实验报告

敖伟智 21307130326

Task1:

```
#define Vol 2
```

```
string bgm_file = audio_dir + "bgm.mp3";
// TODO: add bgm
/* Your code here */
Audio audio_bgm(bgm_file,true,Vol);
audio_bgm.play_loop();
```

定义 Vol 为 2，方便统一更改音量

创建 Audio 对象 audio_bgm，使用 bgm_file 作为调用文件，循环开启，音量为 Vol;

调用对象函数 play_loop 播放音乐

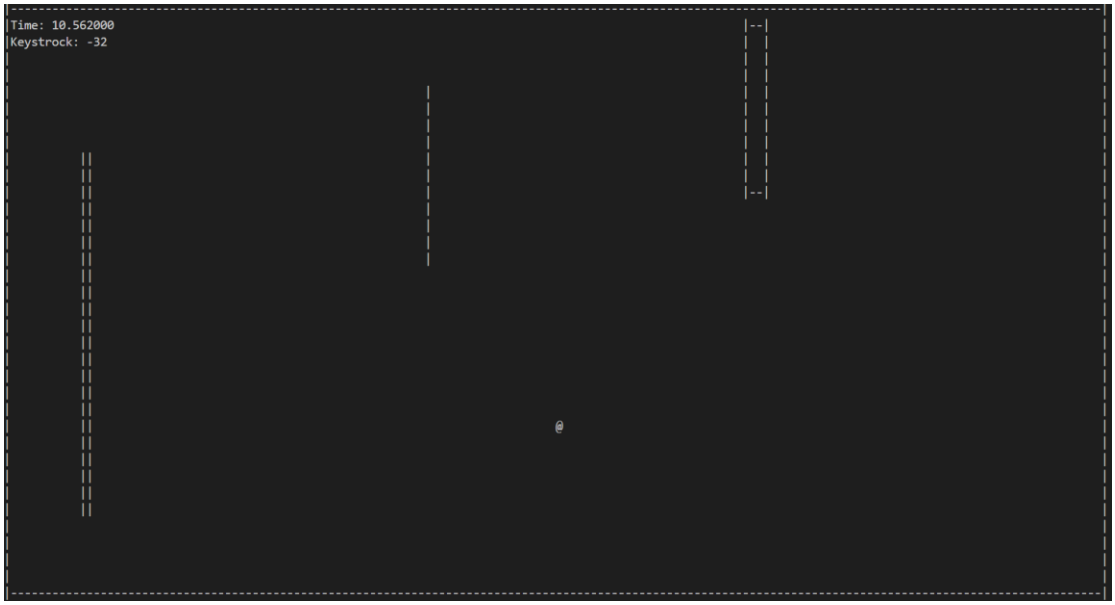
```
// TODO: stop bgm and add game over audio
/* Your code here */
audio_bgm.stop();
string end_music = audio_dir + "applause.mp3";
Audio audio_end_bgm(end_music,false,Vol);
audio_end_bgm.play_once();
```

在结束时，调用对象 audio_bgm 中 stop 函数停止播放音乐

创建 Audio 对象 audio_end_bgm 并调用函数 play_once 播放一次结束鼓掌音乐

Task2:

运行截图：



运行逻辑：

```
Bird* bird = new Bird(d.get_cols() / 2, d.get_rows() / 2, 0, 0, 0, 0);
Border* border = new Border(0, 0, d.get_cols(), d.get_rows());
```

先初始化鸟和边框

```
//suppot for music
string bgm_file = audio_dir + "bgm.mp3";
Audio audio_bgm(bgm_file,false,Vol);
audio_bgm.play_loop();

string flap_file = audio_dir + "flap.mp3";
Audio audio_flap(flap_file,false,Vol);

string collode_file = audio_dir + "collide.mp3";
Audio audio_collide(collode_file,false,Vol);
```

创建不同 Audio 对象做音频文件播放的准备

```
while (true)
```

进入循环：

```
int c = d.get_char(50);
d.clear();
if (c == 'q') {
    break;
}
else if(c == 'z' || c == 'x'){
    flag_for_flap = false;
}
else if (c == ' ') {
    if(flag_for_flap == false){
        audio_flap.play_once();
    }
    flag_for_flap = true;
}
```

首先判断按键，用 flag 辅助在上一次不是空格键时按下空格播放 flap 音乐
在按下 q 时退出循环结束游戏

```
void add_rectangle(ObjPool& pool, Display& d) {
    // TODO: add a moving_rectangle to the pool
    /* Your code here */
    int width = rand()%5 + 1;
    int limit_height = (int)(0.66 * d.get_rows());
    int height = rand()%limit_height +1;
    MovingRectangle* Mov_Rec = new MovingRectangle(d.get_cols() - 1 - width,rand()%(int)(0.33*d.get_rows()),width,height,-1,0);
    pool.emplace_back(Mov_Rec);
}
```

```
#define speed_to_create 50
```

```
count++;
if(count%speed_to_create == 0){
    count++;
    add_rectangle(obj_pool, d);
}
```

Define 一个 speed_to_create 表示每多少帧创建一个长方形：用于控制间距增加或减小难度
每隔 speed_to_create 帧在屏幕右端创建长方形。

```
for(vector<Object*>::iterator it1 = obj_pool.begin(); it1 != obj_pool.end(); it1++){
    if(typeid(**it1) == typeid(MovingRectangle)){
        if(!((**it1).get_valid())){
            delete (*it1);
            obj_pool.erase(it1);
        }
    }
}
```

如果长方形超出边界，将其从对象池中删除

```
void detect_collision(ObjPool& pool) {
    // TODO: detect collision between each pair of objects
    /* Your code here */
    // for(vector<Object*>::iterator it1 = pool.begin(); it1 != pool.end(); it1++){
    //     //Detect Border
    //     if(typeid(**it1) == typeid(Border)){
    //         for(vector<Object*>::iterator it2 = pool.begin(); it2 != pool.end(); it2++){
    //             if(it2 == it1){
    //                 continue;
    //             }else{
    //                 (**it2).detect_collide(**it1);
    //             }
    //         }
    //     }
    //     if(typeid(**it1) == typeid(Bird)){
    //         for(vector<Object*>::iterator it2 = pool.begin(); it2 != pool.end(); it2++){
    //             if(typeid(**it2) == typeid(Border)){
    //                 continue;
    //             }else if(typeid(**it2) == typeid(Bird)){
    //                 continue;
    //             }else{
    //                 (**it1).detect_collide(**it2);
    //             }
    //         }
    //     }
    // }

    for(vector<Object*>::iterator it1 = pool.begin(); it1 != pool.end(); it1++){
        for(vector<Object*>::iterator it2 = pool.begin(); it2 != pool.end(); it2++){
            if(it2 == it1){
                continue;
            }else{
                (**it2).detect_collide(**it1);
            }
        }
    }
}
```

```
detect_collision(obj_pool);
```

碰撞检测：（第一次写复杂了，有一些注释掉的 code）
在检测过程中若鸟有碰撞，将 Bird 的 valid 赋值为 false

```
if(!bird->get_valid()){
    audio_collide.play_once();
    break;
}
```

如果鸟碰撞了就播放 collide 音乐并跳出循环

```
for (auto& obj : obj_pool) {
    obj->draw(d);
```

在终端打印对象池中的每一个元素，完成游戏的一帧

```
d.put_string(1, 1, "Time: " + std::to_string(d.time() / 1000.0));
d.put_string(1, 2, "Keystrock: " + std::to_string(c));
d.refresh();
d.log();
```

在左上角打印 Time 和按键的 ascii 码并将这一帧保存到 log 文件中

```
audio_bgm.stop();
audio_end.play_once();
show_game_over(d,d.time()/1000.0);
```

跳出循环时结束 bgm 并播放结束音乐掌声