现代操作系统应用开发实验报告

学号： 14970011　　　　**班级：** 上午班

姓名： 宋思亭　　　　**实验名称：** homework14

一．参考资料

作业要求文档, 课件 PPT，

官网：http://www.cocos.com/

Github：https://github.com/cocos2d/cocos2d-x

用户手册：http://www.cocos2d-x.org/wiki/Cocos2d-x

商店：http://store.cocos.com/

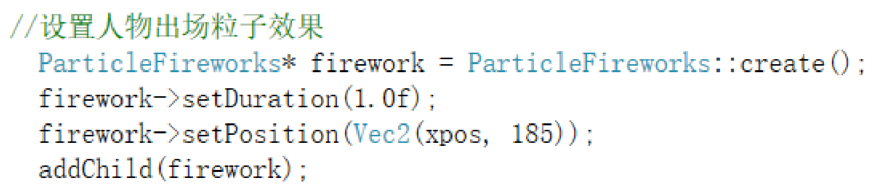
API: http://api.cocos.com/

cocos2d-x 中文显示问题：

http://www.cnblogs.com/cocos2d-x/archive/2012/02/26/2368873.html

二．实验步骤

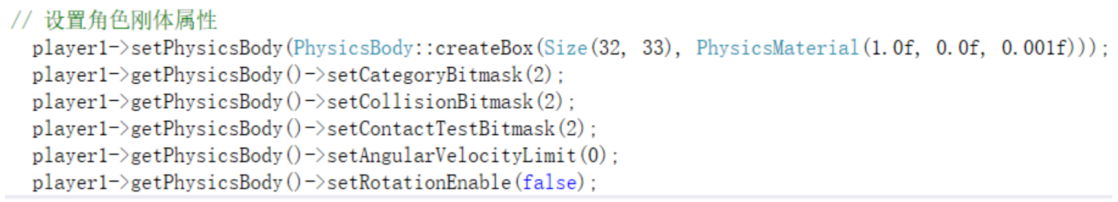
1. 阅读作业需求和课件 PPT，了解阅读课件内容以及作业要求，了解物理引擎与粒子系统的设置。
2. 为人物出场添加火焰粒子效果，持续 1 秒



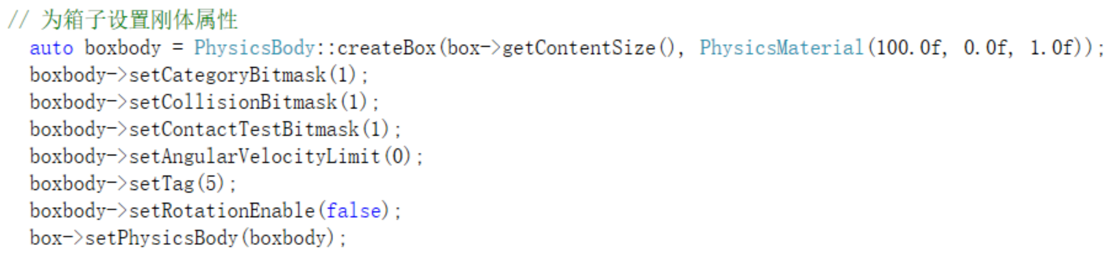
```
//设置人物出场粒子效果
  ParticleFireworks* firework = ParticleFireworks::create();
  firework->setDuration(1.0f);
  firework->setPosition(Vec2(xpos, 185));
  addChild(firework);
```

3. 为玩家和箱子设置刚体属性

设置掩码 1 和 2 位或为 0，此时人物与箱子不会碰撞

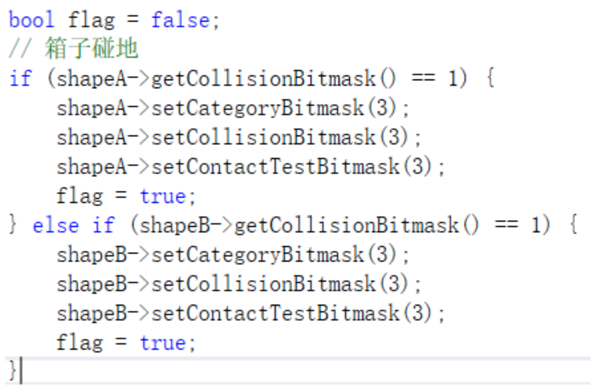


```
// 设置角色刚体属性
  player1->setPhysicsBody(PhysicsBody::createBox(Size(32, 33), PhysicsMaterial(1.0f, 0.0f, 0.001f)));
  player1->getPhysicsBody()->setCategoryBitmask(2);
  player1->getPhysicsBody()->setCollisionBitmask(2);
  player1->getPhysicsBody()->setContactTestBitmask(2);
  player1->getPhysicsBody()->setAngularVelocityLimit(0);
  player1->getPhysicsBody()->setRotationEnable(false);
```



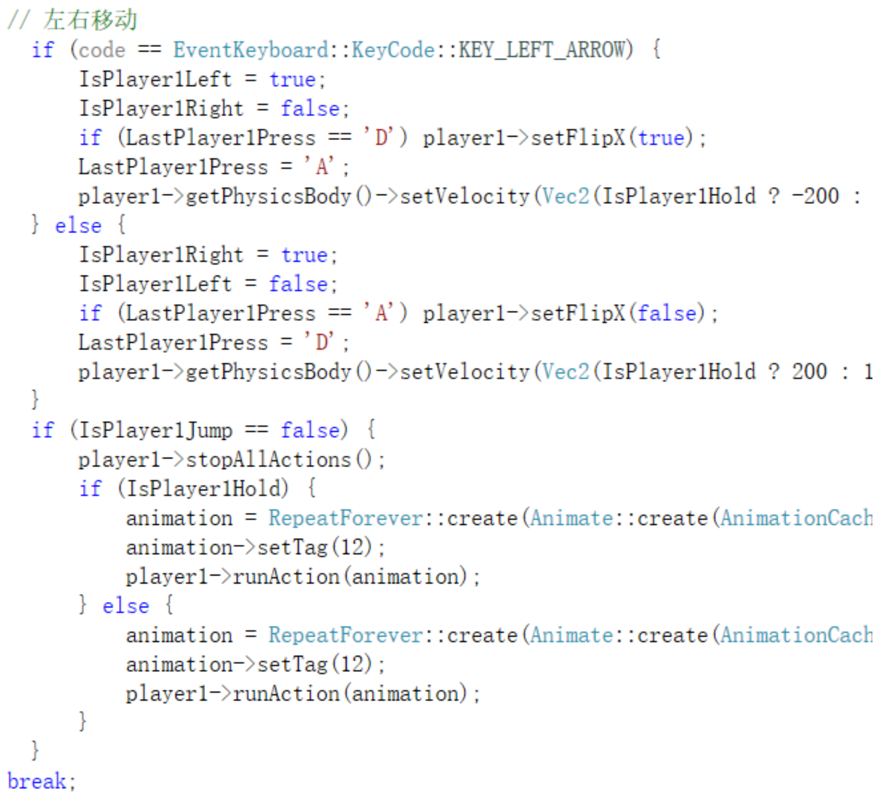
```
// 为箱子设置刚体属性
  auto boxbody = PhysicsBody::createBox(box->getContentSize(), PhysicsMaterial(100.0f, 0.0f, 1.0f));
  boxbody->setCategoryBitmask(1);
  boxbody->setCollisionBitmask(1);
  boxbody->setContactTestBitmask(1);
  boxbody->setAngularVelocityLimit(0);
  boxbody->setTag(5);
  boxbody->setRotationEnable(false);
  box->setPhysicsBody(boxbody);
```

箱子落地后修改掩码为 3，此后人物与箱子产生碰撞



```
bool flag = false;
// 箱子碰地
if (shapeA->getCollisionBitmask() == 1) {
    shapeA->setCategoryBitmask(3);
    shapeA->setCollisionBitmask(3);
    shapeA->setContactTestBitmask(3);
    flag = true;
} else if (shapeB->getCollisionBitmask() == 1) {
    shapeB->setCategoryBitmask(3);
    shapeB->setCollisionBitmask(3);
    shapeB->setContactTestBitmask(3);
    flag = true;
}
```

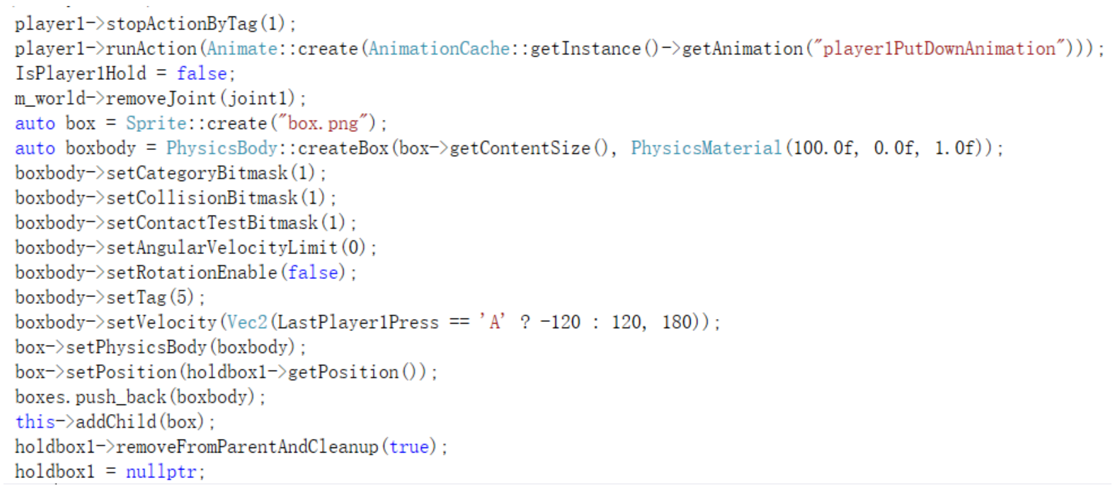
4. 控制玩家左右移动



```
// 左右移动
  if (code == EventKeyboard::KeyCode::KEY_LEFT_ARROW) {
      IsPlayer1Left = true;
      IsPlayer1Right = false;
      if (LastPlayer1Press == 'D') player1->setFlipX(true);
      LastPlayer1Press = 'A';
      player1->getPhysicsBody()->setVelocity(Vec2(IsPlayer1Hold ? -200 :
  } else {
      IsPlayer1Right = true;
      IsPlayer1Left = false;
      if (LastPlayer1Press == 'A') player1->setFlipX(false);
      LastPlayer1Press = 'D';
      player1->getPhysicsBody()->setVelocity(Vec2(IsPlayer1Hold ? 200 : 1
  }
  if (IsPlayer1Jump == false) {
      player1->stopAllActions();
      if (IsPlayer1Hold) {
          animation = RepeatForever::create(Animate::create(AnimationCach
          animation->setTag(12);
          player1->runAction(animation);
      } else {
          animation = RepeatForever::create(Animate::create(AnimationCach
          animation->setTag(12);
          player1->runAction(animation);
      }
  }
break;
```

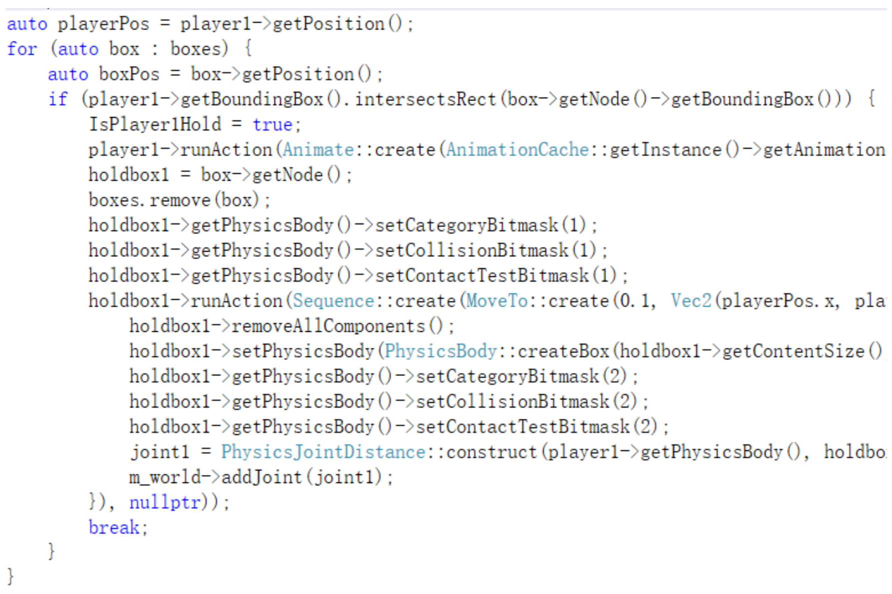
5. 使用固定距离关节举起和扔下箱子

举起箱子



```
player1->stopActionByTag(1);
player1->runAction(Animate::create(AnimationCache::getInstance()->getAnimation("player1PutDownAnimation")));
IsPlayer1Hold = false;
m_world->removeJoint(joint1);
auto box = Sprite::create("box.png");
auto boxbody = PhysicsBody::createBox(box->getContentSize(), PhysicsMaterial(100.0f, 0.0f, 1.0f));
boxbody->setCategoryBitmask(1);
boxbody->setCollisionBitmask(1);
boxbody->setContactTestBitmask(1);
boxbody->setAngularVelocityLimit(0);
boxbody->setRotationEnable(false);
boxbody->setTag(5);
boxbody->setVelocity(Vec2(LastPlayer1Press == 'A' ? -120 : 120, 180));
box->setPhysicsBody(boxbody);
box->setPosition(holdbox1->getPosition());
boxes.push_back(boxbody);
this->addChild(box);
holdbox1->removeFromParentAndCleanup(true);
holdbox1 = nullptr;
```

扔下箱子



```
auto playerPos = player1->getPosition();
for (auto box : boxes) {
    auto boxPos = box->getPosition();
    if (player1->getBoundingBox().intersectsRect(box->getNode()->getBoundingBox())) {
        IsPlayer1Hold = true;
        player1->runAction(Animate::create(AnimationCache::getInstance()->getAnimation
        holdbox1 = box->getNode();
        boxes.remove(box);
        holdbox1->getPhysicsBody()->setCategoryBitmask(1);
        holdbox1->getPhysicsBody()->setCollisionBitmask(1);
        holdbox1->getPhysicsBody()->setContactTestBitmask(1);
        holdbox1->runAction(Sequence::create(MoveTo::create(0.1, Vec2(playerPos.x, pla
            holdbox1->removeAllComponents();
            holdbox1->setPhysicsBody(PhysicsBody::createBox(holdbox1->getContentSize()
            holdbox1->getPhysicsBody()->setCategoryBitmask(2);
            holdbox1->getPhysicsBody()->setCollisionBitmask(2);
            holdbox1->getPhysicsBody()->setContactTestBitmask(2);
            joint1 = PhysicsJointDistance::construct(player1->getPhysicsBody(), holdbo
            m_world->addJoint(joint1);
        }), nullptr));
        break;
    }
}
```

6. 实现人物跳跃

```
// 跳
  if (IsPlayer1Jump) break;
  player1->stopAllActions();
  player1->getPhysicsBody()->setVelocity(Vec2(0, 250));
  if (holdbox1) holdbox1->getPhysicsBody()->setVelocity(player1->getPhysicsBody()->getVelocity());
  IsPlayer1Jump = true;
  SimpleAudioEngine::getInstance()->playEffect("jump.mp3", false);
  if (IsPlayer1Hold) {
      auto animation = Animate::create(AnimationCache::getInstance()->getAnimation("player1JumpWit
      animation->setTag(11);
      player1->runAction(animation);
  } else {
      auto animation = Animate::create(AnimationCache::getInstance()->getAnimation("player1JumpWit
      animation->setTag(11);
      player1->runAction(animation);
  }
break;
```

7. 轮船倾斜以及翻船

将轮船锚点设置在中心 0.5，0 处，每次箱子落下时计算左右箱子质量和距离的乘积，并使用 RotateBy 更新旋转角度。

```
double leftWeight = 0, rightWeight = 0;
for (auto box : boxes) {
    if ((box->getPosition()).x < visibleSize.width / 2)
        leftWeight += box->getTag() * (visibleSize.width / 2 - (box->getPosition()).x);
    else
        rightWeight += box->getTag() * ((box->getPosition()).x - visibleSize.width / 2);
}
height = (rightWeight - leftWeight) / 2000;
```

```
// 更新船的平衡和倾斜(加分项)
void FriendShip::updateShip(float dt) {
    if (height != newHeight) {
        ship->runAction(RotateBy::create(0.5f, height - newHeight));
        newHeight = height;
    }
}
```

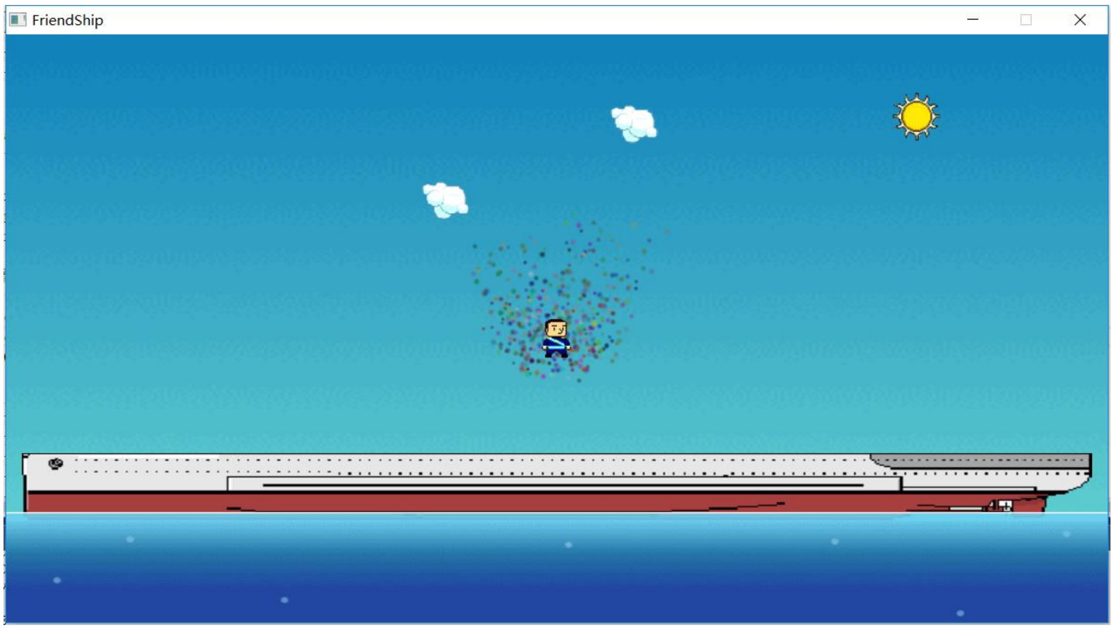
当左右箱子质量之差大于一定值时，翻船并游戏结束

```
if (fabs(height) > 10) {
    if (height > 0) ship->runAction(RotateTo::create(0.5f, 40));
    else ship->runAction(RotateTo::create(0.5f, -40));
    GameOver();
    return true;
}
```

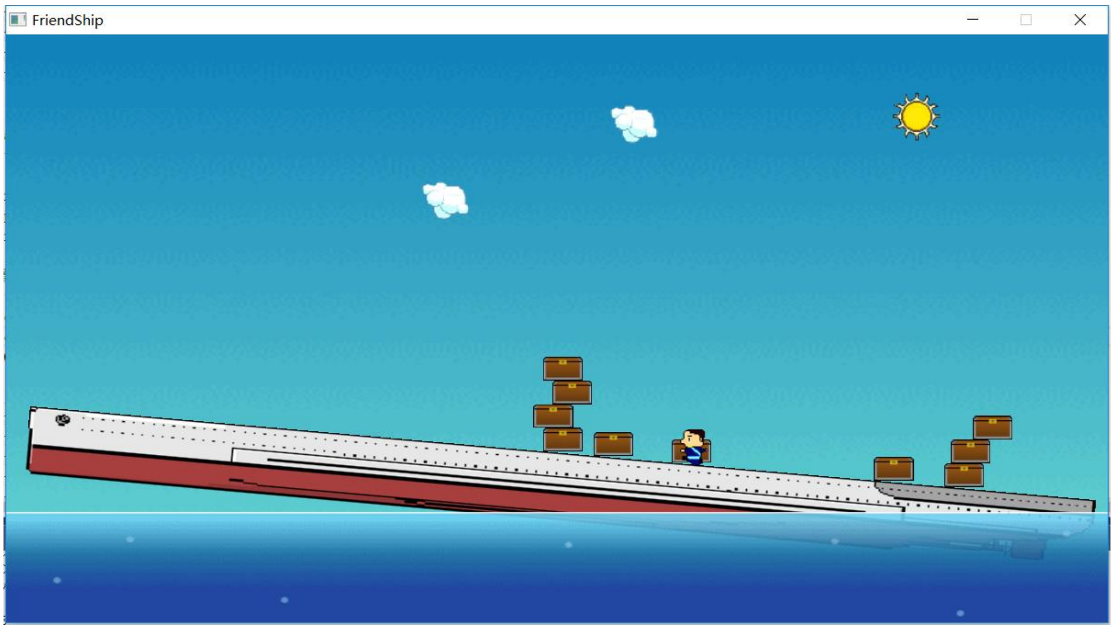
8. 调试项目

三．实验结果截图

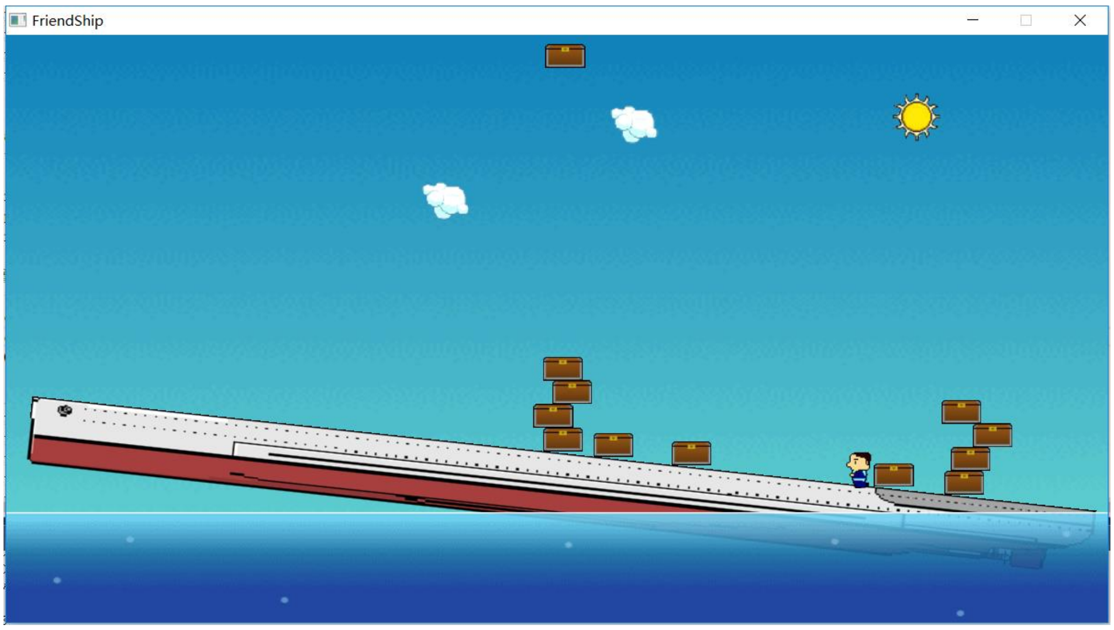
1. 打开窗口



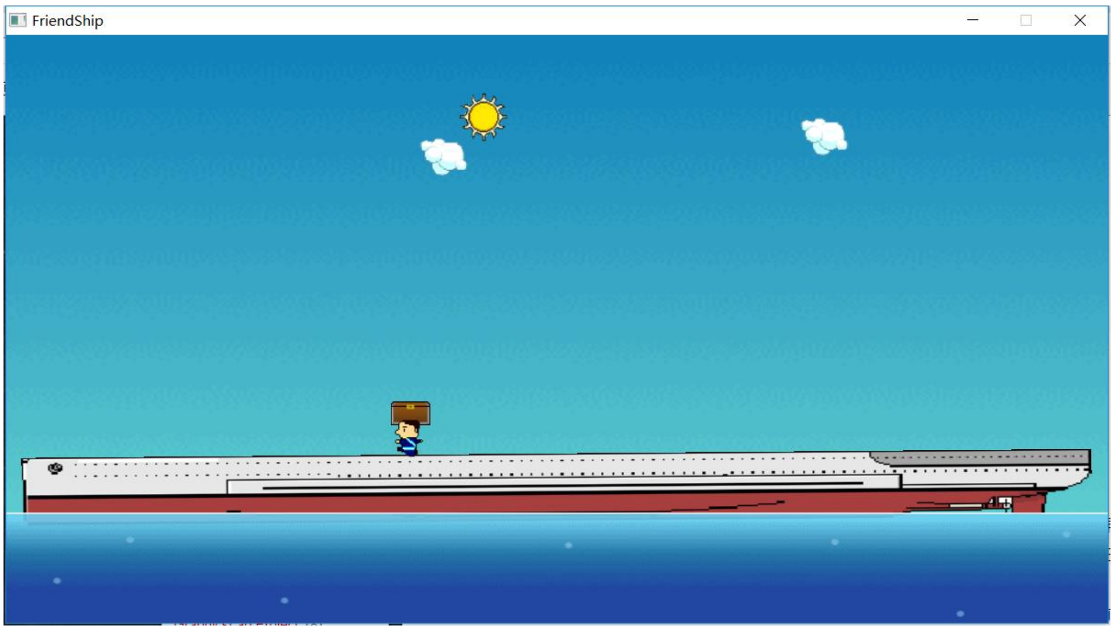
2.　箱子落下与人物不碰撞

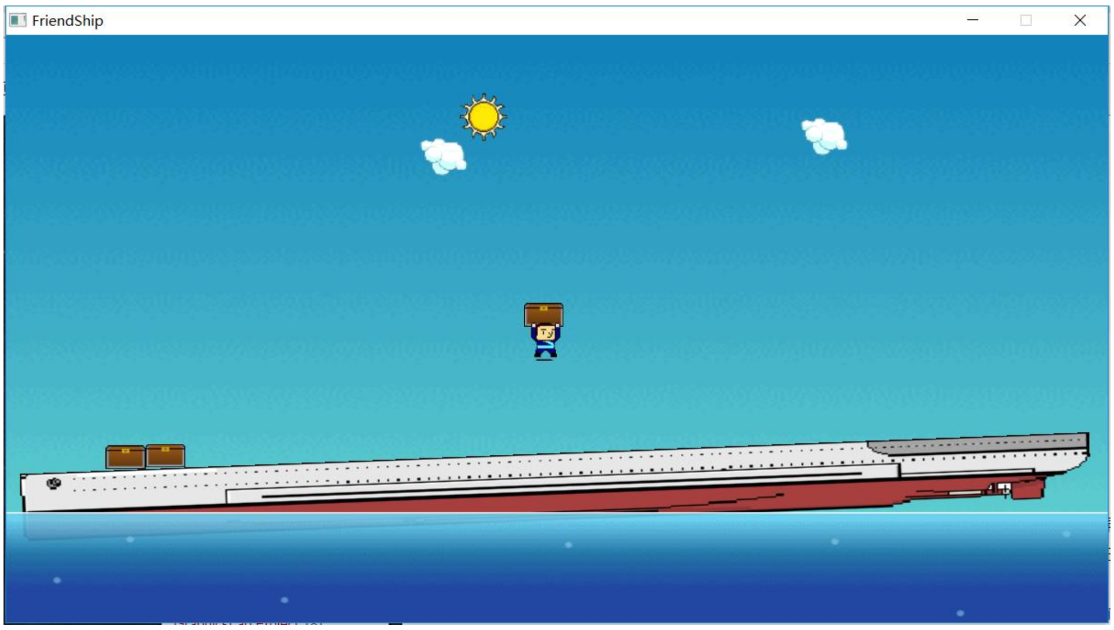


3.　箱子落下后与人物碰撞

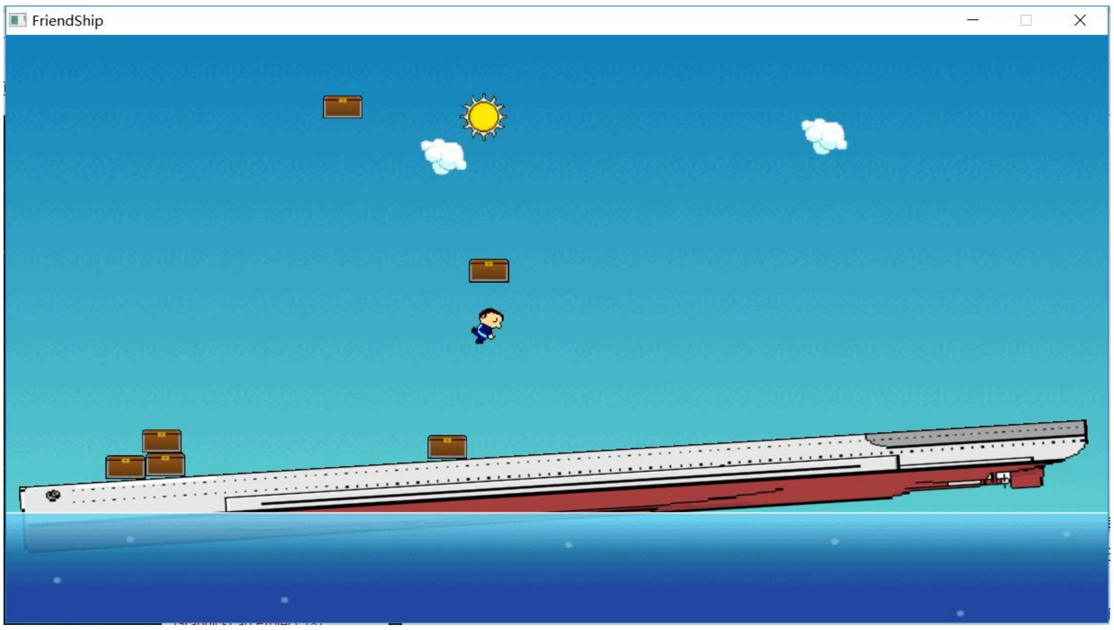


4.　举起箱子跑跳

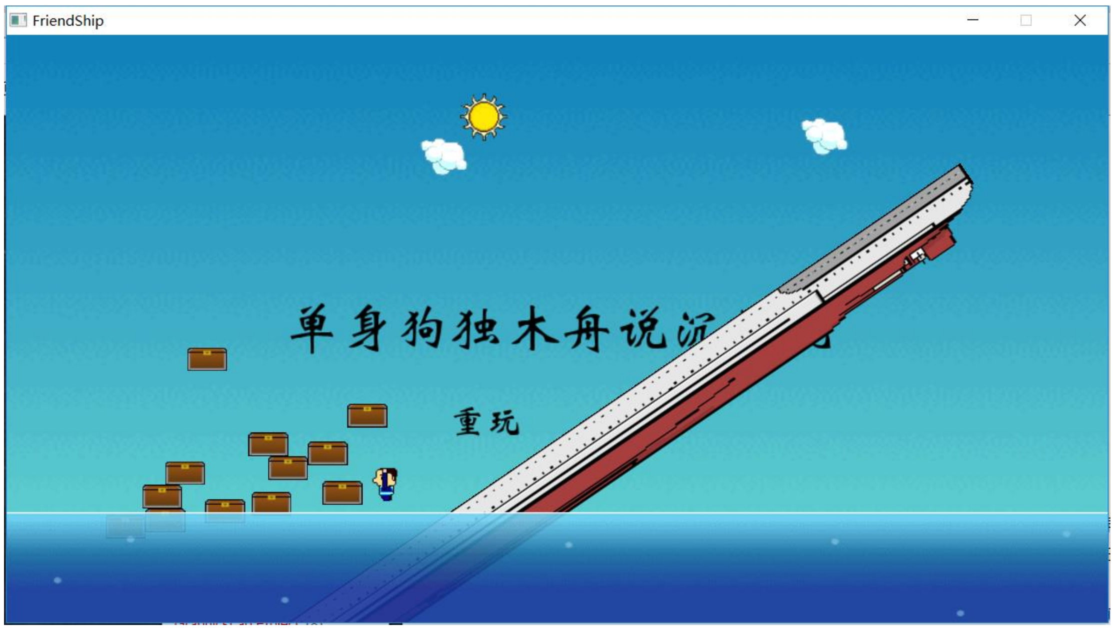




5.　扔出箱子



6.　游戏结束，翻船



四．实验过程遇到的问题

1. `auto label3 = Label::createWithTTF("退出", "fonts/STXINWEI.TTF", 40);`语句报错。

   修改编码方式为 UTF-8 带签名版解决。

2. 为了相对好的游戏体验，调整船的平衡函数，height 值等花了一些时间

3. 船倾斜一定角度后，存在人物或箱子掉到水里的 bug，暂时没有改好

4. 感谢 ta 帮我们排好了所有坑（逃

五．思考与总结

1. 真看似简单的一个小游戏，实现起来还是遇到很多问题，需要通过网络寻找答案，经过这次作业，对物理引擎与粒子系统的设置有了更深的了解。

2. 把程序分解成一个个小的部分，分而治之，更有效率而且更容易排错。

3. 学习理论之后直接去实践很有用，虽然途中遇到了很多 bug 甚至是很难找出的错误的断点。但是只要肯花时间，一定会解决眼前的困难的。