中山大学 SUN YAT-SEN UNIVERSITY 现代操作系统期末项目实验报告

一、 项目分工

学号	名字	角色	班级	职责	贡献
16340039	陈星宇	组长	下午班	负责物理引擎和刚体制作	25%
16340043	陈彦行	组员	下午班	负责游戏逻辑处理以及各种事件	25%
16340042	陈衍斌	组员	下午班	负责整合，修理一些 bug，撰写报告和录像	25%.
16340010	曾广韬	组员	下午班	负责游戏的基本设置和主菜单界面	25%

二、开发环境

Windows10, Visual Studio 2017, python2.7

三、 项目阐述

项目名称：重力版俄罗斯方块（Crazy Tetris）

项目简介：俄罗斯方块是我们每个人都耳熟能详的经典小游戏。试想一下，如果俄罗斯方块不是一格一格的下落，而是自由落体，那又别有一番乐趣。Carzy Tetris 带你体验全新的玩法。

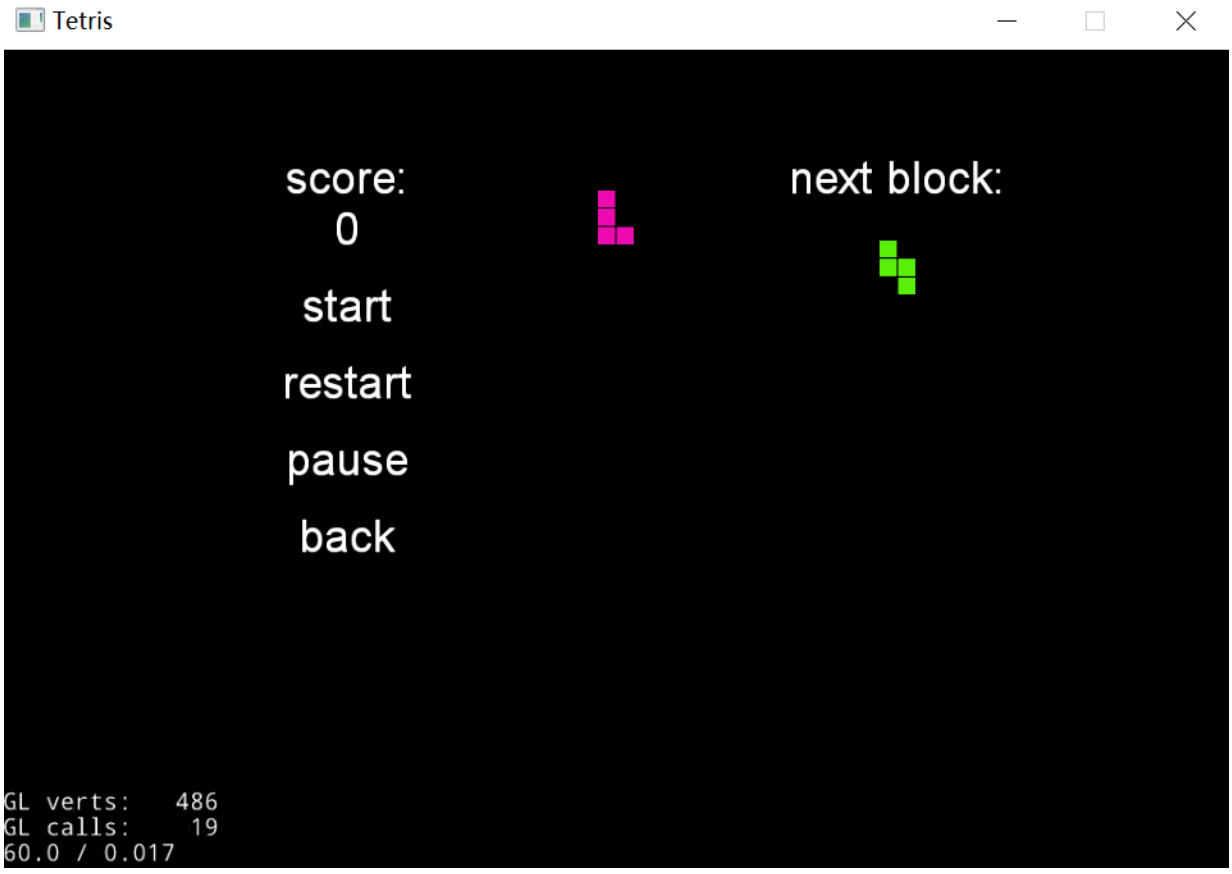
项目功能：与传统的俄罗斯方块相同，方块会从顶部下落，玩家需要控制方向来调整方块是他们在屏幕底部拼出完整的一条或几条，然后这些完整的横条会随即消失。没有被消除掉的方块不断堆积起来，一旦堆到屏幕顶端，玩家便告输，游戏结束。不同的地方在于，每个不同形状的板块是重力下落的，这意味着，板块不一定能够平稳地落到底部，比如 L 型板块有可能以这种方式落到底部：

项目亮点：这是一个全新版本的俄罗斯方块，不到 5 分钟你就会爱上介款游戏！

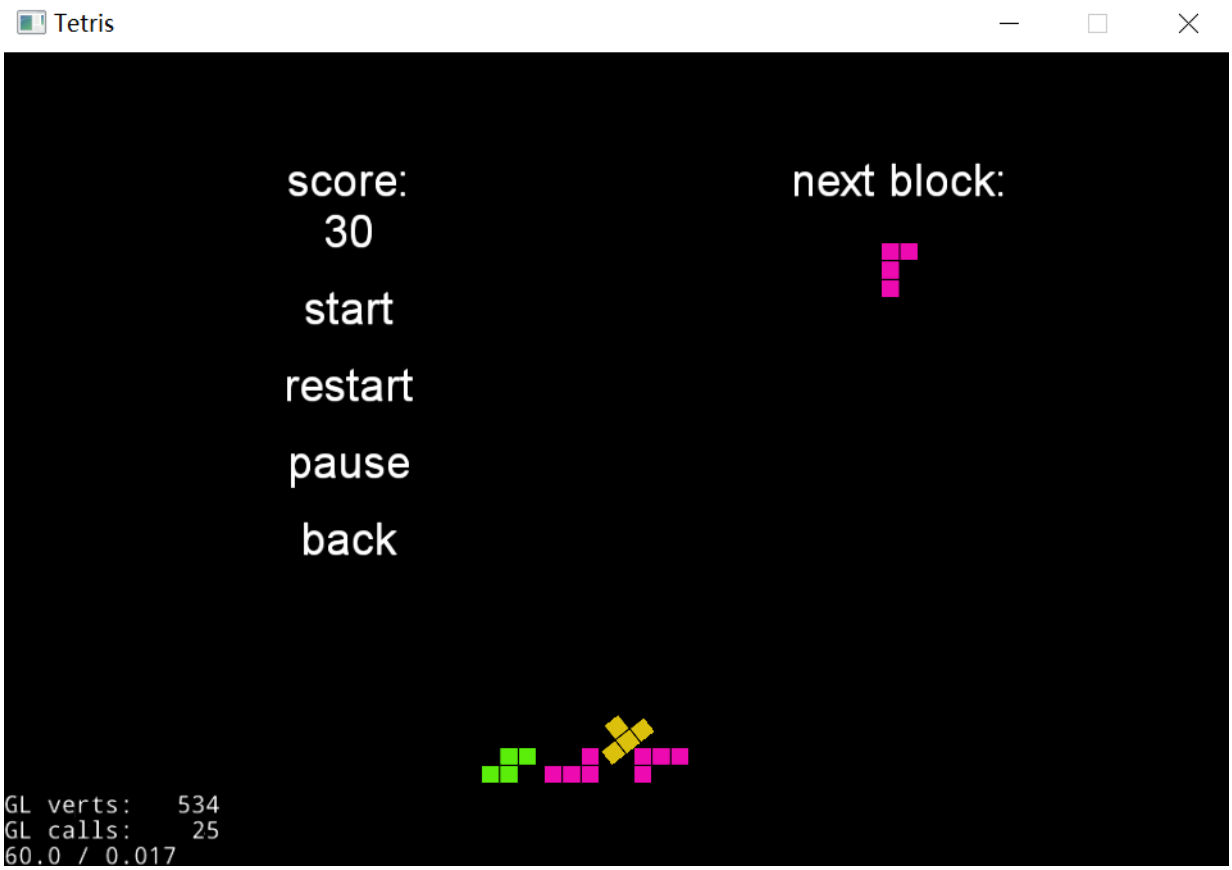
四、 项目展示

这个是主菜单界面：

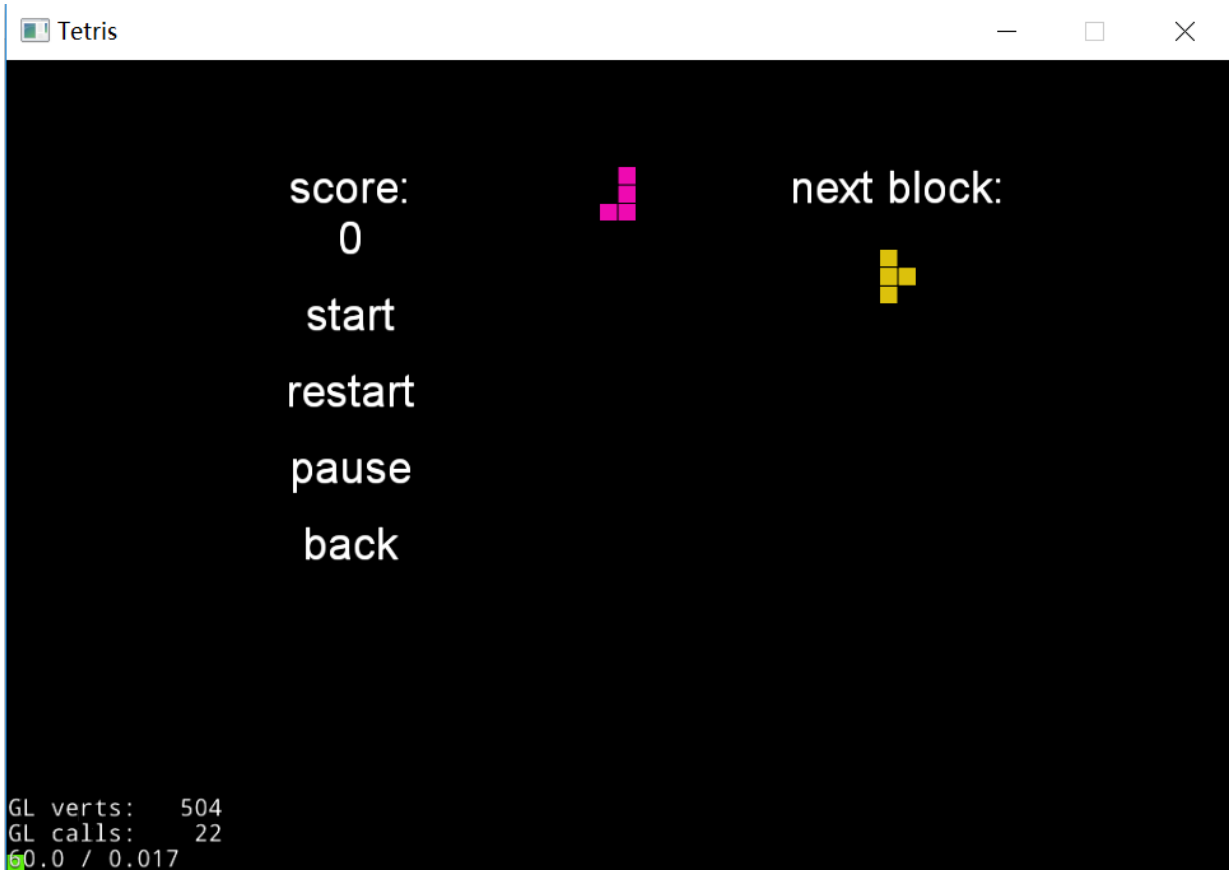
点击 start，进入游戏：



方块是重力下落，所以在底部会有各种各样的姿态：



点击 restart 会重置游戏界面：



点击 pause 会暂停。

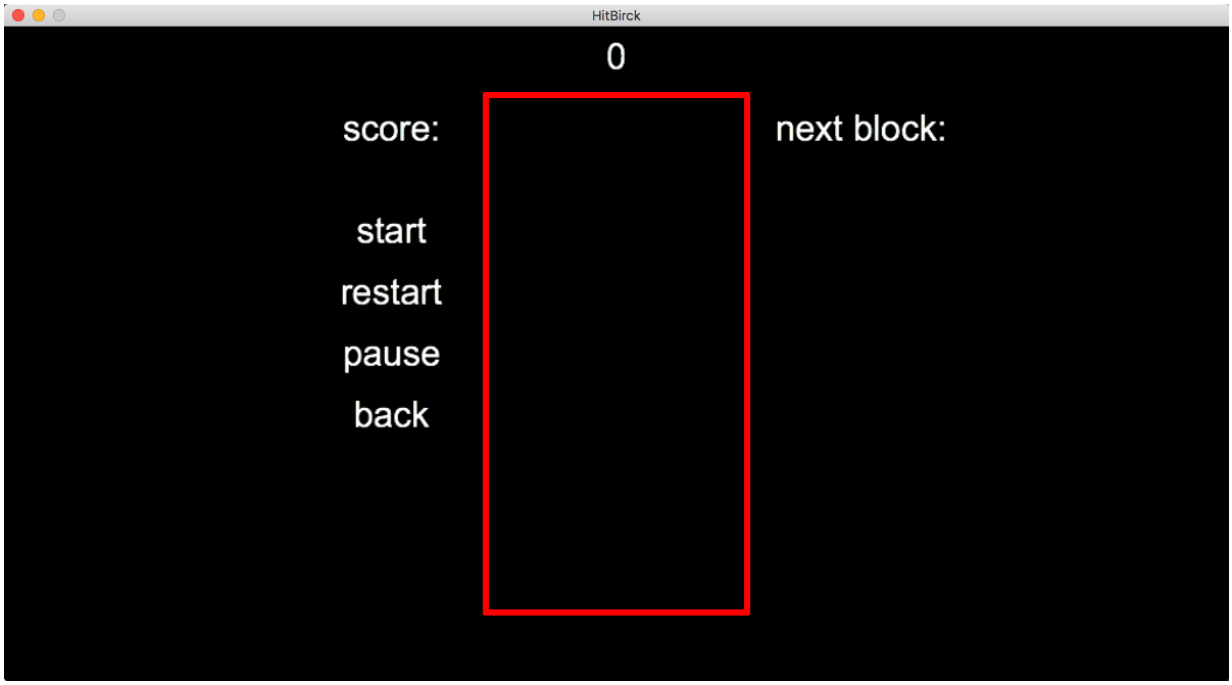
当游戏结束时会进入 GameOver 界面（因为一些特殊原因，这个游戏实在是太难

GameOver 了，具体界面可以在 Resources 文件夹查看）

五、 项目难点及解决方案

本次期末作业是实用 cocos2d-x 编写一款游戏。首先到底制作怎样的一款游戏就是一个难题，我们小组考虑了很久。首先我们想过做一款 rpg 类的游戏，不过 rpg 类的游戏玩法基本上已经被定型了，而且如果非要做的话，rpgmaker 的功能更加全面，能够做更复杂的 rpg 游戏而且使用起来方便，所以没必要用 cocos2d-x 来做（最重要的是没有什么创新性）。最终考虑了市面上各种各样的小游戏的玩法后，我们选择了一个几乎所有人的接触过的游戏——俄罗斯方块。不过显然，如果只做传统的俄罗斯方块，那就没什么意思了，毕竟虽然是经典，但是实在是老掉牙了。所以我们就想到：如果方块不是一格格下落。而是像现实世界一样有重力的呢？想一想感觉就很有意思。不过显然这样为本来平淡的俄罗斯方块增添了不少难度——每个板块不在一定是垂直下落了，这样就会出现方块不规则摆放的情况，从而导致消除一行的难度大大增加。不过换个角度来说，这样子这个游戏是不是更加具有趣味性和挑战性了呢?

接下来就是要着手制作了，首先我们需要制作菜单，也就是游戏的入口。这个主要时间花在界面布局上。这个游戏主菜单的界面很简单，就一张背景，两个按钮——开始游戏和退出游戏。因此这里主要就是添加背景以及菜单及菜单按钮的制作。界面布局如下：



制作完主菜单和场景，紧接着就是要制作游戏场景 GameView。游戏场景的创建和主菜单是一样的，其中的布景、UI 都可以参照主菜单场景做出来，难度较低。

然后就是刚体的创建。这个其实在之前的“打砖块”作业里面我们已经接触过了。考虑到俄罗斯方块有好几种形状，那么我们需要为他们每一个添加刚体。刚开始用 createPolygon 去创建多边形刚体报错，后来才知道这个只能创建凸多边形。但是方块的形状有好几个是凹多边形的，我们的小组成员当时没有找到好的解决方法，就索性直接它们固定为同一个模型的刚体，随机图片来生成（虽然显然是错误的方法），后来才想到用凸多边形来拼凑出各个形状的方法，但是由于时间紧迫，最终没能实现。

物理引擎方面，就本游戏而言，就是要让下落的方块拥有重力、速度、碰撞等基本的物理属性。而这些属性设置起来并不难。

接下来需要考虑游戏逻辑。主要难点在于需要考虑以下几个问题：

- 判断方块是否已经落下停止，并下落新的方块；
- 方块落下后，判断是否符合消行条件，进行消行并进行计分；
- 判断方块是否已经到达游戏顶部，如果是则游戏结束。

这些判断需要定时处理。所以这里我们需要用到 update()函数，对每一个条件进行一

些判断。具体的代码如下：

```
void GameScene::update(float dt) {
    if (currentBlock->getPositionX() >= visibleSize.width / 2 - 50
        && currentBlock->getPositionX() <= visibleSize.width / 2 + 50
        && currentBlock->getPositionY() >= visibleSize.height / 2 + 220)
    {
        gameOver();
    }

    if (is_pause == 0 && std::fabs(currentBlock->getPositionY() == lastPosition.y && is_next)) {
        score += 10;
        syncscore();
        buildNextBlock();
        is_next = 0;
    }
    else {
        lastPosition = currentBlock->getPosition();
    }

}
```

游戏结束后专门创建了个界面，主要写在 GameOver.cpp 里。

接下来就该实现各种各样的触发事件了。首先是键盘事件，我们需要重写一下方法：

```
void onKeyPressed(EventKeyboard::KeyCode code, Event* event);
void onKeyReleased(EventKeyboard::KeyCode keyCode, Event* event);
```

其中的 KeyCode 就是被按的那个键的编码。左右方向键的话，键盘按下后会给一个左右方向的速度，而在松开后会将速度置为 0：

```
if (code == EventKeyboard::KeyCode::KEY_A || code == EventKeyboard::KeyCode::KEY_LEFT_ARROW)
{
    currentBlock->getPhysicsBody()->setVelocity(Vec2(-100.0f, currentBlock->getPhysicsBody()->getVelocity().y));
} else if (code == EventKeyboard::KeyCode::KEY_D || code == EventKeyboard::KeyCode::KEY_RIGHT_ARROW)
{
    currentBlock->getPhysicsBody()->setVelocity(Vec2(100.0f, currentBlock->getPhysicsBody()->getVelocity().y));
}
```

向上的按键将会触发旋转事件：

```
else if (code == EventKeyboard::KeyCode::KEY_W || code == EventKeyboard::KeyCode::KEY_UP_ARROW)
{
    currentBlock->getPhysicsBody()->setAngularVelocity(-5.0f);
    isRotating = true;
}
```

向下的按键将会加速方块下落的速度：

```
else if (code == EventKeyboard::KeyCode::KEY_S || code == EventKeyboard::KeyCode::KEY_DOWN_ARROW)
{
    currentBlock->getPhysicsBody()->setVelocity(Vec2(currentBlock->getPhysicsBody()->getVelocity().x, -150.0f));
    isAccelerating = true;
}
```

然后按钮点击释放事件中恢复，那么基本的事件也就完成了。

六、 项目总结

这次的期末项目让我们学到了许多知识，首先这是我们对课内所学的知识点的巩固和应用。之所以这么说是因为之前的作业 TA 基本给出了框架，我们只需要在框架内把函数给补全即可。而这次需要我们自己来构建框架，这样难度就不仅仅是大了一点半点。一些之前学过的但是掌握程度非常不牢固的知识，比如物理引擎那部分还有刚体，在这次作业里算是非常好的巩固了一下，这让我们收益匪浅。当然，这次作业还有许许多多的 bug 没有解决，这说明我们的能力还有所欠缺，代码能力有待提高，还有很多东西没有真正掌握，还需要不断地学习。

总之，虽然课程结束了，但是我们的学习之路还远远没有结束，我们会继续深入学习并复习 uwp 和 cocos2d-x ~~