现代操作系统应用开发实验报告

姓名：_____王晶_____

学号：___16340217____

实验名称：cocos2dx 第 1-3 次作业

一、参考资料

https://www.jianshu.com/p/bc80a373242c

https://blog.csdn.net/ac_huang/article/details/28463357

https://blog.csdn.net/chenqiai0/article/details/46820979

https://blog.csdn.net/yongg3224/article/details/47042097

二、实验步骤

第一次作业：创建一个项目，显示自己的学号和姓名，更换显示的图片。

首先是创建一个新的项目 ,主要的框架已经给出 ,那么只要改一下即可。首先要添加 labal，但是直接添加中文会无法显示，这时需要通过词典类，将 xml 文件加载进来，再通过 key 获取 value。然后创建 label，通过 setColor 改变颜色。然后通过图片创建精灵。除此之外创建 menuitem，通过回调函数来实现动作，并在控制台输出。

第二次作业：完成 demo

首先完善主界面，先添加开始按钮，还有金块。然后通过回调函数进行场景切换。然后是游戏界面。Demo 中已经有了触屏响应的框架，只要增加内容即可。获取 touch 的位置，然后创建精灵 cheese，添加在该位置上即可，然后让老鼠移动到此处。点击 shoot，射出石头，让老鼠留下 diamond，然后移动。仿照 leg 的动画，读入 level-sheet.plist 文件，加载老鼠的帧动画。

第三次作业：完成 demo，实现 wasdxy 的功能。

首先完成倒计时功能，然后完成按 X 减少血条，按 Y 增加血条的功能。完成点击 WASD 使人移动的功能。然后需要增加一个变量，判断当前是否有动作或动画在发生，来防止同时进行不同的动作和动画。限制人物无法移动到窗口外，并且添加动画。

三、关键步骤截图

第一次作业：

```
//创建一个精灵
auto sprite = Sprite::create("myPic.jpg");
//创建词典类实例，将xml文件加载到词典中
auto *chnStrings = Dictionary::createWithContentsOfFile("myName.xml");
//通过xml文件中的key获取value
const char *str1 = ((String*)chnStrings->objectForKey("name"))->getCString();
const char *str2 = ((String*)chnStrings->objectForKey("num"))->getCString();

//创建一个label
auto label = Label::create(str1, "Arial", 24);
auto label1 = Label::create(str2, "Arial", 24);
//修改颜色
label1->setColor(Color3B(249, 110, 123));
```


```
now click!
now click!
now click!
now click!
```

All Output Filter

第二次作业：

场景跳转

```
void MenuSence::menuChangeCallback(Ref* pSender)
{
    CCDirector::sharedDirector()->replaceScene(CCTransitionFade::create(0.5, GameSence::createScene()));
```

创建两个 layer，并设置锚点

```
GameSence::stoneLayer = LayerColor::create();
stoneLayer->setPosition(Point(0, 0));

GameSence::mouseLayer = LayerColor::create();
mouseLayer->setPosition(Point(0, visibleSize.height / 2));
```

触摸响应函数

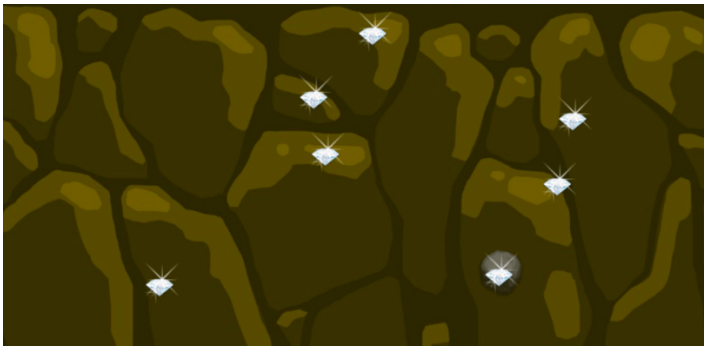
```
bool GameSence::onTouchBegan(Touch *touch, Event *unused_event) {
    Size visibleSize = Director::getInstance()->getVisibleSize();
    auto location = touch->getLocation();

    auto cheese = Sprite::create("cheese.png");
    cheese->setPosition(location);
    this->addChild(cheese, 1);
    cheese->runAction(FadeOut::create(4));

    auto moveTo = MoveTo::create(1, Point(location.x, location.y - visibleSize.height / 2));
    auto actions = Sequence::create(moveTo, NULL);
    mouse->runAction(actions);

    return true;
}
```

Shoot 点回调击函数

```
void GameSence::menu(Ref* pSender)
{
    Size visibleSize = Director::getInstance()->getVisibleSize();
    auto stone_ = Sprite::create("stone.png");
    stone_->setPosition(Point(560, 480));
    stoneLayer->addChild(stone_, 1);
    stone_->runAction(FadeOut::create(3));

    auto moveTo = MoveTo::create(1, Point(mouse->getPosition().x, mouse->getPosition().y + visibleSize.height / 2));
    auto actions = Sequence::create(moveTo, NULL);
    stone_->runAction(actions);

    auto diamond = Sprite::create("diamond.png");
    diamond->setPosition(Point(mouse->getPosition().x, mouse->getPosition().y + visibleSize.height / 2));
    stoneLayer->addChild(diamond, 1);

    auto moveTo1 = MoveTo::create(1, Point(rand()%900, rand()%600 - 300));
    auto actions1 = Sequence::create(moveTo1, NULL);
    mouse->runAction(actions1);
```

实际效果


Shoo




Shoot

第三次作业：

时间

```
time = Label::createWithTTF(ttfConfig, "180");
//倒计时周期性调用调度器
schedule(schedule_selector(HelloWorld::updateTime), 1.0f, kRepeatForever, 0);
//倒计时数字
dtime = 180;
time->setPosition(Vec2(origin.x + visibleSize.width / 2,
                       origin.y + visibleSize.height - time->getContentSize().height));
```

X 和 Y 的动画以及对血条的改变：

```
switch (cid) {
    case 'X':
        if (this->en) {
            this->player->runAction(Sequence::create(Dead, callbackMove, NULL));
            this->en = false;
            i = pT->getPercentage();
            pT->setPercentage(i - 20);
        }
        break;
    case 'Y':
        auto animation = Animation::createWithSpriteFrames(attack, 0.1f);
        animation->setRestoreOriginalFrame(true);
        myattack = Animate::create(animation);
        if (this->en) {
            this->player->runAction(Sequence::create(this->myattack,callbackMove, NULL));
            this->en = false;
            i = pT->getPercentage();
            pT->setPercentage(i + 20);
        }
        break;
}
```

WASD 的移动功能，每次移动 50：

```
auto action_W = MoveBy::create(0.5, Point(0, 50));
auto action_A = MoveBy::create(0.5, Point(-50, 0));
auto action_S = MoveBy::create(0.5, Point(0, -50));
auto action_D = MoveBy::create(0.5, Point(50, 0));

switch (cid) {
    case 'W':
        if (judge_move(Point(0, 40)) && en) {
            auto seq = Sequence::create(action_W, callbackMove, NULL);
            player->runAction(seq);
            player->runAction(run);
            en = false;
        }
        break;
    case 'A':
        if (judge_move(Point(-50, 0)) && en) {
            auto seq = Sequence::create(action_A, callbackMove, NULL);
            player->runAction(seq);
            player->runAction(run);
            en = false;
        }
        break;
    case 'S':
        if (judge_move(Point(0, -50)) && en) {
            auto seq = Sequence::create(action_S, callbackMove, NULL);
            player->runAction(seq);
            player->runAction(run);
            en = false;
        }
        break;
    case 'D':
        if (judge_move(Point(50, 0)) && en) {
            auto seq = Sequence::create(action_D, callbackMove, NULL);
            player->runAction(seq);
            player->runAction(run);
            en = false;
        }
        break;
```

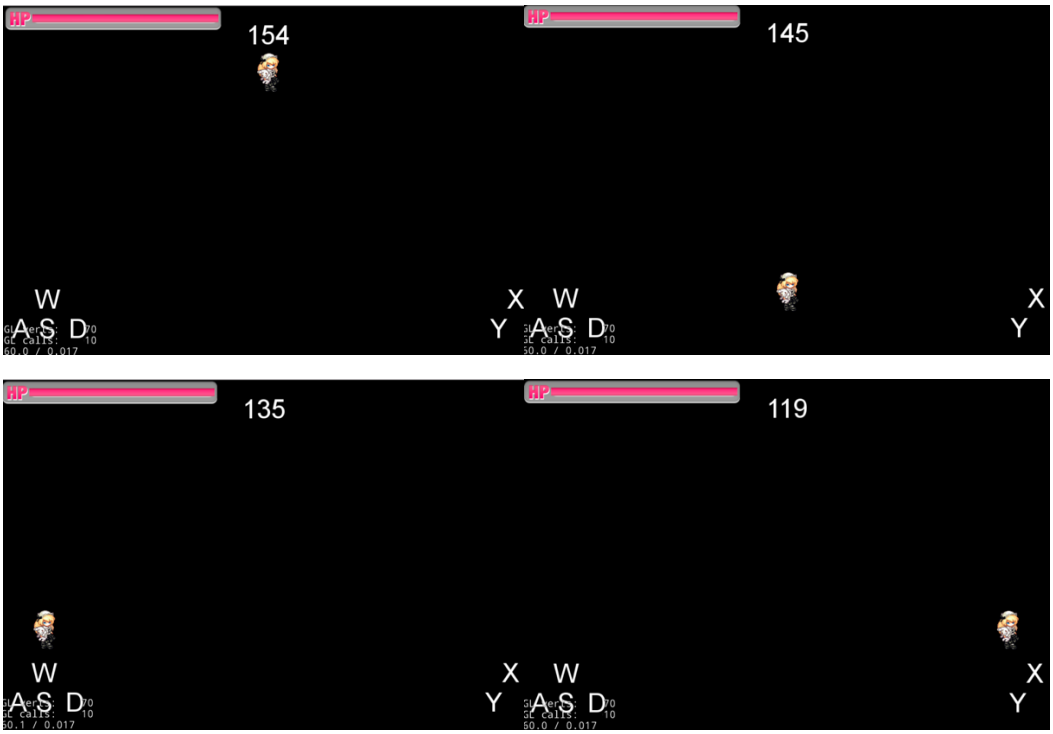
bool 变量 en 防止同时进行多个动作：

```
bool en;

auto callbackMove = CallFunc::create([&] {
    this->en = true;
});
```

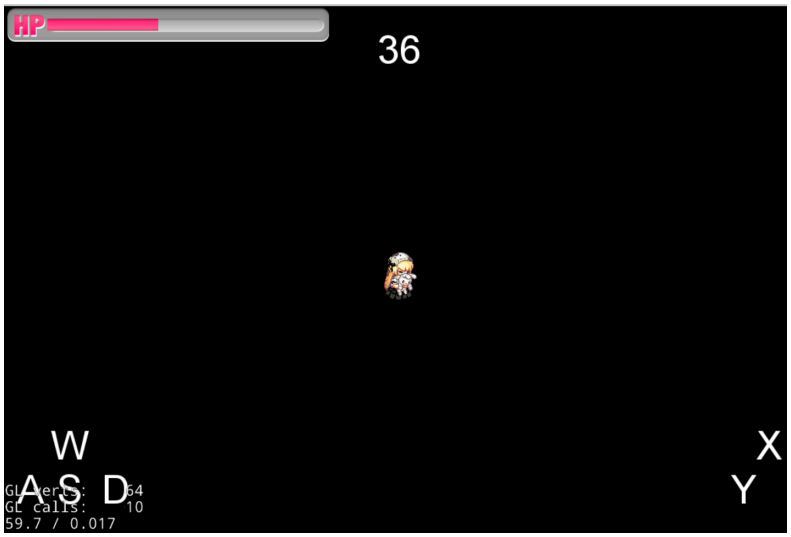
判断是否会走出窗口：

```
bool HelloWorld::judge_move(Point b) {
    Point p = player->getPosition() + b;
    if (p.y >= this->visibleSize.height || p.x >= this->visibleSize.width || p.y <= 0 || p.x <= this->origin.x) {
        return false;
    }
    return true;
}
```

WASD 移动（同时有动画）：



X 和 Y 的动作以及血条效果（有各自对应的动画）：



四、亮点与改进（可选）

第一次作业：增加了点击事件，点击 click 按钮，会在控制台输出“now click”。

```
auto menuItem_1 = MenuItemFont::create("Click", CC_CALLBACK_1(HelloWorld::myClick, this));
menuItem_1->setPosition(origin.x + visibleSize.width/3,
                        origin.y + visibleSize.height/2);
menu->addChild(menuItem_1, 2);
```

```
void HelloWorld::myClick(Ref* pSender)
{
    cout << "now click!\n" << endl;
}
```

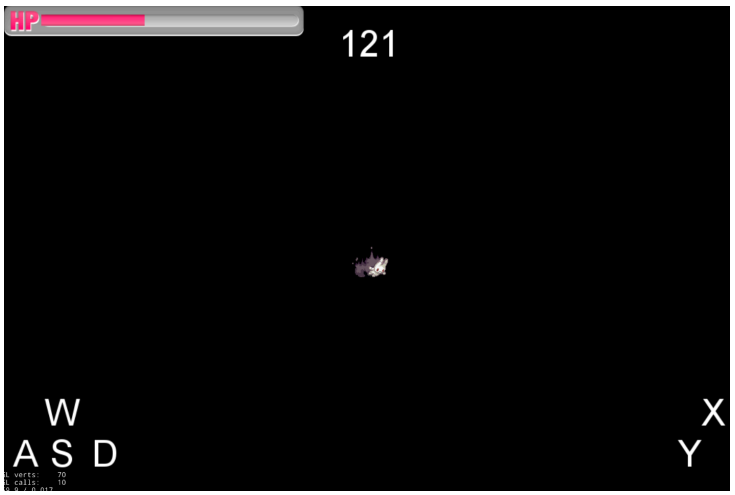
```
now click!
now click!
now click!
now click!
All Output
Filter
```

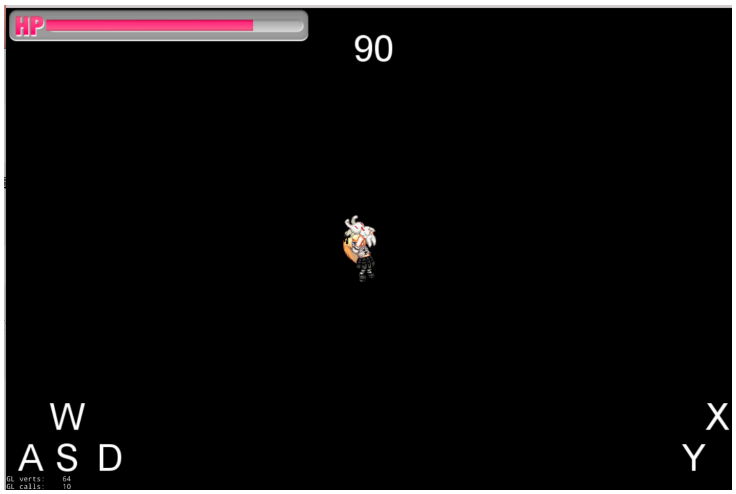
第二次作业：增加了老鼠的帧动画，在游戏中腿会有摆动的效果。

```
//老鼠动画
SpriteFrameCache::getInstance()->addSpriteFramesWithFile("level-sheet.plist");
char totalFrames1 = 8;
char frameName1[30];
Animation* mouseAnimation = Animation::create();

for (int i = 0; i < totalFrames1; i++)
{
    sprintf(frameName1, "gem-mouse-%d.png", i);
    mouseAnimation->addSpriteFrame(SpriteFrameCache::getInstance()->getSpriteFrameByName(frameName1));
}
mouseAnimation->setDelayPerUnit(0.1);
AnimationCache::getInstance()->addAnimation(mouseAnimation, "mouseAnimation");
```

第三次作业：

X 和 Y 动作时可以分别让血条减少和增加





实现原理如下：

```
i = pT->getPercentage();
pT->setPercentage(i - 20);
```

五、遇到的问题

第一次作业中，倒是没什么问题，因为内容比较简单。在第二次作业中，不太理解动画加载的机制，以及实现。在浏览了博客内容之后，我在 Xcode 中打开了 level-sheet.plist 文件，这才发现已经分隔好，并且图片已经合成在对应的 png 文件里了。

而第三次作业中，一开始没有考虑到如何防止同时发生动作，于是偶然同时点击了 X 和 Y，发现人物闪动，出现问题。一开始想查找是否有什么相关函数可以使用，后来想想直接设置一个变量就能够解决了。同理还有越界的问题，每次移动前进行判断就可以了。

六、思考与总结

这是第一次接触制作游戏，而 2D 游戏的复杂度相对于 3D 游戏可能会较低一些。我想每个接触过电脑游戏的人都会期待着能够做出一款属于自己的游戏。这几次虽然是在一个给定的框架下进行完善，但也学到了很多东西，例如触摸事件监听和调用，还有动画的加载和使用，以及第一次接触 plist 这种形式的内容。也算是解开了以前遇到的一些疑惑（在游戏的文件中有一些多个元素组合成的图片）。除此之外，也学会了调度器的使用，完成了倒计时的功能。还发现了一个简单的变量能够让整个游戏运行地更加地合理和正确，例如防止叠加动画和防止走出窗口的变量。这是在一步步地的学习和练习中学到的东西，整个 cocos2dx 还有很多等待挖掘的内容。不过总的来说，学到了很多新的没有接触过的内容，但还有更多的疑惑等待着去解决。