时器 blanktime（）设计思路

为实现对游戏帧率的精确控制，我制作了一个通过读取 CPU 主频和 CPU 运作次数的函数，来实现对游戏帧率的皮秒级控制：

```
void blanktime()
{
	static double temp;
	QueryPerformanceCounter(&tickEndTime);//获取帧结束滴答
	QueryPerformanceFrequency(&cpuFreq);//获取 cpu 频率
	lasttime = (((double)tickEndTime.QuadPart - (double)tickBeginTime.QuadPart) / (double)cpuFreq.QuadPart)*1000;//获取该帧从开始到结束总共消耗的毫秒数
	sleeptime = (double)(1000.0 / fps) - lasttime;//获取需要 sleep 的毫秒数
	if (sleeptime > 0 && sleeptime < 1000) Sleep(sleeptime);//Sleep 对应的毫秒数
	QueryPerformanceCounter(&tickBeginTime);//获取帧开始滴答
	TRUE_FPS = 1.0/(((double)tickBeginTime.QuadPart - temp) / (double)cpuFreq.QuadPart);
	temp = tickBeginTime.QuadPart;
	total_flame++;
}
```

这个计时器的原理很简单，首先，通过 QueryPerformanceFrequency 读取 cpu 的主频并赋值进全局变量 cpuFreq，那之后，用 QueryPerformanceCounter(&tickEndTime)获取帧结束时的 cpu 总工作次数，并与上一个循环中通过 QueryPerformanceCounter(&tickBeginTime)获取的 cpu 总工作次数以及 cpufreq 变量通过简单的运算得到上一帧已经经过的时间（单位：毫秒），再用 sleep 补全目标帧率的剩余毫秒数即可达到高精度控制。

3.1.2 关卡控制器 level（）函数实例：

```
void level() {
	leveljumpout = 0;
	while (true)
	{
		if (leveljumpout == 1)break;//判定是否跳出大循环
		loadimage(&back[0], _T("image\\background\\2.png"));//加载背景图片
		nowtime = time(NULL);
		int deadkill = 1;
		standmpup = 3;
		hp = maxhp;
		mp = maxmp;
		int i = 0;
		int t = 0;
		qte = 0;
		qterand[0] = rand() % 135;
		qterand[1] = 180 + rand() % 135;
		(float)lt;
		leveltime = 21600;
		fps = 60;
		startup();
```

```
object[1].y = -100;
while (true)
{
    if (leveljumpout == 1)break; //判定是否跳出大循环
    lt = 21600 / fps - leveltime / fps;//设置 lt 为该关卡所经过的秒数
```

//

在此区间设置时间轴循环

例：

```
if (lt > 35 && lt < 81) {//当关卡经过的秒数在 35 与 80 之间时
    if ((int)leveltime % 10 == 0&&(int)leveltime%360<=80) {
        t += 10;
        if (t > 90)t -= 93;
        for (i = 0; i < 9; i++) {
            create_bullet(5, 0, i * 100, t * 10, 45, 7, 5, 0xfffa57, -1);//生成子弹
            create_bullet(5, 0, i * 100, t * 10, 135, 7, 5, 0xfffa57, -1);
        }
    }
    if ((int)leveltime % 360 == 80)setclock(-1, 5, 10,1);
}
```

//

////////////////当关卡时长为 0 时的操作//////////////////

```
if (leveltime <= 0 && leveltime > -60) {
    setlinecolor(WHITE);
    for (i = 0; i < 900; i++) {
        circle(object[1].x, object[1].y, i);
        FlushBatchDraw();
        leveltime = -60;
    }
    destory(object[1]);
    for (i = 0; i <= bulno; )
        destory(bullet[i]);
    lasttime = time(NULL) - nowtime;
}
```

////////////////当血量为 0 时的操作//////////////////

```
if (hp <= 0 && deadkill) {
    for (int i = 0; i < 20; i++) {
        setlinecolor(BLACK);
        for (int t = 0; t < 45; t++)circle(player.x, player.y, 900 - (i * 45 + t));
        if (GetAsyncKeyState(VK_SPACE) & 0x8000) {
            standmpup = 2;
```

```
                        mpup = standmpup;
                        hp = maxhp / 2;
                        deadkill = 0;
                        for (int i = 0; i <= bulno; )
                            destory(bullet[i]);
                        setlinecolor(WHITE);
                        for (int i = 0; i < 20; i++) {
                            for (int t = 0; t < 45; t++)circle(player.x, player.y, i * 45 + t);
                            FlushBatchDraw();
                            blanktime();
                        }
                        break;
                    }
                    FlushBatchDraw();
                    blanktime();
                }
            }if (hp <= 0) {
                setlinecolor(BLACK);
                destory(object[1]);
                for (i = 0; i <= bulno; )
                    destory(bullet[i]);
                for (i = 0; i < 900; i++) {
                    circle(player.x, player.y, i);
                    FlushBatchDraw();
                }
                dialogue_lost();
                break;
            }


//////////////////每一个循环都必须进行的更新//////////////////
            updateWithoutInput_before();//与操作无关的更新
            updateWithInput();//与操作有关的更新
            show();//刷新界面
            updateWithoutInput_after();//与操作无关的更新
            leveltime--;//关卡时间-1 帧

        }
    }
}
```

3.2：界面显示相关函数示例

3.2.1 imagefresh()贴图刷新函数设计思路：

该函数通过循环来达到批量刷新贴图的功能。由于子弹的刷新比实体的刷新算法要复杂得多，此次仅展示实体刷新的函数部分。

```
#define Sprite struct sprite
Sprite//该结构体专门用来存储长带图信息，在 object 结构体中被包含。
{
	int speed = 1;//动画速度
	int clock = speed;//帧计时器
	float zoom = 1;//缩放倍率
	float angle = 0;//贴图角度
	IMAGE* img[2];//贴图与遮罩层对应的地址指针
	int strip = 8;//总帧数
	int frame = 1;//当前帧数
};

void imageFresh()
{
	int i,d;
	for (d = 1; d <= 5; d++) {

	for (i = 0; i < objno + 1; i++) {
//objno 为当前激活的总实体数，当实体被创建或删除时会随之改变。
	if (object[i].depht == d)
	{
		if (object[i].sprite.clock == 0) {//判定增帧计时器是否为 0
			object[i].sprite.clock = object[i].sprite.speed;
			if (object[i].sprite.frame != object[i].sprite.strip) //判定是否刷新动画的下一帧
				object[i].sprite.frame++;
			else
				object[i].sprite.frame = 1;
		}
		transparent(i);//显示透明帧图片
		object[i].sprite.clock--;//帧计时器-1


	}
}
```

3.2.2 transparent()显示透明动态贴图的函数设计思路:

该函数通过读取保存在 sprite 结构体中的内容，切换输出的图片范围在总长带图中的区域来达到输出动态透明图片的效果。

```
void transparent(int i)
{
	IMAGE I = *object[i].sprite.img[0];
	int tstrip = object[i].sprite.strip;//获取目标 sprite 的总帧数
	int tframe = object[i].sprite.frame; //获取目标 sprite 的当前帧编号
	int widt = I.getwidth() / tstrip;
	int heig = I.getheight();

	putimage(
		object[i].x - widt / 2,//sprite 对应实体位置
		object[i].y - heig / 2,
		widt, heig,
		object[i].sprite.img[1],
		(tframe - 1) * widt, 0, SRCAND);//  显示蒙版
	putimage(
		object[i].x - widt / 2,
		object[i].y - heig / 2,
		widt, heig,
		object[i].sprite.img[0],
		(tframe - 1) * widt, 0, SRCPAINT);//显示图片

}
//该函数须结合上一个函数使用
```

3.3：物理引擎函数示例

```
#define Object struct object
Object//该结构体存储实体的位置，对应的 sprite 结构体以及运动参数等数据
{
	int NO = -1;
	int isused = 0;//实体启用开关
	float x = 0.0;//物体 x 坐标
	float y = 0.0;//物体 y 坐标
	double velocity = 0.0;//物体速度
	double Vangel = 0.0;//物体运动方向
	Sprite sprite;//物体所对应的精灵
```

```
    int hitbox = 0;//碰撞盒子
    int depht = 5;//实体深度
    int clock[5] = { -1,-1,-1,-1,-1};//旗标（计时器）
};
```

3.3.1 Vmove()在速度值影响下的 object 实体位移函数设计思路

读取 object 结构体中的 velocity 速度大小变量和 Vangel 速度矢量方向等实参，通过三角函数运算，将位移值赋值于 object 实体中 x 与 y 变量中。通过循环实现批量位移。
（bullet 结构体同理）

```
void Vmove()
{
    int i;
    for (i = 0; i <= objno; i++) {
        //if (object[i].isused == 0)
            //break;

        object[i].x = object[i].x + object[i].velocity * cos((object[i].Vangel / 180) * pi);
        object[i].y = object[i].y + object[i].velocity * sin((object[i].Vangel / 180) * pi);
    }for (i = 0; i <= bulno; i++) {

        bullet[i].x = bullet[i].x + bullet[i].velocity * cos((bullet[i].Vangel / 180) * pi);
        bullet[i].y = bullet[i].y + bullet[i].velocity * sin((bullet[i].Vangel / 180) * pi);
    }
}
```

3.3.2 hit()判定一个实体与一个子弹之间是否发生碰撞的函数

原理很简单，通过对比 hitbox 与距离实现判定

```
int hit(Object a,Bullet b)
{
    int distance = sqrt(pow((a.x - b.x), 2) + pow((a.y - b.y), 2));
    int totalhitbox = a.hitbox + b.hitbox;
    if (totalhitbox <= distance) {
        return 0;
    }
    else {
        return 1;
    }
}
```

3.3.3 hitcheck()利用 hit 函数在碰撞发生后执行事件的函数

```
void hitcheck() {
	int i = 0, t = 1;
	for (i = 0; i <= bulno; i++) {
		if (bullet[i].type == -1) {
			for (t = 1; t <= objno; t++) {
				if (hit(object[t], bullet[i])) {//若两个实体发生碰撞
					destory(bullet[i]);
					score = score + 10;//分数增加
					if (skillnumber[4] && hp == maxhp)score = score + skillnumber[4] * 2;//发动技能 4 效果
					if (skillnumber[7]&&hp<maxhp&&hpup<maxhp)hpup++;//发动技能 7 效果

					leveltime = leveltime - atk;//缩短关卡时间
				}
			}
		}
		else
		{
			if (hit(player, bullet[i])) {
				destory(bullet[i]);
				if (cutclock == 60) {
					for (int j = 0; j < 20; j++) {
						if ((GetAsyncKeyState(0x5a) & 0x8000)) {
							dmgrate = 0.3;
							break;
						}
						blanktime();
					}
				}
				hp = hp - DMG*dmgrate;//对玩家造成伤害
				hpup = hp + (float)DMG * hpuprate*dmgrate;//设置体力恢复量数值
				score -= 1000;//分数减少
				setbkcolor(RED);
				cleardevice();
				if (cutrush < 2) {
					player.y = player.y + 30;
					for (int t = 0; t <= bulno; )
						destory(bullet[t]);
```

```
                }
                else
                    mp = maxmp;
                FlushBatchDraw();
                hpclock = 180;
            }
        }

    }

}
```

3.4 计时器相关函数示例

3.4.1 Clock()控制实体与全局计时器变量的函数设计思路

通过循环实现所有的 clock 变量在一帧中-1。

```
void Clock()
{
    int i,t;
    if (shootclock > -1)shootclock--;
    if (hpclock > -1)hpclock--;
    if(cutclock>-1)cutclock--;
    if (cutrush > -1)cutrush--;
    for (i = 0; i < objno + 1; i++) {
        if (object[i].isused == 0)break;
        for (t = 0; t < 5; t++) {
            if (object[i].clock[t] == -1)continue;
            object[i].clock[t]--;
        }
    }
    for (i = 0; i < bulno + 1; i++) {
        if (bullet [i] .isused == 0)break;
        for (t = 0; t < 10; t++) {
            if (bullet[i].clock[t] == -1)
                continue;
            bullet[i].clock[t]--;
        }
    }

}
```

3.4.2 setclock()设置某一 bullet 实体中的某一个计时器的值

通过读取含义为【目标实体子弹组】【目标实体中计时器编号】【目标计时器值】【零时参数】三个形参，对目标子弹实体的计时器进行设置。
【目标实体子弹组】用来决定该次计时器设定适用的子弹群体。【子弹组】在子弹生成的时候被赋予数值。
【零时参数】用来存储与计时器相对应的操作值。例如某个子弹在计时器归零后改变运动角度，此处用来填写改变的角度值，倘如需要改变速度，此处则填写需要改变的速度值。不同的计时器编号代表不同的改变效果。

```
void setclock(int group, int clocknumber, int clocktime,int temp) {
	int i;
	for (i = 0; i <= bulno; i++)
		if (bullet[i].group == group) {
			bullet[i].tempdate = temp;
			bullet[i].clock[clocknumber] = clocktime;
		}
}
```

4.感想与小结

邵远航

首先作为组长，我要向我的组员们致上最诚恳的歉意。由于鄙人的分工不当，导致游戏的设计与制作全部由我完成了，导致组员们失去了一次宝贵的实践机会。制作游戏是一件令人成瘾的工作，在制作的过程中，制作者可以获得巨大的乐趣，希望组员们将来能够多多把握住这种机会。同时感谢石子悦同学优秀的文案剧本及在剧本录入上所花费的心血，你的文案真的很优秀，感谢你的付出。

我对学校给的这次游戏制作实践的机会十分感激。我曾经是一个独立游戏社团的发起者，后来因为一些管理原因导致社团关闭，曾经那种熬夜设计游戏、修改 bug 的乐趣也随之一去不复返。而这次实践让我久违的重温了那种感觉。从构思游戏玩法，到将其付诸实践，最后修改 bug 将其在游戏内实装，这种乐趣是非同凡响的。

最后，这个游戏到此刻为止仅仅是一个 demo 的不完全版本，完整剧情及流程我将在之后的工作中实装，请关注游戏官方网站 http://doyagame.rthe.net/获取最新消息！
源代码已开源，分享在个人 github 仓库中，感兴趣可前去下载浏览，并告诉我你宝贵的意见与建议！git 地址：

组员感想：

石子悦:

经过了这几周的实践与学习，本人不仅巩固了上学期所学的 c 语言相关的知识，同时也更加了解了 vs 在运用上的一些技巧。通过向老师的询问及在制作过程中与组长的交流的方式，我认识了更多的函数及语言在游戏中的运用方式，收获颇多。

郑佳诚:

这次的程序设计让我受益颇丰，作为一个才接触 c 语言的新人来说，能够做出一款游戏是非常有成就感的，没想到经过一个学期的学习我们也可以制作游戏。跟着有经验的组长我学到很多。

成员分工：

邵远航【组长】：游戏设计 游戏制作 素材制作
石子悦**【最佳组员】**：剧本编写 人物对话部分录入

《程序设计报告与总结》文档撰写：3-3 组全体成员

成员列表：邵远航、石子悦、孔世龙、郑佳诚、张强

此致