现代操作系统应用开发实验报告

姓名：谷雨

学号：16341005

实验名称：lab5 实验报告

一、参考资料

1. **cocos2dx[3.4](25)——瓦片地图 TiledMap**

   http://blog.51cto.com/shahdza/1613527

2. **cocos2d-x 自带 11 种粒子特效**

   https://blog.csdn.net/song_hui_xiang/article/details/8712240

3. **cocos2d-x 3.6 physics 构建小车**

   https://www.jianshu.com/p/a30c63bf8d33

二、实验步骤

请在这里简要写下你的实验过程。

第十三周：

本周任务中大部分实现代码均已给出。

主要任务有两部分

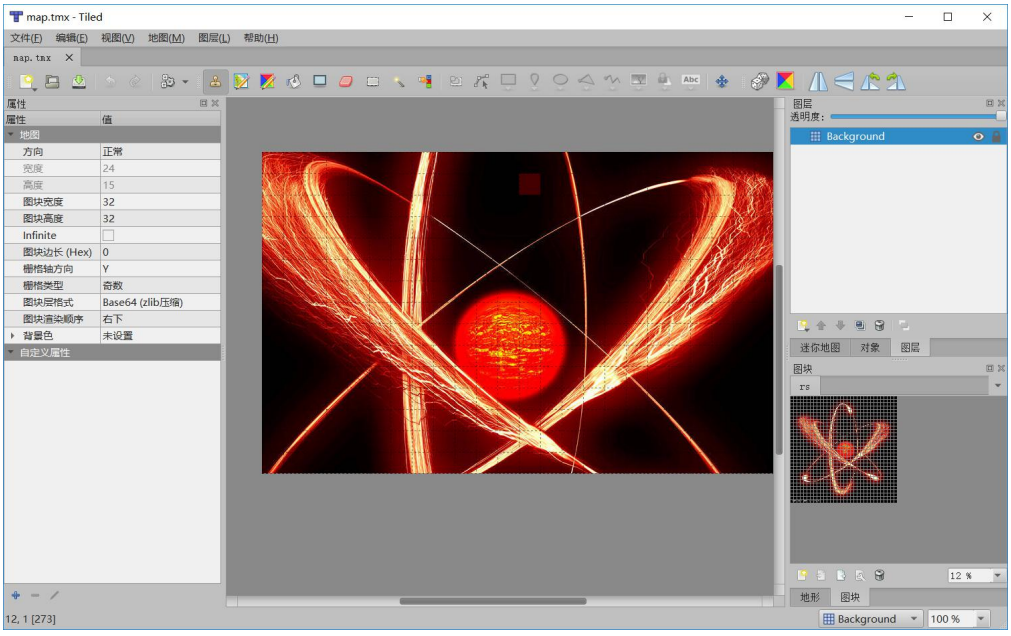
1. 本地文件的存储，读写。

   UserDefault 的使用

```
#include "Monster.h"
#define database    UserDefault::getInstance()

using namespace cocos2d;
```

```
    }
    Sprite* collision = fac->collider(attackRect);
    if (collision != NULL) {
        //
        fac->removeMonster(collision);
        if (pT->getPercentage() < 100)
        {
            pT->runAction(ProgressTo::create(2, pT->getPercentage() + 20));
        }
        dcount++;
        count->setString(std::to_string(dcount));
        if (database->isXMLFileExist()) {

            log("exist");
            log(database->getXMLFilePath().c_str());
            database->setIntegerForKey("value", dcount);
            database->flush();
        }
        else {
            log("notexist");
        }

    }
```

2. TileMap 的使用



```
void createMap() {
    TMXTiledMap * tmx = TMXTiledMap::create("map.tmx");
    tmx->setPosition(visibleSize.width / 2, visibleSize.height / 2);
    tmx->setAnchorPoint(Vec2(0.5, 0.5));
    tmx->setScale(Director::getInstance()->getContentScaleFactor());
    log("%1f,%1f", tmx->getContentSize().width, tmx->getContentSize().height);
    this->addChild(tmx, 0);
}
```

```
CCDictionary* message = CCDictionary::createWithContentsOfFile("tar.xml");    //读取xml文件，文件在Resources目录下
auto name = message->valueForKey("name");
auto num = message->valueForKey("num");
log((name->_string + '\n' + num->_string).c_str());
Label* label = Label::createWithSystemFont(name->_string+'\n'+num->_string, "Arial", 17);
//auto label = Label::createWithTTF(b, "fonts/arial.ttf", 24);
if (label == nullptr)
{
    problemLoading("'fonts/arial.ttf'");
}
```

1. 更换图片

```
auto sprite = Sprite::create("atom.jpg");
sprite->setPosition(50, 50);
sprite->setAnchorPoint(Vec2(0.5, 0.5));
this->addChild(sprite);// 添加到层
auto act= ScaleTo::create(0, 0.2);
sprite->runAction(act);
```

2. Eventlistner

```
//通过 lambda 表达式 直接实现触摸事件的回掉方法
listener1->onTouchBegan = [=](Touch* touch, Event* event) {
    auto target = static_cast<Sprite*>(event->getCurrentTarget());

    Point locationInNode = target->convertToNodeSpace(touch->getLocation());
    Size s = target->getContentSize();
    Rect rect = Rect(0, 0, s.width, s.height);

    if (rect.containsPoint(locationInNode))
    {
        log("sprite began... x = %f, y = %f", locationInNode.x, locationInNode.y);
        target->setPosition(((double)(rand() % 10)) / 10 *visibleSize.width+origin.x,
            ((double)(rand() % 10)) / 10 *visibleSize.height+origin.y);
        return true;
    }
    return false;
};
//将触摸事件绑定到精灵身上
_eventDispatcher->addEventListenerWithSceneGraphPriority(listener1, sprite);
```

第十四周：

小蜜蜂

要求 ：

利用键盘事件实现飞船左右移动。

利用键盘和触摸事件实现子弹发射。

用自定义事件实现：子弹和陨石相距小于一定距离时，陨 石爆炸，子弹消失。

游戏过程中有背景音乐，发射子弹、击中陨石有音效。

注意飞船、子弹的移动范围。

游戏结束飞船爆炸，移除所有监听器

主要流程：

1. 飞船的左右移动

按键监听，设立 bool 标志。

```
void Thunder::onKeyPressed(EventKeyboard::KeyCode code, Event* event) {
    switch (code) {
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
    case EventKeyboard::KeyCode::KEY_CAPITAL_A:
    case EventKeyboard::KeyCode::KEY_A:
        movekey = 'A';
        isMove = true;
        break;
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    case EventKeyboard::KeyCode::KEY_CAPITAL_D:
    case EventKeyboard::KeyCode::KEY_D:
        movekey = 'D';
        isMove = true;
        break;
    case EventKeyboard::KeyCode::KEY_SPACE:
        fire();
        break;
    }
```

调度器调度判断标志变量实现移动

```
void Thunder::update(float f) {
    // 实时更新页面内陨石和子弹数量(不得删除)
    // 要求数量显示正确(加分项)
    char str[15];
    sprintf(str, "enemys: %d", enemys.size());
    enemysNum->setString(str);
    sprintf(str, "bullets: %d", bullets.size());
    bulletsNum->setString(str);

    // 飞船移动
    if (isMove)
        this->movePlane(movekey);
```

2. 飞船撞到陨石爆炸

调度器调度，检查是否碰撞

```
oid Thunder::meet1(EventCustom* event) {
    int i = 0;
    bool flag = false;
    for (auto tar : enemys) {
        if (tar->getPosition().y < 20)
            flag = true;
        else if ((tar->getPosition() - player->getPosition()).getLength() < Vec2(30, 30).getLength())
            flag = true;
        i++;
        if (i ==20||flag)
            break;
    }
    if(flag)
    stopAc();
```

3. 预加载与播放音乐

```
//预加载音乐文件
void Thunder::preloadMusic() {
    // Todo
    auto audio = SimpleAudioEngine::getInstance();
    audio->preloadBackgroundMusic("music/bgm.mp3");
    audio->preloadEffect("music/explore.wav");
    audio->preloadEffect("music/fire.wav");
}

//播放背景音乐
void Thunder::playBgm() {
    // Todo
    SimpleAudioEngine::getInstance()->playBackgroundMusic("music/bgm.mp3", true);
}
```

加分项 ：

利用触摸事件实现飞船移动。（点击飞船后拖动鼠标）
陨石向下移动并生成新的一行陨石
子弹和陨石的数量显示正确

```
// 陨石向下移动并生成新的一行(加分项)
void Thunder::newEnemy() {
    // Todo
    double width = visibleSize.width /6,
        height = visibleSize.height - 50;
    for (Sprite* s : enemys) {
        if (s != NULL) {
            s->setPosition(s->getPosition() + Vec2(0, -50));
        }
    }
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", 3 - stoneType);
    stoneType++;
    stoneType %= 3;
    for (int j = 0; j < 5; ++j) {
        auto enemy = Sprite::create(enemyPath);
        enemy->setAnchorPoint(Vec2(0.5, 0.5));
        enemy->setScale(0.5, 0.5);
        enemy->setPosition(width * (j+0.5), height);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }

}
```

```
// 添加触摸事件监听器
void Thunder::addTouchListener() {
    // Todo
    auto touchListener = EventListenerTouchOneByOne::create();
    touchListener->onTouchMoved = CC_CALLBACK_2(Thunder::onTouchMoved, this);
    touchListener->onTouchBegan = CC_CALLBACK_2(Thunder::onTouchBegan, this);
    touchListener->onTouchEnded = CC_CALLBACK_2(Thunder::onTouchEnded, this);
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(touchListener, this);
}

// 鼠标点击发射炮弹
bool Thunder::onTouchBegan(Touch *touch, Event *event) { ... }

void Thunder::onTouchEnded(Touch *touch, Event *event) {
    isClick = false;
}

// 当鼠标按住飞船后可控制飞船移动 (加分项)
void Thunder::onTouchMoved(Touch *touch, Event *event) {
    // Todo
    if (!isClick)
        return;
    Vec2 delta = touch->getDelta();
    if(delta.x<0&& player->getPosition().x>30)
        player->runAction(MoveBy::create(0.1f, Vec2(delta.x,0)));
    else if (delta.x>0 && visibleSize.width-player->getPosition().x>30)
        player->runAction(MoveBy::create(0.1f, Vec2(delta.x, 0)));
}
```

第十五周：

要求：

控制板子左右移动

在顶部生成小砖块

使用关节固定球与板子

为板子、球、以及砖块设置物理属性

砖、球碰撞则消去砖头，球与地板碰撞则游戏结束

加分要求：

至少使用一种粒子效果（加在球上面，etc）

要点代码：

板子移动：

使用速度

```
// 左右
void HitBrick::onKeyPressed(EventKeyboard::KeyCode code, Event* event) {

  switch (code) {
  case cocos2d::EventKeyboard::KeyCode::KEY_LEFT_ARROW:
      this->player->getPhysicsBody()->setVelocity(Vec2(-200, 0));

    break;
  case cocos2d::EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
      this->player->getPhysicsBody()->setVelocity(Vec2(200, 0));

    // 左右移动
    // Todo
    break;
```

关节：

```
// 关节连接，固定球与板子
// Todo
void HitBrick::setJoint() {
    this->joint1= PhysicsJointPin::construct(ball->getPhysicsBody(), player->getPhysicsBody(),Vec2(0,0));
    m_world->addJoint(joint1);
}
```

设置物理属性：

```
player->setPosition(Vec2(xpos, ship->getContentSize().height - player->getContentSize().height*0.1f));
// 设置板的刚体属性
PhysicsBody* barBodyOne = PhysicsBody::createBox(player->getContentSize(),  PhysicsMaterial(100.0f, 1.0f, 0.0f));

barBodyOne->setDynamic(false);
barBodyOne->setCategoryBitmask(0xFFFFFFFF);
barBodyOne->setCollisionBitmask(0xFFFFFFFF);
barBodyOne->setContactTestBitmask(0xFFFFFFFF);
//把物体添加到精灵中
player->setPhysicsBody(barBodyOne);
//设置标志
player->setTag(1);


this->addChild(player, 2);
```

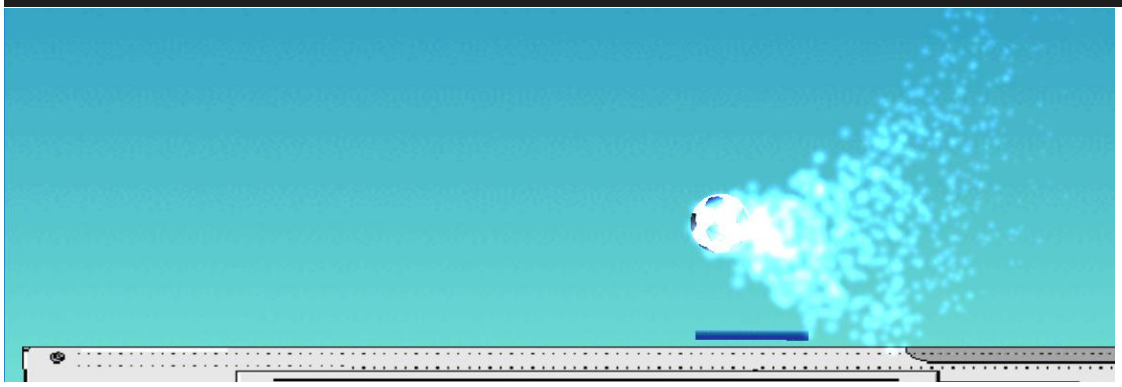
```
ball->setScale(0.1f, 0.1f);
// 设置球的刚体属性
PhysicsBody* ballBodyOne = PhysicsBody::createCircle(ball->getContentSize().width / 2, PhysicsMaterial(1.0f, 0.0f, 0.0f));
ballBodyOne->setCategoryBitmask(0xFFFFFFFF);
ballBodyOne->setCollisionBitmask(0xFFFFFFFF);
ballBodyOne->setContactTestBitmask(0xFFFFFFFF);
ballBodyOne->setGravityEnable(true);
ballBodyOne->setRotationEnable(false);
ball->setPhysicsBody(ballBodyOne);
//设置标志
ball->setTag(2);
ball->setAnchorPoint(Vec2(0.5, 0.5));
ParticleSystem* emitter = ParticleSystemQuad::create("particle_texture.plist");
emitter->setPosition(ball->getContentSize().width / 2, ball->getContentSize().height / 2);
emitter->setTotalParticles(1000);
emitter->setScale(3);
ball->addChild(emitter, 10);


addChild(ball, 3);
```

碰撞检测：

```
// 碰撞检测
// Todo
bool HitBrick::onConcactBegin(PhysicsContact & contact) {
    auto c1 = contact.getShapeA(), c2 = contact.getShapeB();
    auto sp1 = (Sprite*)(c1->getBody()->getNode());
    auto sp2 = (Sprite*)(c2->getBody()->getNode());
    if (!sp1 || !sp2)
        return false;
    if(sp1->getTag() == 3&& sp2->getTag() == 3)
        return false;
    CCLOG("%d", sp1->getTag());
    CCLOG("%d", sp2->getTag());
    if (sp1->getTag() == 3 ) {
        sp1->removeFromParentAndCleanup(true);
    }else if (sp2->getTag() == 3) {
        sp2->removeFromParentAndCleanup(true);
    }
    else if (sp1->getTag() == 11&&sp2->getTag()==2 || sp2->getTag() == 11 && sp1->getTag() == 2)
        GameOver();
    return true;
}
```

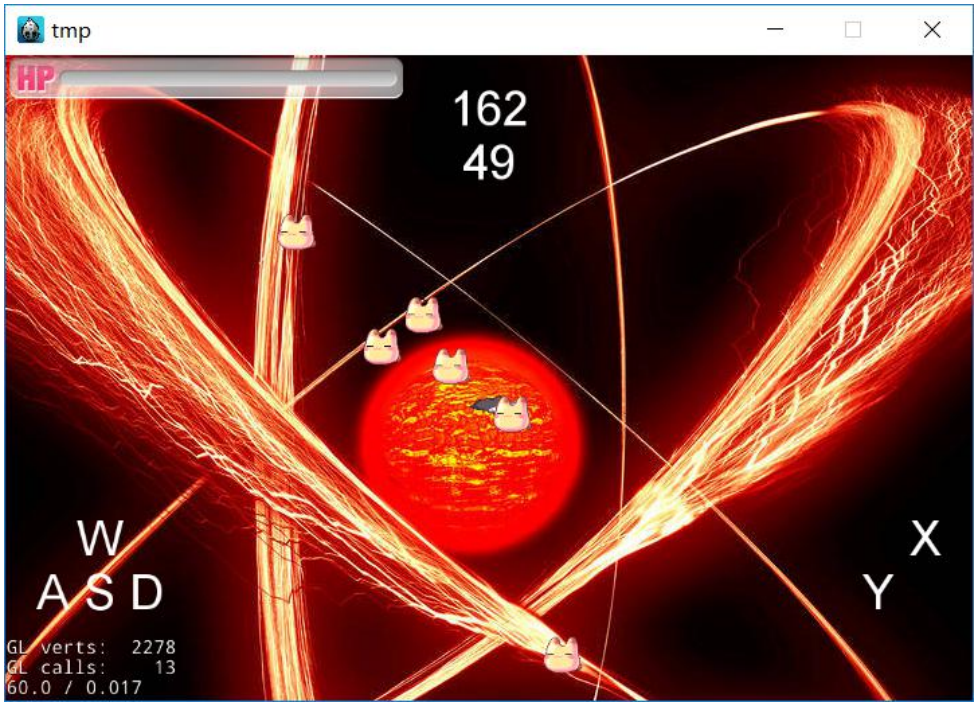
粒子效果：

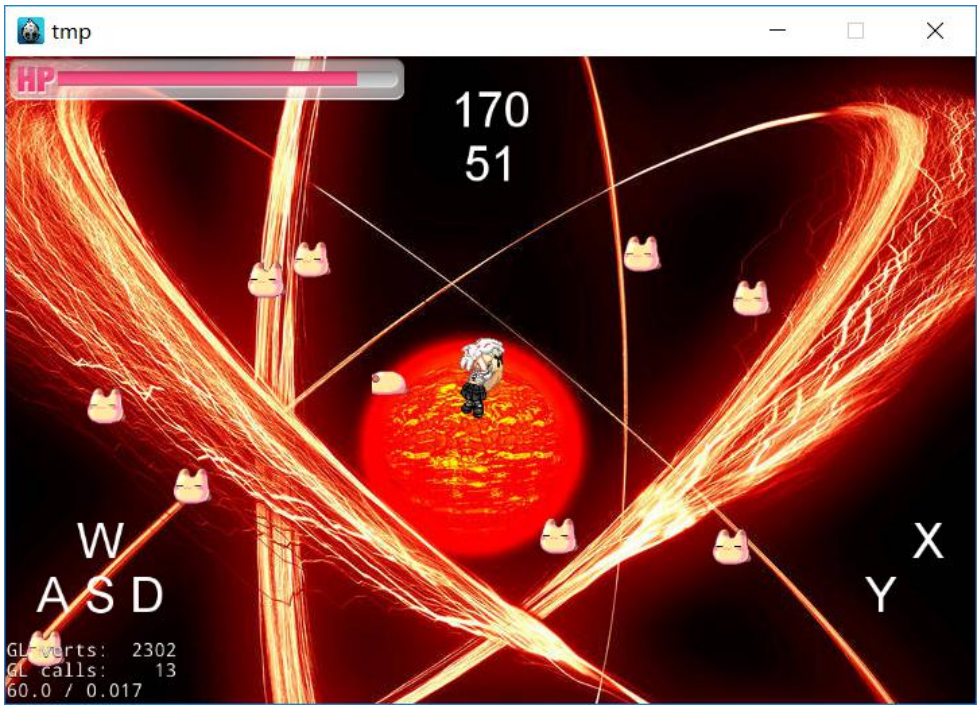
```
ParticleSystem* emitter = ParticleSystemQuad::create("particle_texture.plist");
emitter->setPosition(ball->getContentSize().width / 2, ball->getContentSize().height / 2);
emitter->setTotalParticles(1000);
emitter->setScale(3);
ball->addChild(emitter, 10);


addChild(ball, 3);
```



三、关键步骤截图

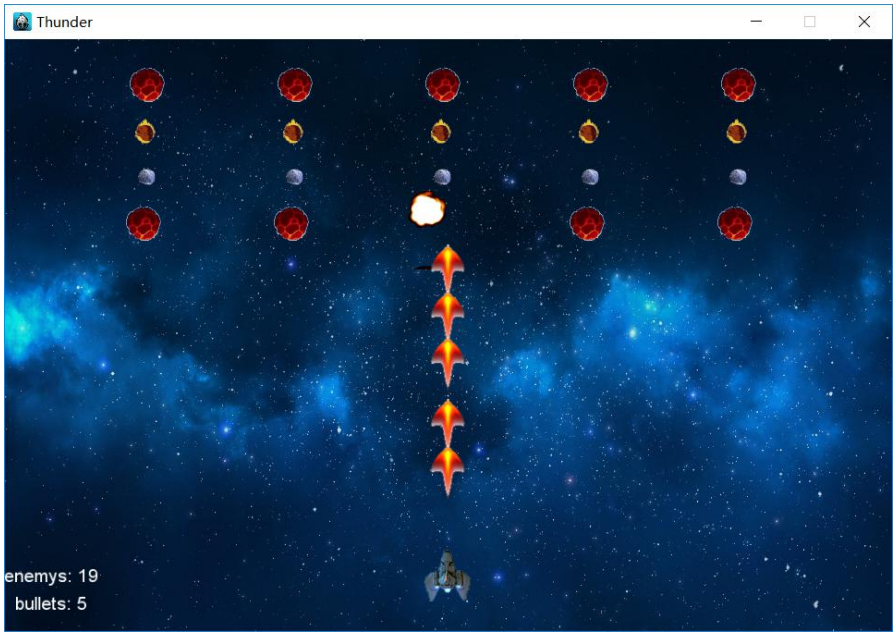
第十三周：
已使用 tilemap，且实现加分项本地存储

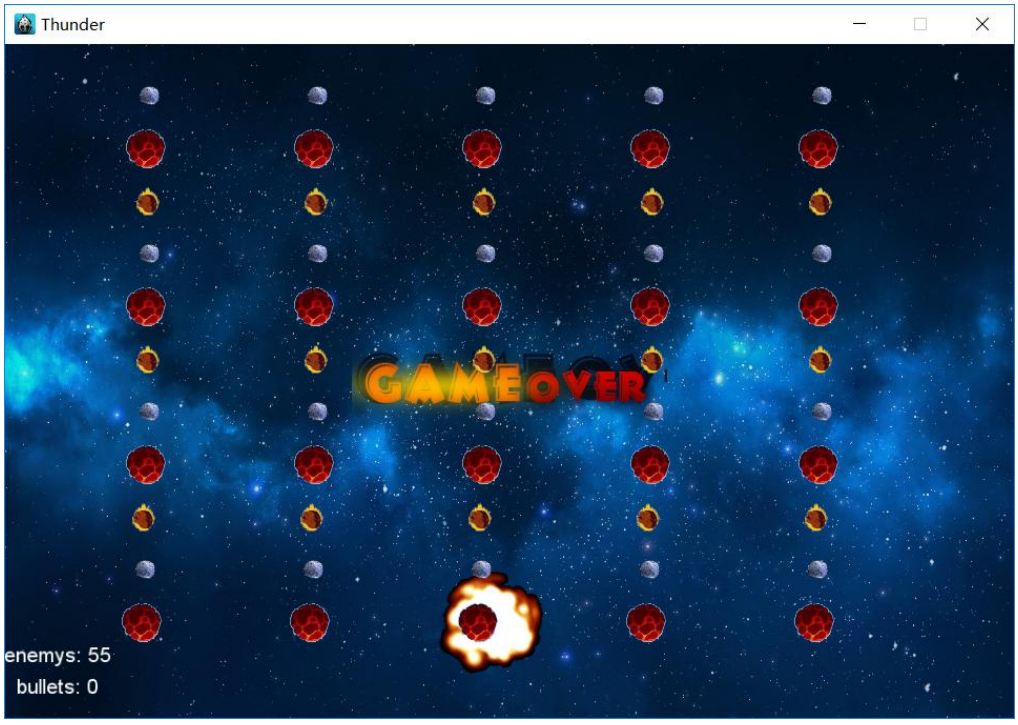

tmp
HP
162
49
W
A S D
X
Y
GL verts: 2278
GL calls: 13
60.0 / 0.017


tmp
HP
170
51
W
A S D
X
Y
GL verts: 2302
GL calls: 13
60.0 / 0.017

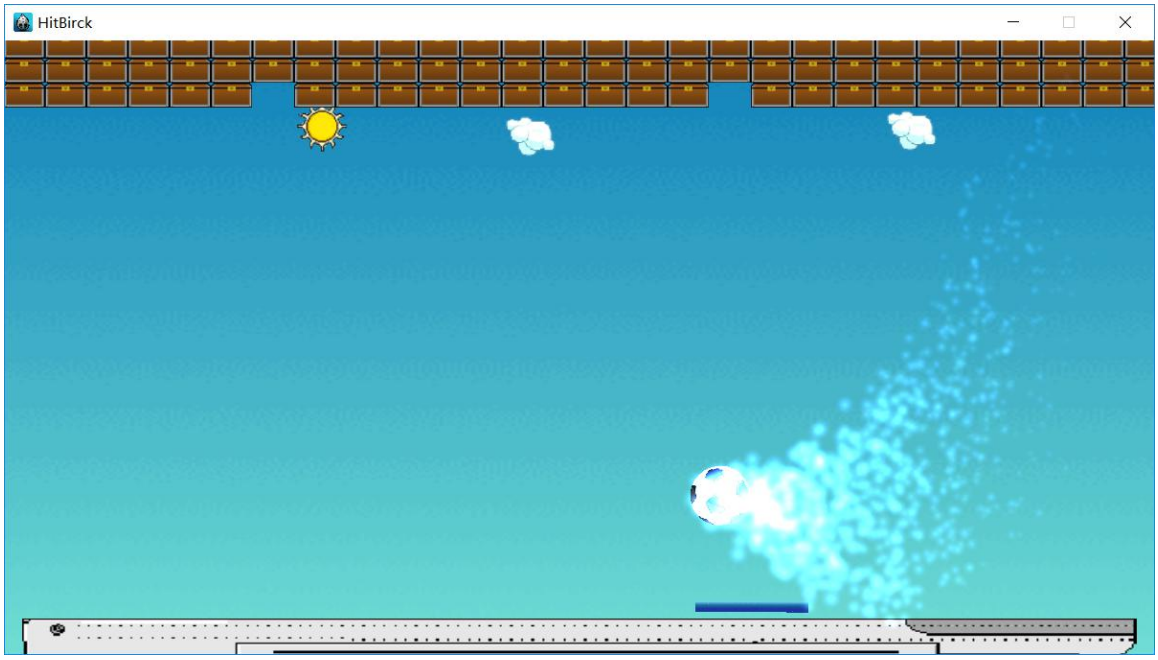
第十四周：

实现加分项陨石的动态增加等





第十五周
实现加分项粒子效果



四、亮点与改进（可选）

实现了每一周的 bonus。
第十三周：
本地存取数据，打倒的怪物数量
第十四周：
陨石的动态增加，鼠标移动
第十五周：
添加了粒子效果

五、遇到的问题

前两周难度较低，没有遇到问题。
十五周作业中遇到了问题。

1. 关节需要添加到物理场景中。忽视这一要求，导致一直无法连接。

```
// 关卡连接，固定球与板子
// Todo
void HitBrick::setJoint() {
    this->joint1= PhysicsJointPin::construct(ball->getPhysicsBody(), player->getPhysicsBody(),Vec2(0,0));
    m_world->addJoint(joint1);
}
```

2. 当球速度过快时，cocos 封装的的物理引擎无法检测到球与板的碰撞，发生穿透。
   通过取消物理场景的默认 step，改用调度器刷新，增加频率，成功解决问题。

```
void HitBrick::MyWorldUpdate(float dt) {
    for (int i = 0; i < 3; ++i)
    {
        m_world->step(1 / 60.0f);
    }
}
```

六、思考与总结

1. 对 cocos 有了更多的了解，了解了更多关于调度器，事件监听，动画效果，粒子效果，物理引擎等相关技术。

2. 初步掌握了 cocos 这个游戏引擎，对如何使用 cocos 完成各种游戏效果有了自己的想法。面对要求不再毫无头绪。

3. Cocos2dx-3.x 版本的教程比较少，官方 API 也不是很详细。给我们 cocos 的学习和使用带来很多麻烦。