# 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：** homework15

## 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/

## 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，了解网络和常用算法的设置。配置 java 环境，打开服务器

```
D:\15软工\现操\Pj 15\2017-week15serverfinal>java -jar server.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.3.3.RELEASE)

2017-06-01 20:19:36.353  INFO 244 --- [           main] hello.App
v0.0.1-SNAPSHOT on DESKTOP-07I3IVM with PID 244 (started by sst i
2017-06-01 20:19:36.446  INFO 244 --- [           main] hello.App
, falling back to default profiles: default
2017-06-01 20:19:37.446  INFO 244 --- [           main] ationConf
framework.boot.context.embedded.AnnotationConfigEmbeddedWebApplic
CST 2017]; root of context hierarchy
2017-06-01 20:19:45.618  INFO 244 --- [           main] o.s.b.f.s
ition for bean 'beanNameViewResolver' with a different definition
false; lazyInit=false; autowireMode=3; dependencyCheck=0; autowir
ingframework.boot.autoconfigure.web.ErrorMvcAutoConfiguration$Whi
meViewResolver; initMethodName=null; destroyMethodName=(inferred)
oot/autoconfigure/web/ErrorMvcAutoConfiguration$WhitelabelErrorVi
scope=; abstract=false; lazyInit=false; autowireMode=3; dependenc
yBeanName=org.springframework.boot.autoconfigure.web.WebMvcAutoCo
odName=beanNameViewResolver; initMethodName=null; destroyMethodNa
ngframework/boot/autoconfigure/web/WebMvcAutoConfiguration$WebMvc
2017-06-01 20:19:46.696  INFO 244 --- [           main] f.a.Autow
.Inject' annotation found and supported for autowiring
```

2. 实现“使用用户名登录”的功能

```
//创建用户名输入框
textField = TextField::create("Input your name", "fonts/arial.TTF", 50);
textField->setPosition(Size(visibleSize.width / 2, visibleSize.height / 4 * 3));
this->addChild(textField, 2);

//创建登录按钮
auto button = Button::create();
button->setTitleText("Login");
button->setTitleFontName("fonts/arial.TTF");
button->setTitleFontSize(50);
button->setPosition(Size(visibleSize.width / 2, visibleSize.height / 2));
button->addClickEventListener(CC_CALLBACK_1(LoginScene::loginButtonCallBack, this));
this->addChild(button, 2);
```

点击 Login 按钮，用 POST 类型向服务器发送登录请求，获取并处理服务器返回的数据

```
void LoginScene::loginButtonCallBack(cocos2d::Ref* pSender) {
    HttpRequest* request = new HttpRequest();
    request->setUrl("http://localhost:8080/login");
    request->setRequestType(HttpRequest::Type::POST);
    request->setResponseCallback(CC_CALLBACK_2(LoginScene::onHttpRequestCompleted, this));
    string textField_str = textField->getString();
    string post_str = "username=" + textField_str;
    const char * postData = post_str.c_str();
    request->setRequestData(postData, strlen(postData));
    request->setTag("POST Login");
    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

最后跳转到游戏场景

```
void LoginScene::onHttpRequestCompleted(HttpClient* sender, HttpResponse* response) {
    //处理response数据保存到global中
    if (!response) {
        return;
    }
    if (!response->isSucceed()) {
        log("response failed!\nerror buffer: %s", response->getErrorBuffer());
        return;
    }
    std::vector<char> * buffer = response->getResponseData();
    std::vector<char> * headertmp = response->getResponseHeader();
    string responseData = Global::toString(buffer);
    string header = Global::toString(headertmp);
    Global::gameSessionId = Global::getSessionIdFromHeader(header);
    log(responseData.c_str());

    //跳转到游戏场景
    auto scene = Scene::create();
    auto thunder = Thunder::create();
    scene->addChild(thunder);
    Director::getInstance()->replaceScene(TransitionFade::create(0.5, scene));
}
```

3. 游戏结束后跳转到分数场景，并传递 score 参数

```
//跳转到分数场景
auto scene = Scene::create();
auto gameScene = GameScene::create(score);
scene->addChild(gameScene);
Director::getInstance()->replaceScene(TransitionFade::create(0.5, scene));
return;
```

4. 实现“提交分数”的功能，用 POST 类型向服务器发送请求

```
void GameScene::submitButtonCallBack(cocos2d::Ref* pSender) {
    //发送网络请求，方式为POST
    HttpRequest* request = new HttpRequest();
    request->setUrl("http://localhost:8080/submit");
    request->setRequestType(HttpRequest::Type::POST);
    request->setResponseCallback(CC_CALLBACK_2(GameScene::onHttpRequestCompleted, this));

    //发送分数
    string post_str = "score=" + tempNum;
    const char * postData = post_str.c_str();
    request->setRequestData(postData, strlen(postData));
    request->setTag("POST Submit");

    //发送GAMESESSIONID
    vector<string> headers;
    headers.push_back("Cookie: GAMESESSIONID=" + Global::gameSessionId);
    request->setHeaders(headers);
    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

5. 实现“查询最好 n 位成绩”的功能

用 GET 类型向服务器发送查询最好位成绩请求，处理返回的数据

```
void GameScene::rankButtonCallBack(cocos2d::Ref* pSender) {
    HttpRequest* request = new HttpRequest();
    request->setUrl("http://localhost:8080/rank?top=10");
    request->setRequestType(HttpRequest::Type::GET);
    request->setResponseCallback(CC_CALLBACK_2(GameScene::onHttpRequestCompleted, this));
    request->setTag("GET Rank");

    vector<string> headers;
    headers.push_back("Cookie: GAMESESSIONID=" + Global::gameSessionId);
    request->setHeaders(headers);

    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

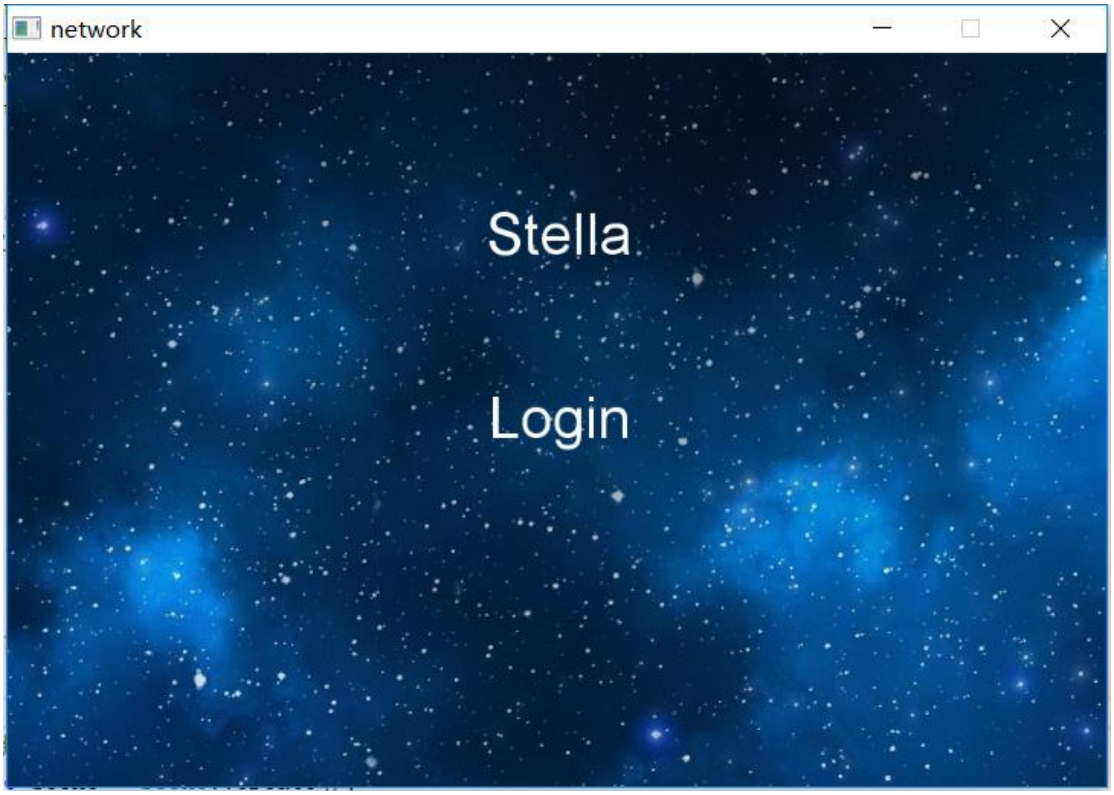
检查返回数据格式正确后，取“|“之间的内容加入到 vector<string>容器中并显示

```
std::vector<char> * buffer = response->getResponseData();
string responseData = Global::toString(buffer);
log(responseData.c_str());

//rank 信息处理
const char* tag_type = response->getHttpRequest()->getTag();
if (!strcmp(tag_type, "GET Rank")) {
    rapidjson::Document doc;
    doc.Parse<0>(responseData.c_str());
    if (doc.HasParseError()) {
        log("GetParseError: %s", doc.GetParseError());
    }
    if (doc.IsObject() && doc.HasMember("info")) {
        string rank_info = doc["info"].GetString();
        log(rank_info.c_str());

        vector<string> rankInfo_vec;
        string tempstr = rank_info.substr(1);
        int index = 0;
        while (tempstr != "") {
            index = tempstr.find('|');
            string find_str = tempstr.substr(0, index);
            rankInfo_vec.push_back(find_str);
            tempstr = tempstr.substr(index + 1);
        }
        string rank_field_text = "";
        for (index = 0; index < rankInfo_vec.size(); index++) {
            rank_field_text += (rankInfo_vec[index] + "\n");
        }
        rank_field->setText(rank_field_text);
    }
```

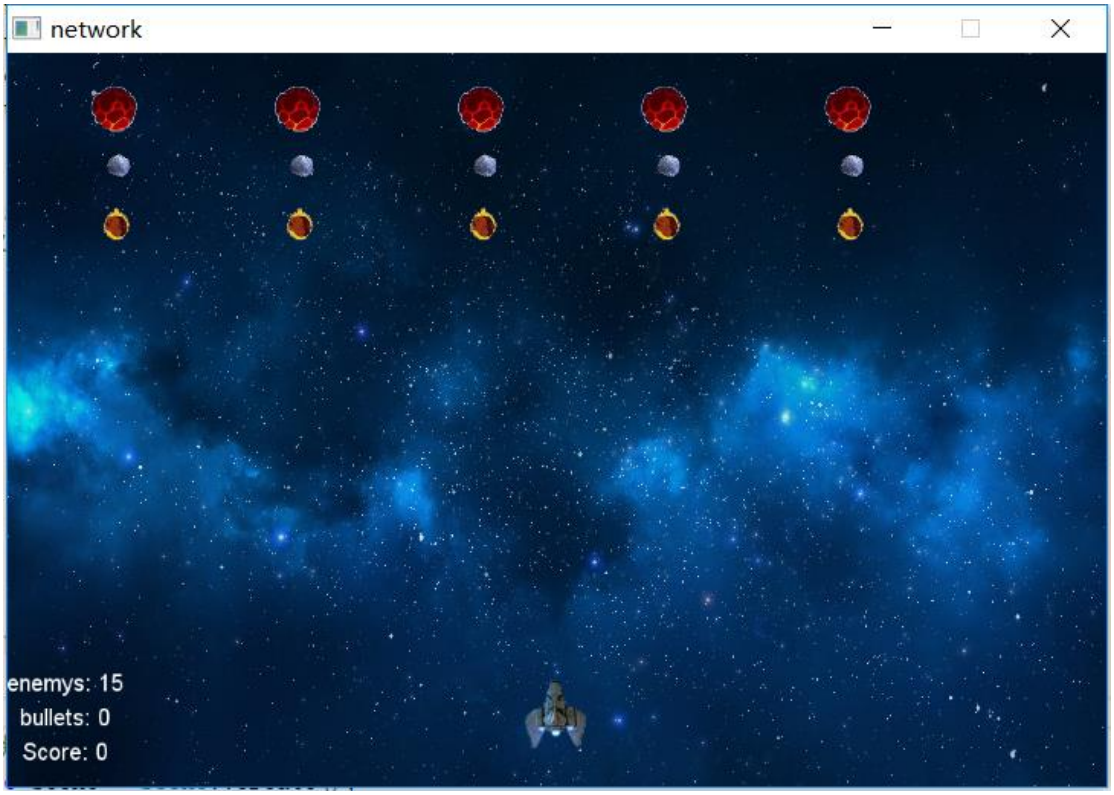
6. 调试项目

## 三．实验结果截图

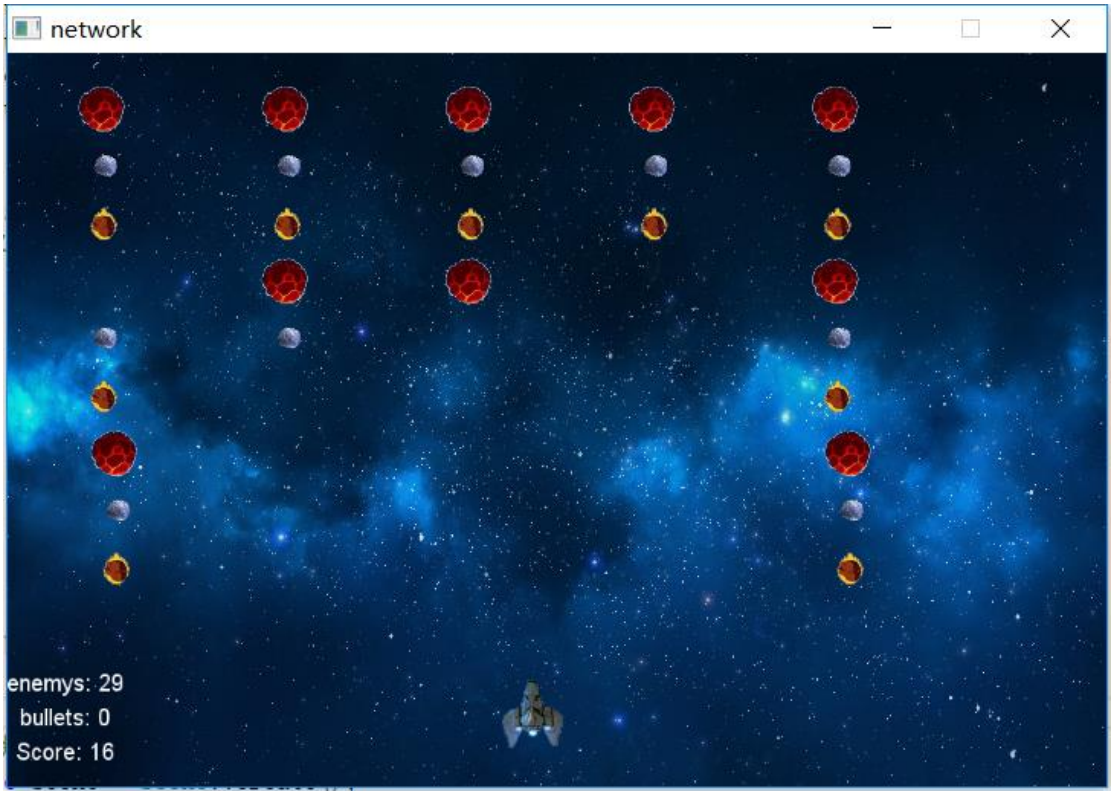
1. 打开窗口，输入用户名




2. 点击 Login，跳转到游戏界面



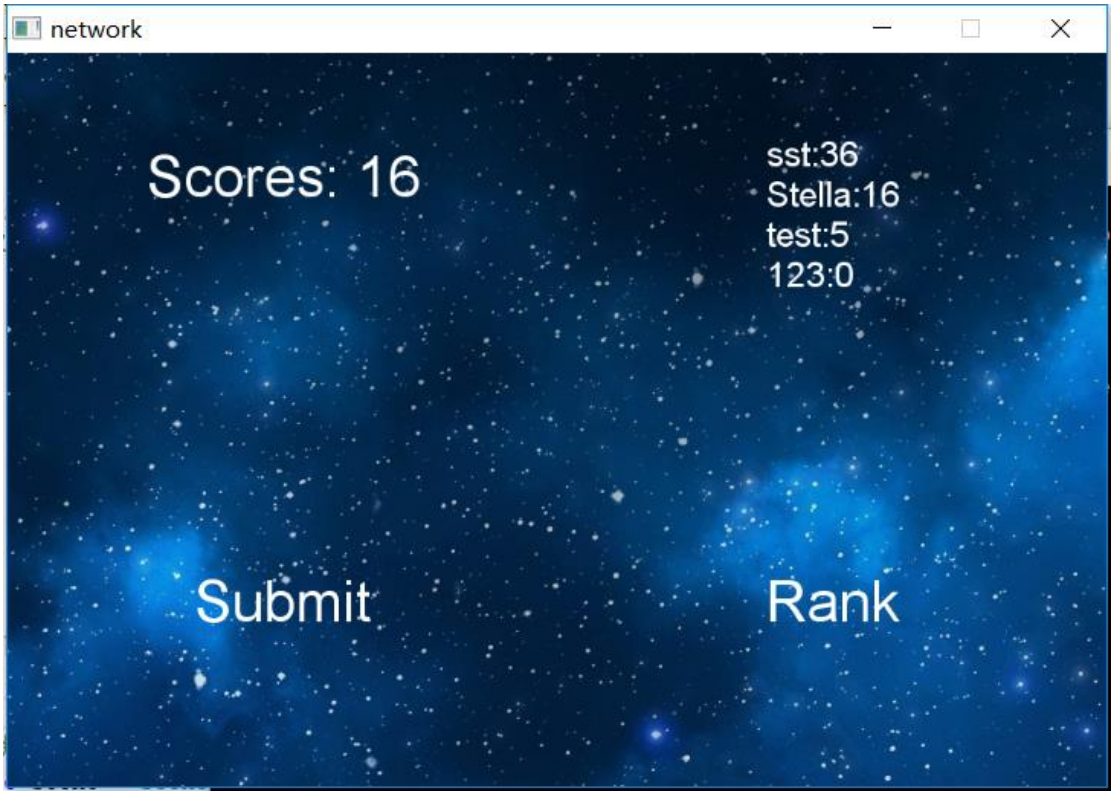

正常进行游戏




3. 游戏结束，跳转到分数界面，点击 Submit 提交分数后，点击 Rank 显示排名

点击 Submit 提交分数 {"result":true,"info":"16"}

点击 Rank 显示排名 {"result":true,"info":"|sst:36|Stella:16|test:5|123:0|"}
|sst:36|Stella:16|test:5|123:0|




## 四．实验过程遇到的问题

1. 游戏得分参数传递到分数场景失败，将 gameSence 的 create 函数改成如下：

```
static GameScene* create(int num) {
    GameScene* pRet = new(std::nothrow) GameScene();
    if (pRet && pRet->init(num)) {
        pRet->autorelease();
        return pRet;
    }
    delete pRet;
    pRet = NULL;
    return NULL;
}
```

## 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。
4. 最后一个实验作业了，现操课很有意思，TA 辛苦，完结撒花。