现代操作系统应用开发实验报告

姓名：余漫霖

学号：16340280

实验名称：cocos2d-x开发入门

一、参考资料

cocos2d-js动画结束监听

CC_CALLBACK之间的区别

实例介绍Cocos2d-x精灵菜单和图片菜单

【cocos2d-x游戏开发】Label标签的使用

二、实验步骤

第一周

- 添加自己的姓名、学号

```
    //创建词典类实例，将xml文件加载到词典中
    auto *chnStrings = Dictionary::createWithContentsOfFile("cnLabel.xml");
    //通过xml文件中的key获取value
    //添加姓名
    const char *name = (chnStrings->valueForKey("name"))->getCString();
    auto label = Label::createWithSystemFont(name, "Microsoft Yahei", 30);

    label->enableOutline(Color4B::RED, 2);

    if (label == nullptr)
    {
        problemLoading("'fonts/Marker Felt.ttf'");
    }
    else
    {
        // position the label on the center of the screen
        label->setPosition(Vec2(origin.x + visibleSize.width / 2 - 60,
             origin.y + visibleSize.height - label->getContentSize().height));

        // add the label as a child to this layer
        this->addChild(label, 1);
    }

    //添加学号
    const char* sid = (chnStrings->valueForKey("sid"))->getCString();
    auto sidLabel = Label::createWithTTF(sid, "fonts/Marker Felt.ttf", 30);
    sidLabel->setTextColor(Color4B::YELLOW);
    sidLabel->enableOutline(Color4B::RED, 2);

    if (sidLabel == nullptr)
    {
        problemLoading("'fonts/Marker Felt.ttf'");
    }
    else
    {
        // position the label on the center of the screen
        sidLabel->setPosition(Vec2(origin.x + visibleSize.width / 2 + 60,
             origin.y + visibleSize.height - label->getContentSize().height));

        // add the label as a child to this layer
        this->addChild(sidLabel);
    }
```

- 更换图片

```
    //更换图片
    auto sprite = Sprite::create("haikyuu.jpg");
    if (sprite == nullptr)
    {
        problemLoading("'haikyuu.jpg'");
    }
    else
    {
        // position the sprite on the center of the screen
        sprite->setPosition(Vec2(visibleSize.width/2 + origin.x, visibleSize.height/2 +
origin.y));

        // add the sprite as a child to this layer
        this->addChild(sprite, 0);
    }
```

- 设置文字样式

```
    //添加学号
    const char* sid = (chnStrings->valueForKey("sid"))->getCString();
    auto sidLabel = Label::createWithTTF(sid, "fonts/Marker Felt.ttf", 30);
    sidLabel->setTextColor(Color4B::YELLOW);
    sidLabel->enableOutline(Color4B::RED, 2);
```

- 添加一个MenuItem

```
//设置自定义的图片菜单
MenuItemImage *settingMenuItem = MenuItemImage::create(
  "button.png",
  "button.png",
  CC_CALLBACK_1(HelloWorld::menuItemSettingCallback, this));
settingMenuItem->setPosition(Vec2(visibleSize.width * 0.7f + origin.x, visibleSize.height * 0.2f
+ origin.y));
```

- 该MenuItem有简单的触发事件

```
//自定义的图片菜单的回调函数，点击该菜单项会出现一个精灵
void HelloWorld::menuItemSettingCallback(Ref* pSender)
{
    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    auto sprite = Sprite::create("HelloWorld.png");
    if (sprite == nullptr)
    {
        problemLoading("'HelloWorld.png'");
    }
    else
    {
        sprite->setPosition(Vec2(visibleSize.width * 0.8f + origin.x, visibleSize.height * 0.8f
+ origin.y));

        this->addChild(sprite);
    }
}
```

第二周

菜单界面

- 添加开始按钮

```
    //按钮
    auto title = Sprite::create("gold-miner-text.png");
    title->setPosition(origin.x + visibleSize.width / 2, visibleSize.height / 2 + origin.y +
180);
    this->addChild(title, 1);

    //作为按钮背景的金子
    auto gold = Sprite::create("menu-start-gold.png");
    gold ->setPosition(origin.x + visibleSize.width / 2 + 200, visibleSize.height / 2 + origin.y
- 150);
    this->addChild(gold, 1);

    auto startMenu = MenuItemImage::create(
        "start-0.png",
        "start-1.png",
        CC_CALLBACK_1(MenuScene::startMenuCallback, this));
    auto menu = Menu::create(startMenu, NULL);
    menu->setPosition(origin.x + visibleSize.width / 2 + 200, visibleSize.height / 2 + origin.y
- 100);
    this->addChild(menu, 1);
```

- 为开始按钮添加事件——切换场景

```
void MenuScene::startMenuCallback(Ref* pSender) {
    auto scene = GameScene::create();
    Director::getInstance()->replaceScene(TransitionPageTurn::create(0.5f, scene, true));
}
```

- 添加人物动画——吹口哨

```
    //吹口哨动画
    auto whistle = Sprite::createWithSpriteFrameName("miner-face-whistle-0.png");
    Animate* whistleAnimate = Animate::create(AnimationCache::getInstance()-
>getAnimation("whistleAnimation"));
    whistle->runAction(RepeatForever::create(whistleAnimate));
    whistle->setPosition(163 + origin.x, 364 + origin.y);
```

游戏界面

- 添加layer

```
    //游戏石头层
    stoneLayer = Layer::create();
    this->addChild(stoneLayer, 1);
    //游戏老鼠层
    mouseLayer = Layer::create();
    mouseLayer->setPosition(0, visibleSize.height / 2);
    this->addChild(mouseLayer, 2);
```

- 添加精灵、菜单项

```
    //石头精灵
    stone = Sprite::create("stone.png");
    stoneLayer->addChild(stone);
    stone->setPosition(560, 480);

    //老鼠精灵
    mouse = Sprite::createWithSpriteFrameName("gem-mouse-0.png");
    mouseLayer->addChild(mouse);
    mouse->setPosition(visibleSize.width / 2, 0);
    //使用老鼠的动画资源
    Animate* mouseAnimate = Animate::create(AnimationCache::getInstance()-
>getAnimation("mouseAnimation"));
    mouse->runAction(RepeatForever::create(mouseAnimate));

    //shoot菜单项
    MenuItemFont::setFontName("Times New Roman");
    MenuItemFont::setFontSize(86);
    MenuItemFont *shootMenu = MenuItemFont::create("Shoot",
            CC_CALLBACK_1(GameScene::shootMenuCallback, this));

    //菜单
    auto menu = Menu::create(shootMenu, NULL);
    menu->setPosition(visibleSize.width / 2 + origin.x + 300, visibleSize.height / 2 + origin.y
+ 170);
    this->addChild(menu, 3);
```

- 点击屏幕任意位置，在该位置添加一块奶酪，老鼠跑到该位置吃掉奶酪。

```
//触摸事件
bool GameScene::onTouchBegan(Touch *touch, Event *unused_event) {

    auto location = touch->getLocation();
    auto cheese = Sprite::create("cheese.png");
    cheese->setPosition(mouseLayer->convertToNodeSpace(location));
    mouseLayer->addChild(cheese);

    //老鼠动作：移动到芝士位置
    mouse->runAction(MoveTo::create(0.5f, cheese->getPosition()));

    //芝士动作：逐渐消失
    cheese->runAction(FadeOut::create(3.0f));

    return true;
}
```

- 点击shoot按钮，石头发射到老鼠所在的位置，老鼠跑开，留下钻石。

```
//触摸事件
bool GameScene::onTouchBegan(Touch *touch, Event *unused_event) {

    auto location = touch->getLocation();
    auto cheese = Sprite::create("cheese.png");
    cheese->setPosition(mouseLayer->convertToNodeSpace(location));
    mouseLayer->addChild(cheese);

    //老鼠动作：移动到芝士位置
    mouse->runAction(MoveTo::create(0.5f, cheese->getPosition()));

    //芝士动作：逐渐消失
    cheese->runAction(FadeOut::create(3.0f));

    return true;
}
```

第三周

- 添加动画资源

```
    // 攻击动画
    attack.reserve(18);
    for (int i = 0; i < 17; i++) {
        auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(113
* i, 0, 113, 113)));
        attack.pushBack(frame);
    }
    attack.pushBack(frame0);
    auto attackAnimation = Animation::createWithSpriteFrames(attack, 0.1f);
    AnimationCache::getInstance()->addAnimation(attackAnimation, "attack");

    // 可以仿照攻击动画
    // 死亡动画(帧数：22帧，高：90，宽：79）
    auto texture2 = Director::getInstance()->getTextureCache()->addImage("$lucia_dead.png");
    dead.reserve(23);
    for (int i = 0; i < 22; i++) {
        auto frame = SpriteFrame::createWithTexture(texture2, CC_RECT_PIXELS_TO_POINTS(Rect(79
* i, 0, 79, 90)));
        dead.pushBack(frame);
    }
    dead.pushBack(frame0);
    auto deadAnimation = Animation::createWithSpriteFrames(dead, 0.1f);
    AnimationCache::getInstance()->addAnimation(deadAnimation, "dead");

    // 运动动画(帧数：8帧，高：101，宽：68）
    auto texture3 = Director::getInstance()->getTextureCache()->addImage("$lucia_forward.png");
    run.reserve(8);
    for (int i = 0; i < 8; i++) {
        auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(68
* i, 0, 68, 101)));
        run.pushBack(frame);
    }
    auto runAnimation = Animation::createWithSpriteFrames(run, 0.1f);
    AnimationCache::getInstance()->addAnimation(runAnimation, "run");
```

- 使用TTFConfig来预先设定字体格式，并使用TTFConfig来创建标签

```
    //使用TTFConfig来预先设定字体格式
    TTFConfig ttfConfig;
    ttfConfig.fontFilePath = "fonts/arial.ttf";
    ttfConfig.fontSize = 36;

    //使用TTFConfig创建标签
    auto WLabel = Label::createWithTTF(ttfConfig, "W");
    auto ALabel = Label::createWithTTF(ttfConfig, "A");
    auto SLabel = Label::createWithTTF(ttfConfig, "S");
    auto DLabel = Label::createWithTTF(ttfConfig, "D");
    auto XLabel = Label::createWithTTF(ttfConfig, "X");
    auto YLabel = Label::createWithTTF(ttfConfig, "Y");
```

- 添加菜单项

```
    //使用标签创建菜单项
    auto WItem = MenuItemLabel::create(WLabel, CC_CALLBACK_0(HelloWorld::WASDMenuCallback, this,
'W'));
    auto AItem = MenuItemLabel::create(ALabel, CC_CALLBACK_0(HelloWorld::WASDMenuCallback, this,
'A'));
    auto SItem = MenuItemLabel::create(SLabel, CC_CALLBACK_0(HelloWorld::WASDMenuCallback, this,
'S'));
    auto DItem = MenuItemLabel::create(DLabel, CC_CALLBACK_0(HelloWorld::WASDMenuCallback, this,
'D'));
    auto XItem = MenuItemLabel::create(XLabel, CC_CALLBACK_0(HelloWorld::XYMenuCallback, this,
'X'));
    auto YItem = MenuItemLabel::create(YLabel, CC_CALLBACK_0(HelloWorld::XYMenuCallback, this,
'Y'));

    //设置菜单项的位置
    WItem->setPosition(origin.x + 50, origin.y + 60);
    AItem->setPosition(origin.x + 20, origin.y + 30);
    SItem->setPosition(origin.x + 50, origin.y + 30);
    DItem->setPosition(origin.x + 80, origin.y + 30);
    XItem->setPosition(origin.x + visibleSize.width - 30, origin.y + 60);
    YItem->setPosition(origin.x + visibleSize.width - 60, origin.y + 30);

    //创建菜单并添加到游戏场景
    auto menu = Menu::create(WItem, AItem, SItem, DItem, XItem, YItem, NULL);
    menu->setPosition(Vec2::ZERO);
    this->addChild(menu, 1);
```

- 设置WASD菜单项回调函数，要保证角色不会移动到可视窗口外

```
//WASD菜单项的触发事件
void HelloWorld::WASDMenuCallback(char direction) {
    Vec2 distance = Vec2::ZERO; //要移动的向量
    float length = 30.0f;       //步长
    switch (direction) {
        case 'W':
            distance = Vec2(0, length);
            break;
        case 'A':
            distance = Vec2(-length, 0);
            break;
        case 'S':
            distance = Vec2(0, -length);
            break;
        case 'D':
            distance = Vec2(length, 0);
            break;
        default:
            break;
    }

    Vec2 pos = player->getPosition();
    Vec2 pos1 = pos;
    Vec2 pos2 = pos;
    pos1 -= Vec2(10, 10);   //pos1记录player的左下角
    pos2 += Vec2(10, 10);   //pos2记录player的右上角
    pos1.add(distance);     //pos1记录如果移动了player，会有的新的左下角
    pos2.add(distance);     //pos2记录如果移动了player，会有的新的右上角

    //x和y的范围
    int xMin = origin.x;
    int xMax = visibleSize.width + origin.x;
    int yMin = origin.y;
    int yMax = visibleSize.height + origin.y;

    //如果新的player的位置在可视窗口内，就移动player
    if (xMin <= pos1.x && pos2.x <= xMax && yMin <= pos1.y && pos2.y <= yMax) {
        Animate* animate = Animate::create(AnimationCache::getInstance()->getAnimation("run"));
        player->runAction(Repeat::create(animate, 1));
        player->runAction(MoveBy::create(0.8f, distance));
    }
}
```

- 设置XY菜单项回调函数，要保证X、Y播放的动画不会同时播放

```
/*
** HelloWorldScene.h
*/

bool isRunningAction;    //表示当前是否有attack/dead动画在运行


/*
** HelloWorldScene.cpp
*/

//X和Y菜单项的回调函数
void HelloWorld::XYMenuCallback(char item) {
    if (isRunningAction) return;     //如果已有attack/dead动画在运行，则不运行当前所选动画
    Animate* animate;
    if (item == 'X') {
        animate = Animate::create(AnimationCache::getInstance()->getAnimation("dead"));
        schedule(schedule_selector(HelloWorld::decreaseBlood), 0.1f, 19, 0);     //X对应掉血
    }
    else if (item == 'Y') {
        animate = Animate::create(AnimationCache::getInstance()->getAnimation("attack"));
        schedule(schedule_selector(HelloWorld::increaseBlood), 0.1f, 19, 0);     //Y对应回血
    }
    else return;
    isRunningAction = true;          //表示当前会有动画运行
    bool* ptrRunning = &isRunningAction;
    auto EndCallback = CallFunc::create([ptrRunning]() {         //用于放在序列动作的末尾，在运行完
一个动画后，即可重置isRunningAction
        *ptrRunning = false;
    });
    auto seq = Sequence::create(Repeat::create(animate, 1), EndCallback, nullptr);   //创建序列动
作
    player->runAction(seq);
}
```

- 点击X，减血；点击Y，回血

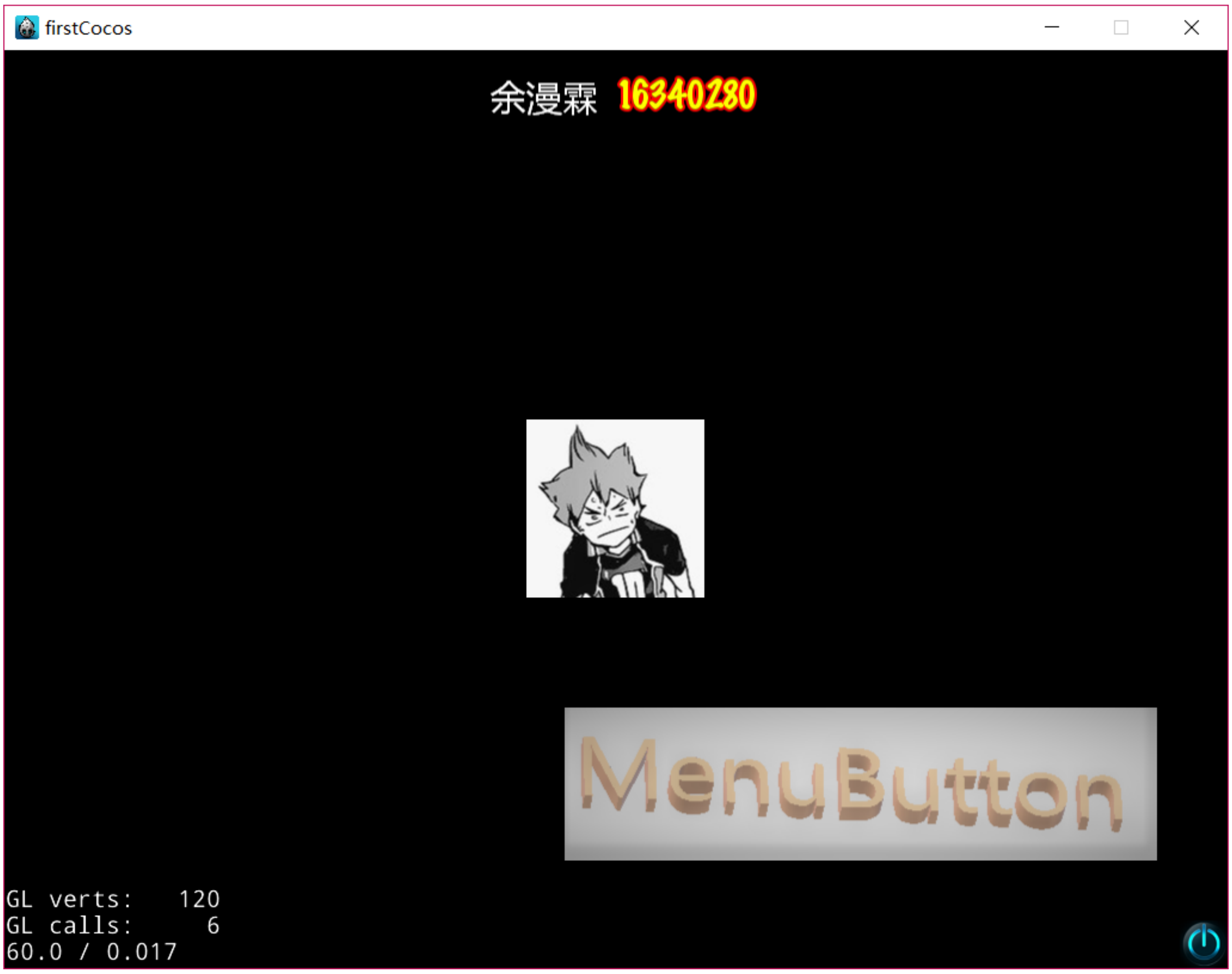
```
//回血
void HelloWorld::increaseBlood(float dt) {
    float per = pT->getPercentage();
    if (per == 100) return;
    else pT->setPercentage(++per);
}

//掉血
void HelloWorld::decreaseBlood(float dt) {
    float per = pT->getPercentage();
    if (per == 0) return;
    else pT->setPercentage(--per);
}
```

- 添加倒计时

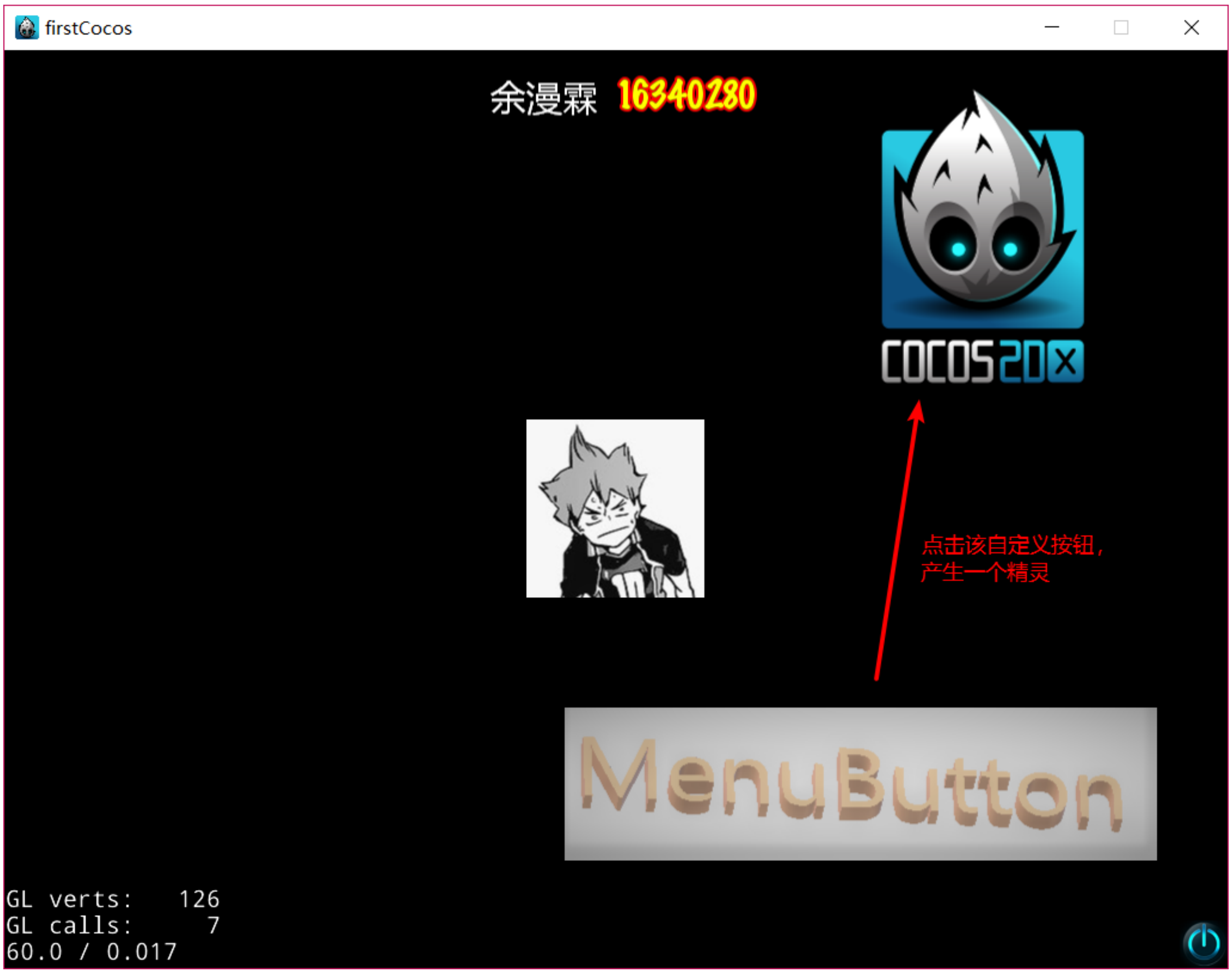
```
//bool HelloWorld::init()
    //设置倒计时
    dtime = 180;
    string str = "";
    stringstream ss;
    ss << dtime;
    ss >> str;
    time = Label::createWithTTF(ttfConfig, str);
    time->setPosition(origin.x + visibleSize.width / 2, origin.y + visibleSize.height - 30);
    this->addChild(time, 1);
    schedule(schedule_selector(HelloWorld::updateTime), 1.0f, dtime - 1, 0);


//自定义调度器，用于更新倒计时
void HelloWorld::updateTime(float dt) {
    dtime--;

    string str = "";
    stringstream ss;
    ss << dtime;
    ss >> str;

    time->setString(str);
}
```

三、关键步骤截图

第一周

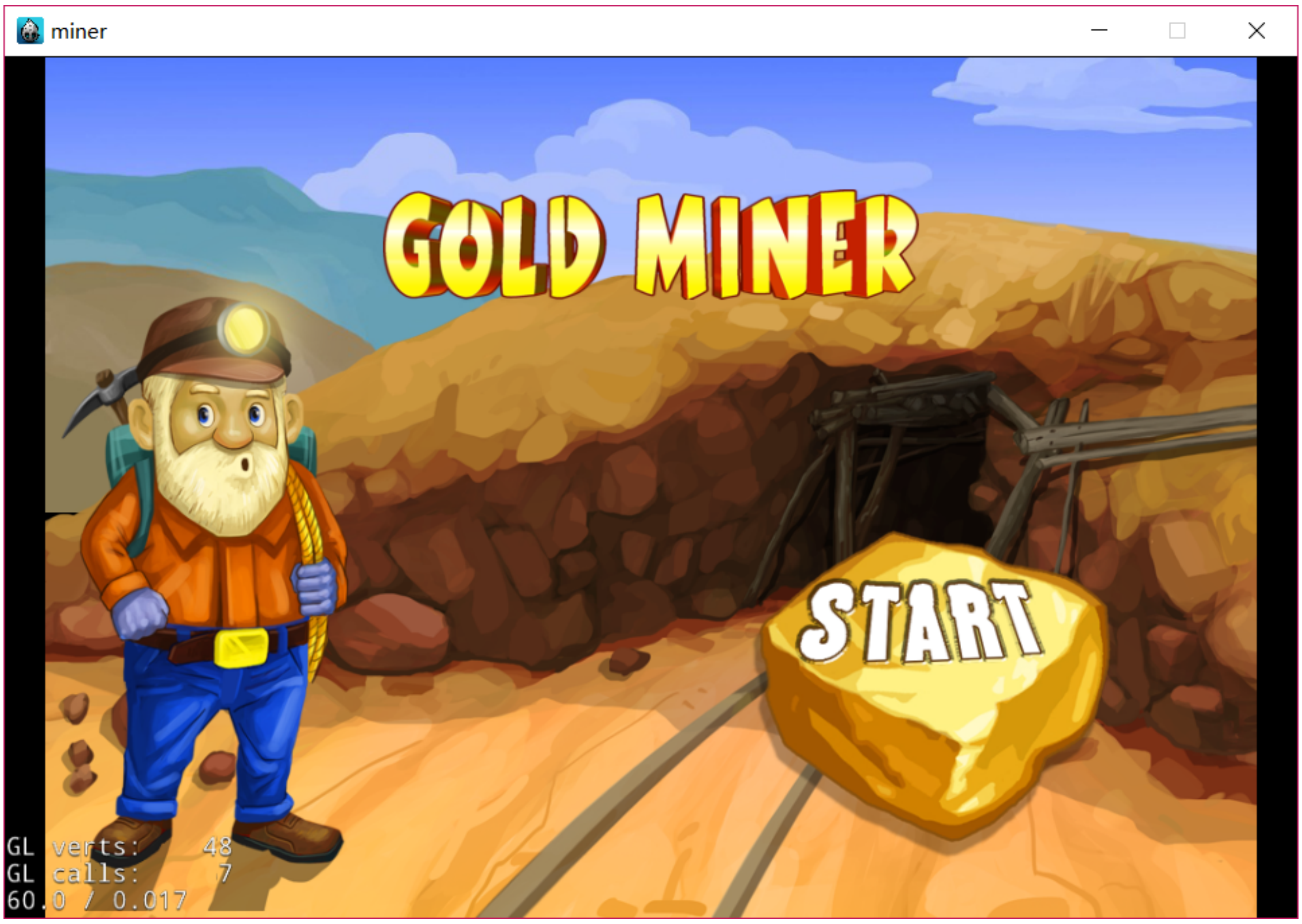


```
GL verts:    120
GL calls:      6
60.0 / 0.017
```

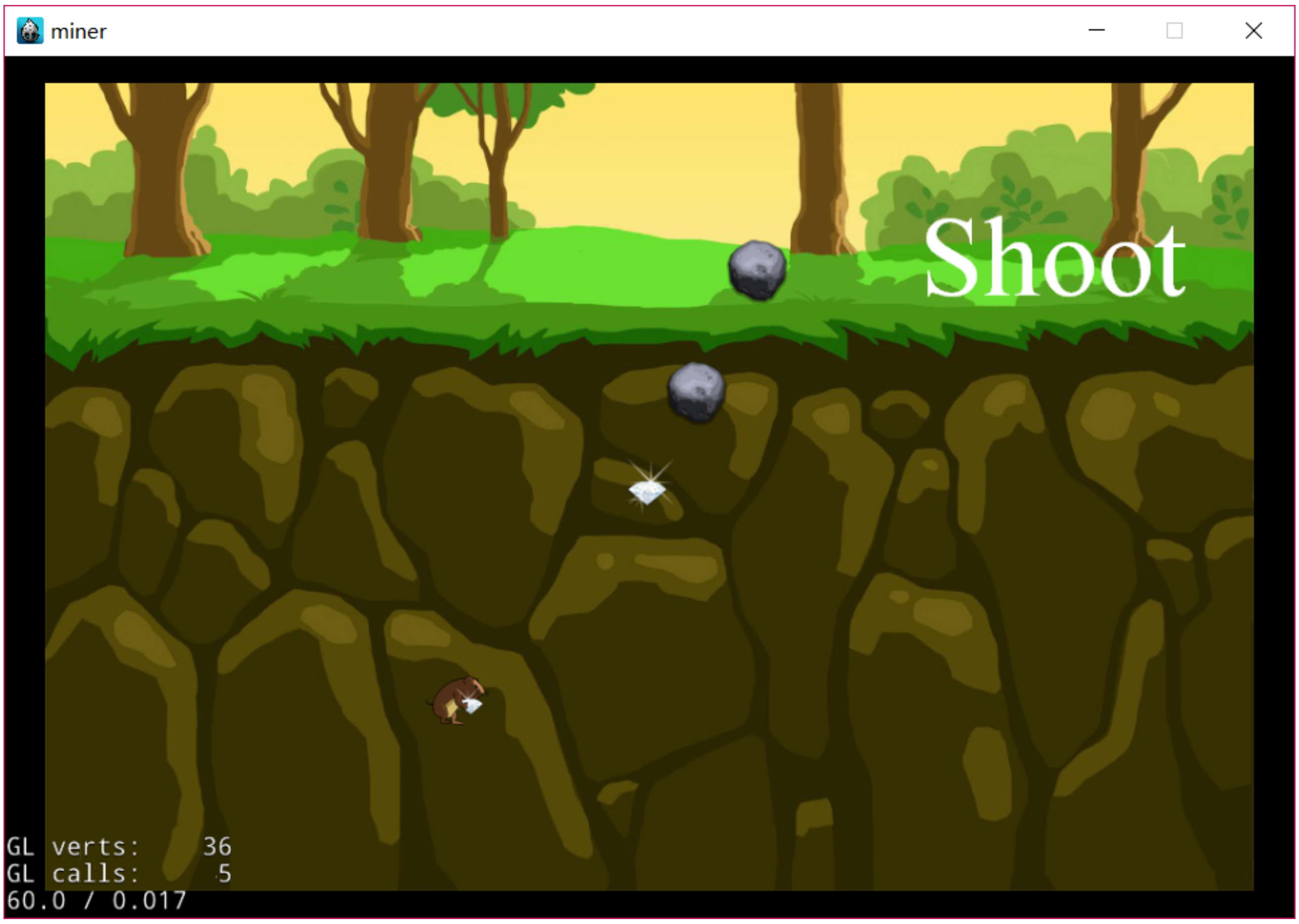


```
GL verts:    126
GL calls:      7
60.0 / 0.017
```

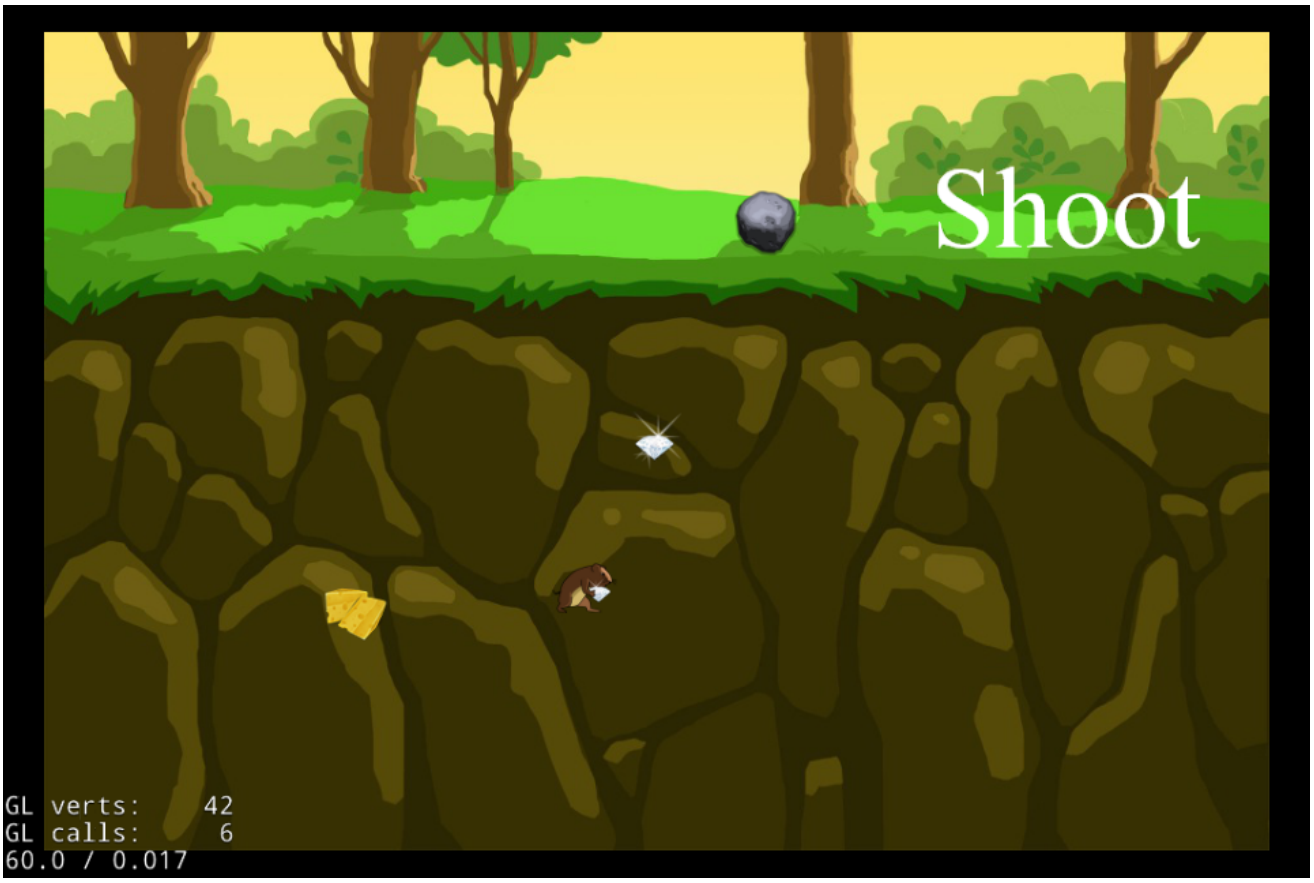
第二周



```
GL verts:     48
GL calls:      7
60.0 / 0.017
```

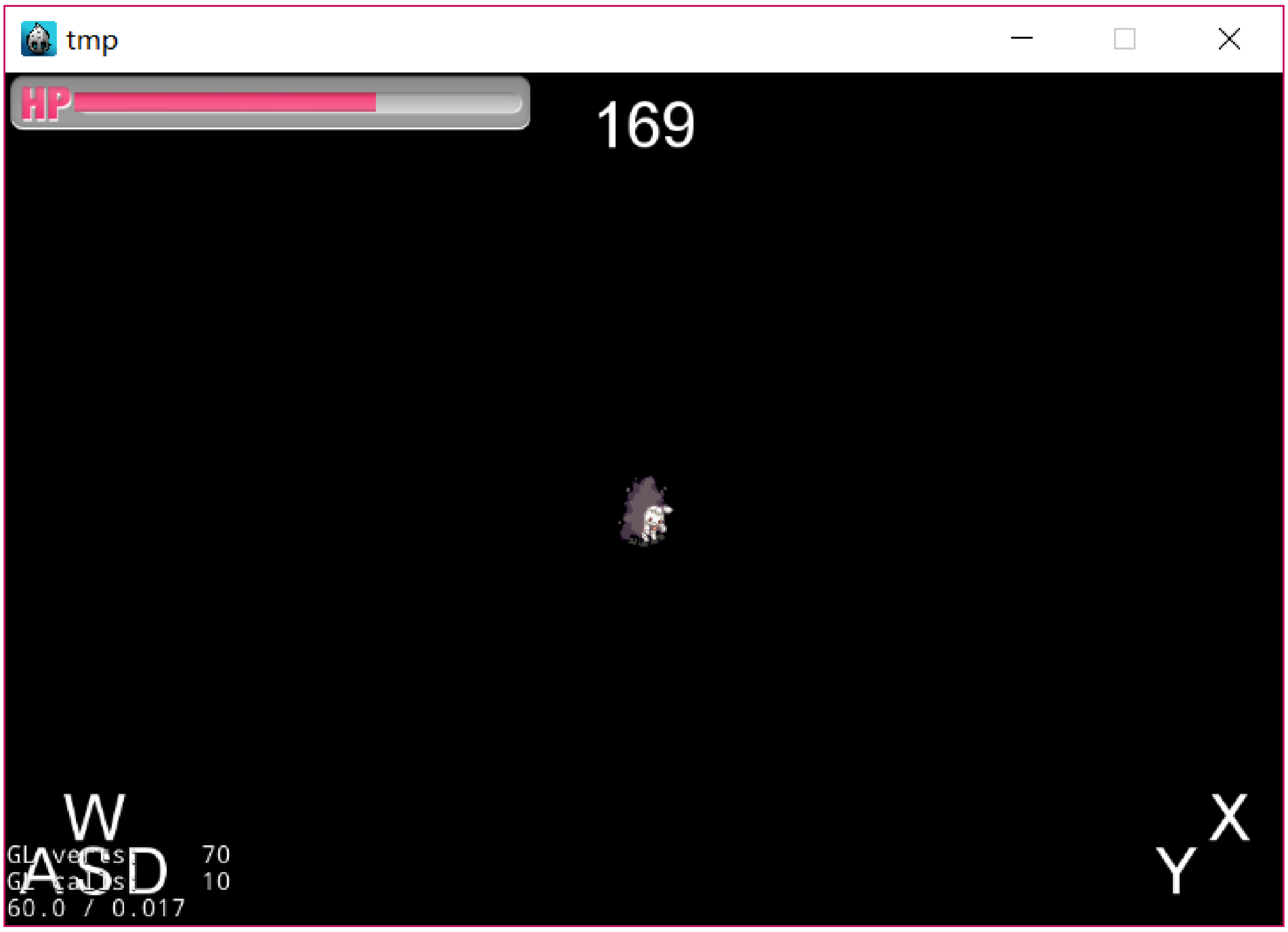


```
GL verts:     36
GL calls:      5
60.0 / 0.017
```

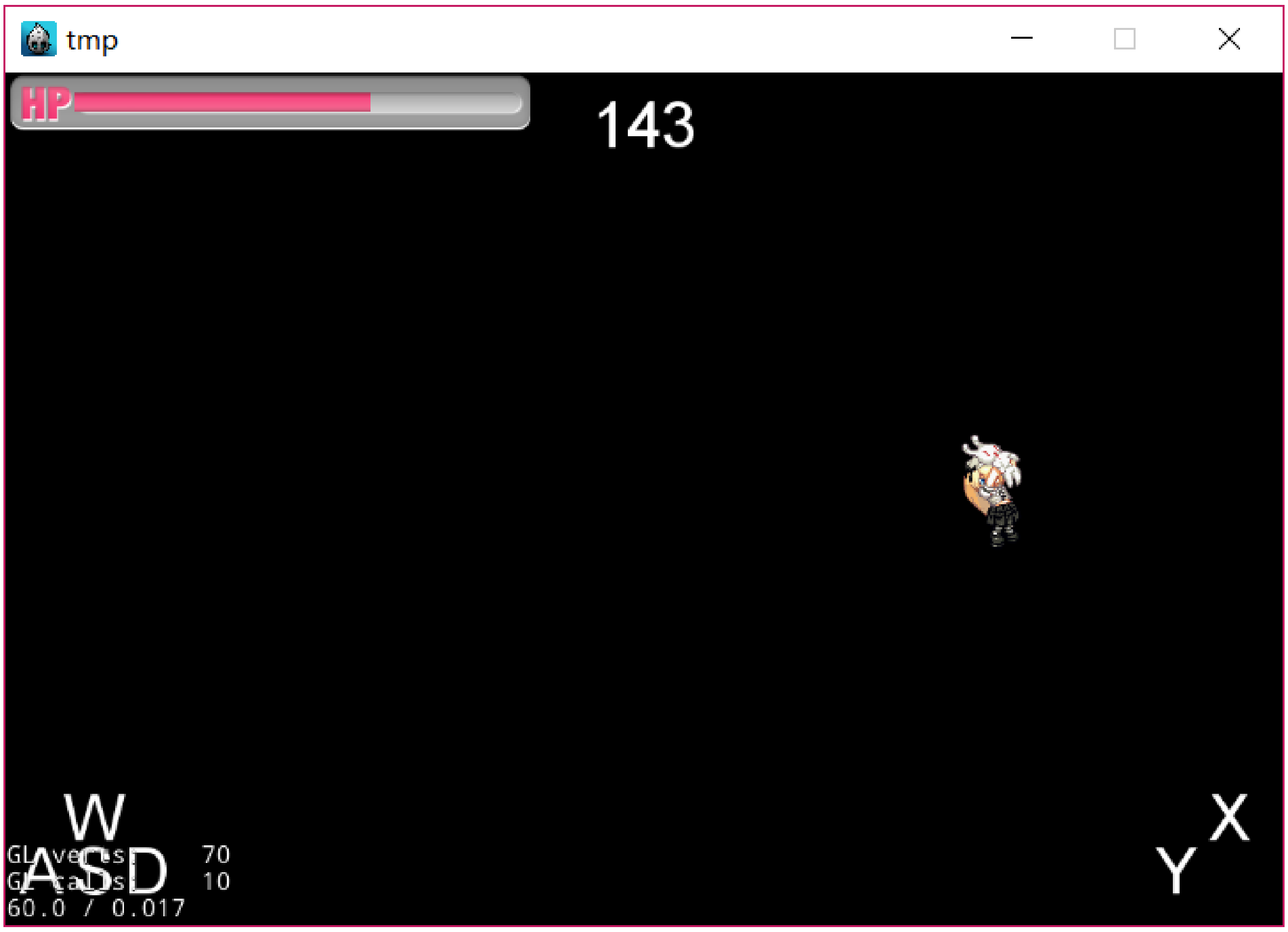


```
GL verts:     42
GL calls:      6
60.0 / 0.017
```

第三周



```
GL verts:     70
GL calls:     10
60.0 / 0.017
```



四、亮点与改进

第一周

- 设置文字样式
- 添加一个MenuItem
- 有简单的触发事件

第二周

- 添加人物动画——吹口哨

第三周

- 点击X，减血；点击Y，回血

以上各项的具体代码已放在【二、实验步骤】。

五、遇到的问题

第一周

- 从xml中读取的中文无法显示，只显示□□□这样的乱码。
  - 这是因为创建Label时所使用的字体不支持中文，改成另一支持中文的字体即可。

第三周

- 动画执行完后，不恢复到原来的状态。
    - 将原来的状态对应的frame也加入到存放帧动画的Vector容器的末尾。

-
  ```
  /*
  ** HelloWorldScene.h
  */

  bool isRunningAction;    //表示当前是否有attack/dead动画在运行


  /*
  ** HelloWorldScene.cpp
  */

  //X和Y菜单项的回调函数
  void HelloWorld::XYMenuCallback(char item) {
      if (isRunningAction) return;     //如果已有attack/dead动画在运行，则不运行当前所选动画
      Animate* animate;
      if (item == 'X') {
          animate = Animate::create(AnimationCache::getInstance()->getAnimation("dead"));
      }
      else if (item == 'Y') {
          animate = Animate::create(AnimationCache::getInstance()->getAnimation("attack"));
      }
      else return;
      isRunningAction = true;
      player->runAction(Repeat::create(animate, 1));
      isRunningAction = false;
  }
  ```

  一开始是试图通过这样来实现X和Y动画的互斥，但没有预期的效果。两个动画会并存。

    - 因为事实上runAction是非阻塞的，开始执行后，会在 `isRunningAction = false;` 这条语句执行完后才执行完毕。

      借鉴了这篇博客cocos2d-js动画结束监听里的思想，使用序列动作来检测动画是否执行完，执行完则通过回调操作进行相关的处理。

      ```
          isRunningAction = true;          //表示当前会有动画运行
          bool* ptrRunning = &isRunningAction;
          auto EndCallback = CallFunc::create([ptrRunning]() {         //用于放在序列动作的末尾，在运行完一个动画后，即可重置isRunningAction
              *ptrRunning = false;
          });
          auto seq = Sequence::create(Repeat::create(animate, 1), EndCallback, nullptr);   //创建序列动作
          player->runAction(seq);
      ```

六、思考与总结

- 通过本次学习，我了解到了cocos2d-x游戏的基本设计元素：场景、精灵、菜单、事件、动作等。将它们通过一定的规则有机地组合到一起，就可以得到一个游戏的大致框架。
- 了解到了帧动画的原理，可以通过SpriteSheet来添加帧，也可以直接从贴图中以像素单位分割，创建关键帧。将多个形象集中到同一张图片中，是很有利于减少内存的使用的。
- 了解到了丰富多样的动作，如移动、旋转、缩放、淡入淡出等，还有可以包括动作对象、函数（CallFunc对象）、甚至另一个序列的序列动作。利用好基本动作和序列动作，可以实现各种各样的游戏效果。
- cocos2d-x同样为许多元素的创建提供了丰富多样的实现方式。比如菜单项，可以通过图片、精灵或Label来创建。而Label的字体可以使用Resources自带的字体，也可以使用系统字体。而场景的切换同样有几十种方式。
- 回调函数是一个重要的概念，CC_CALLBACK_X中的X表示绑定第X个参数后面的所有参数的值。
- 总的来说，cocos2d-x实验很有意思，知识与乐趣并存。