现代操作系统应用开发实验报告

学号： 14331388 **班级 ：** 教务三班

姓名： 郑泽佳 **实验名称：** 实验 14

一．参考资料

课件和作业要求

http://www.cocos.com/doc/article/index?type=cocos2d-x&url=/doc/cocos-docs-master/manual/framework/native/v3/httpclient-session/zh.md

二．实验步骤

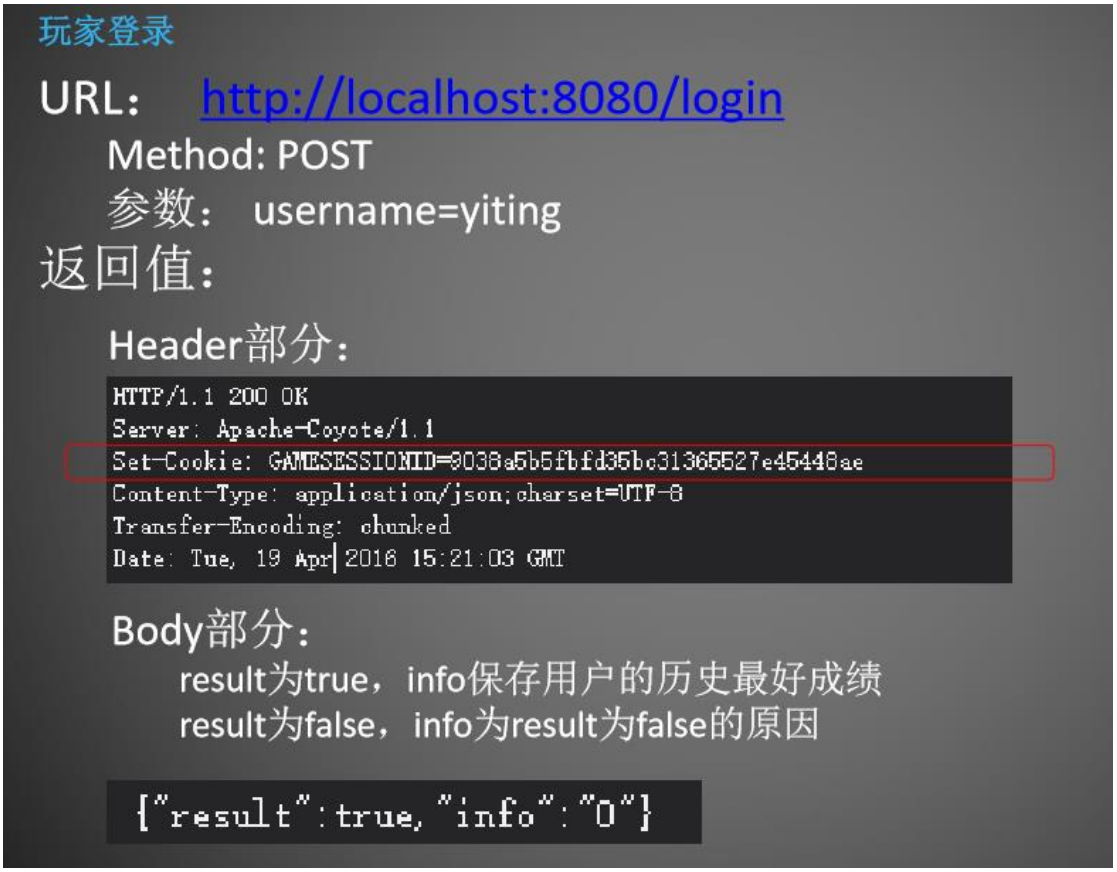
注：实验平台：win32

1、实现“使用用户名登录”的功能

在 LoginScene 界面中卫 Button 添加点击事件

```
button->addClickEventListener(CC_CALLBACK_0(LoginScene::Login, this));
```

根据下图信息，再结合课件中 HttpCliend 使用方法的 7 个步骤，可以得到代码



点击事件调用函数：

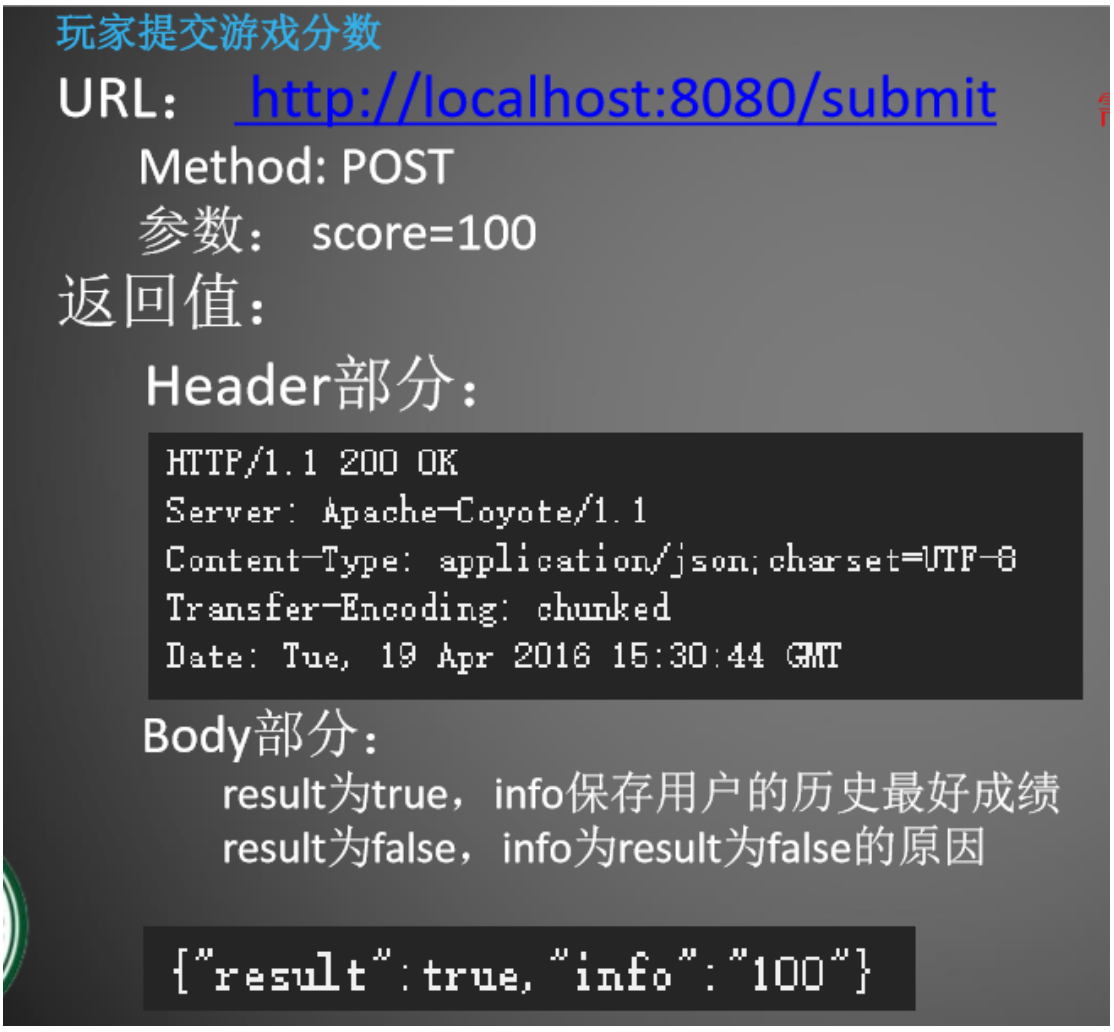
```
void LoginScene::Login() {
    HttpRequest* request = new HttpRequest();
    request->setUrl("http://localhost:8080/login");
    request->setRequestType(HttpRequest::Type::POST);
    request->setResponseCallback(CC_CALLBACK_2(LoginScene::onHttpRequestCompleted, this));
    string _postData = "username="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

回调函数：

```cpp
void LoginScene::onHttpRequestCompleted(HttpClient *sender, HttpResponse *response) {
    if (!response) {
        return;
    }
    if (!response->isSucceed()) {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }
    //get SessionId
    std::vector<char> *header_buffer = response->getResponseHeader();
    Global::gameSessionId = Global::getSessionIdFromHeader(Global::toString(header_buffer));

    std::vector<char> *buffer = response->getResponseData();
    rapidjson::Document d;
    d.Parse<0>(Global::toString(buffer).c_str());
    if (d.HasParseError()) //打印解析错误
    {
        log("GetPareseError %s\n", d.GetParseError());
        return;
    }
    if (d.IsObject() && d.HasMember("result")) {
        if (d["result"].GetBool()) {  //登陆成功
            auto scene = GameScene::createScene();
            Director::getInstance()->replaceScene(scene);
        }
```

2、实现“提交分数”的功能



跟步骤一类似，根据上图信息，实现代码如下，只不过需要在请求的 Header 部分加入 GAMESESSIONID ,而 GAMESESSIONID 在步骤一中已做处理将数据保存在 Global 中的静态变量 gameSessionId，还有在确定数据无误后，并不需要回调函数

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

点击事件：

```
void GameScene::Submit() {
    HttpRequest* request = new HttpRequest();
    request->setUrl("http://localhost:8080/submit");
    request->setRequestType(HttpRequest::Type::POST);
    //request->setResponseCallback(CC_CALLBACK_2(GameScene::onHttpRequestCompleted, this));
    string _postData = "score=" + score_field->getString();
    const char* postData = _postData.c_str();
    request->setRequestData(postData, strlen(postData));
    request->setTag("Post Submit");

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

3、实现“查询最好 n 位成绩”的功能

同样使用服务器提供的 API 进行操作，因为方法为 GET，不同于 POST 方法设置参数的方法，GET 只需要在 URL 末尾直接加入参数,例如 “?top=10”.

点击事件

```
void GameScene::Rank() {
    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");
    //在请求的Header部分加入GAMESESSIONID
    vector<string> headers;
    headers.push_back("Cookie: GAMESESSIONID=" + Global::gameSessionId);
    request->setHeaders(headers);
    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

回调函数：

```
void GameScene::onHttpRequestCompleted(HttpClient *sender, HttpResponse *response) {
    if (!response) {
        return;
    }
    if (!response->isSucceed()) {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }
    std::vector<char> *buffer = response->getResponseData();
    rapidjson::Document d;
    d.Parse<0>(Global::toString(buffer).c_str());
    if (d.HasParseError()) //打印解析错误
    {
        log("GetPareseError %s\n", d.GetParseError());
        return;
    }
    if (d.IsObject() && d.HasMember("result")) {
        if (d["result"].GetBool()) {
            string top_rank = d["info"].GetString();
            top_rank.erase(top_rank.begin());
            for (unsigned int i = 0; i < top_rank.length(); i++) {
                if (top_rank[i] == '|')
                    top_rank[i] = '\n';
            }
            rank_field->setString(top_rank);  //实现排行榜
```

4、HttpClient 的 enableCookies 函数的作用（加分项）

HttpClient::enableCookies 方法支持 Http 会话使用 cookie。 由于 sessionID 的保存和传递是通过 cookie 实现的，所以我们可以很方便的使用 HttpClient::enableCookies 方法来实现 http client session，这是 sessionId 的另一种实现方法。

也就是说在需要登陆界面中加入

```
cocos2d::network::HttpClient::getInstance()->enableCookies(NULL);
```

就不需要在游戏界面中额外地在请求的 Header 部分加入 GAMESESSIONID，因为 cookie 已将 GAMESESSIONIN 保存传递给 submit 和 rank 中的请求，因此可以将下面语句注释掉

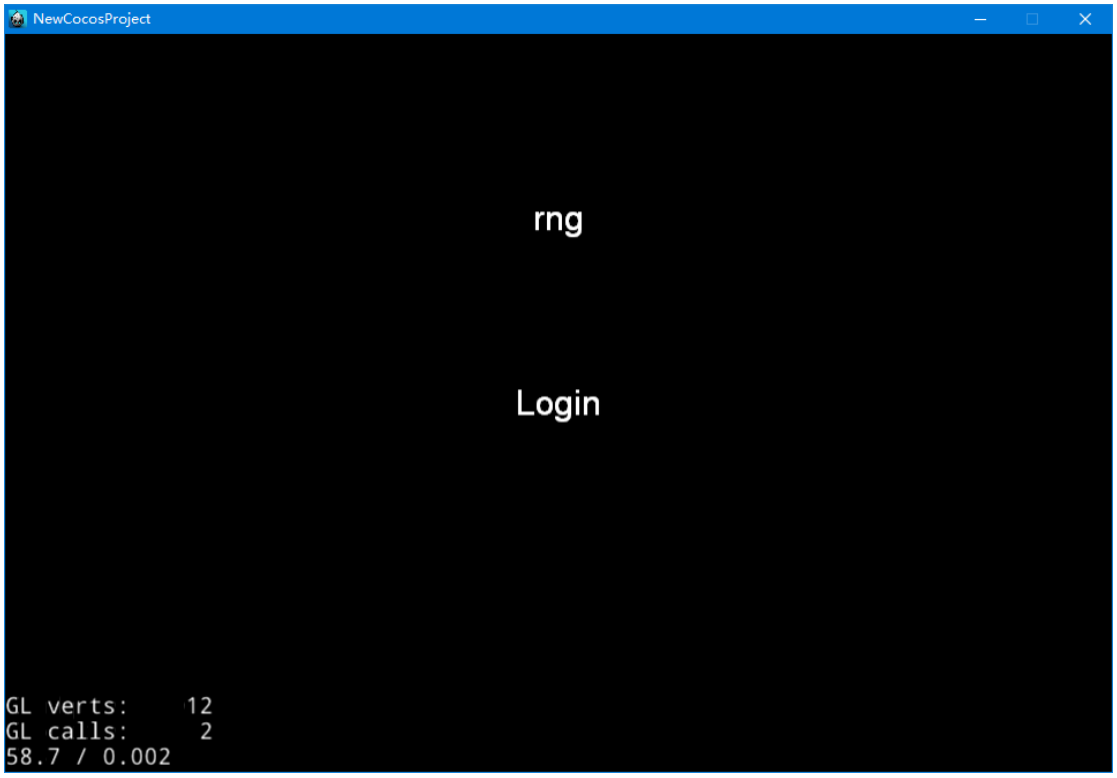
```
//get SessionId
std::vector<char> *header_buffer = response->getResponseHeader();
Global::gameSessionId = Global::getSessionIdFromHeader(Global::toString(header_buffer));
```

```
//在请求的Header部分加入GAMESESSIONID
vector<string> headers;
headers.push_back("Cookie: GAMESESSIONID=" + Global::gameSessionId);
request->setHeaders(headers);
```

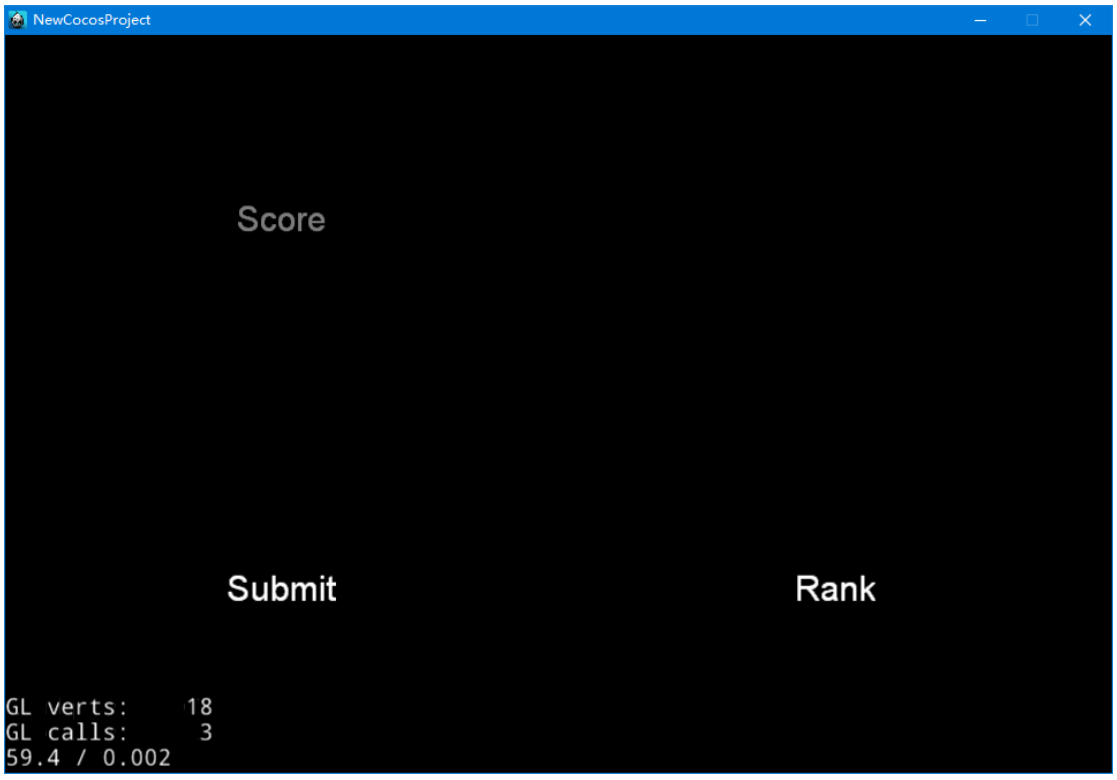
三．实验结果截图

启动服务器

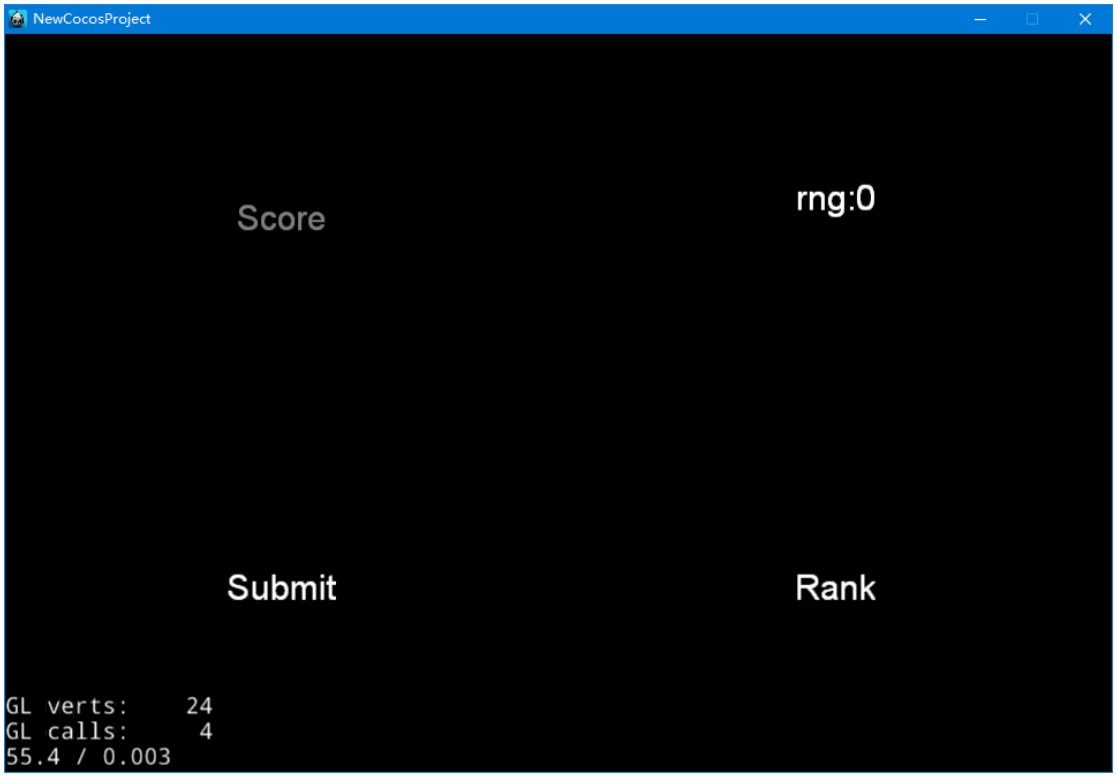
运行程序，程序界面



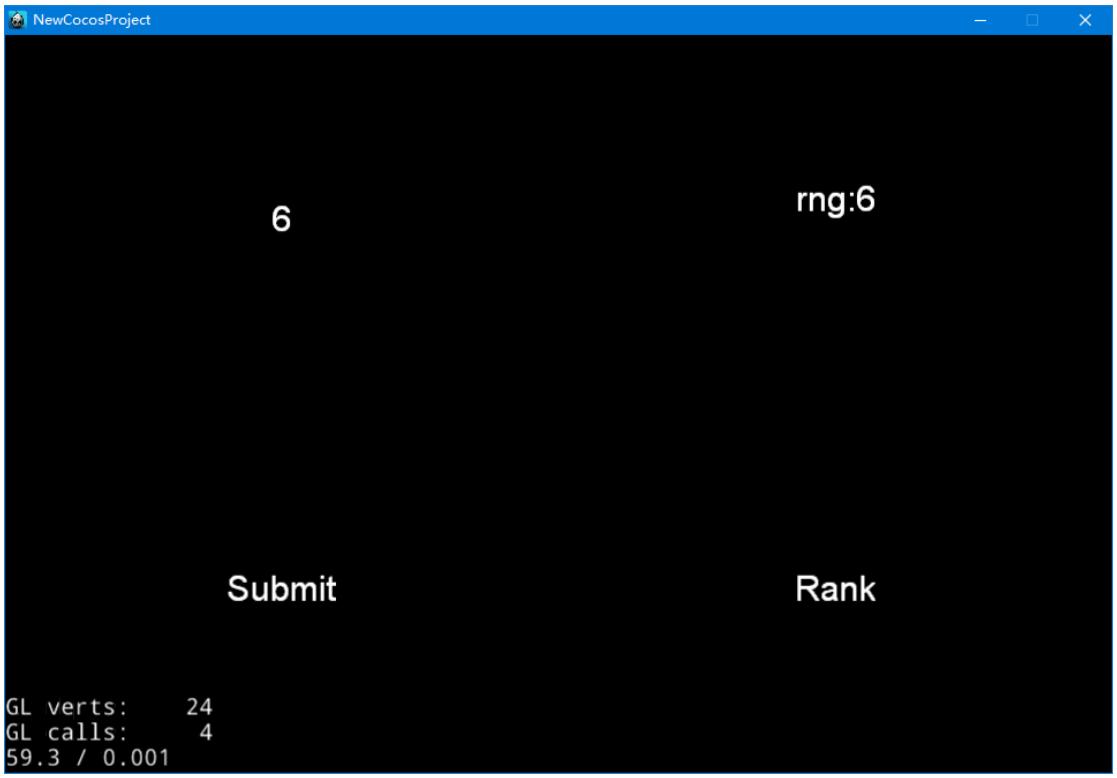
登陆



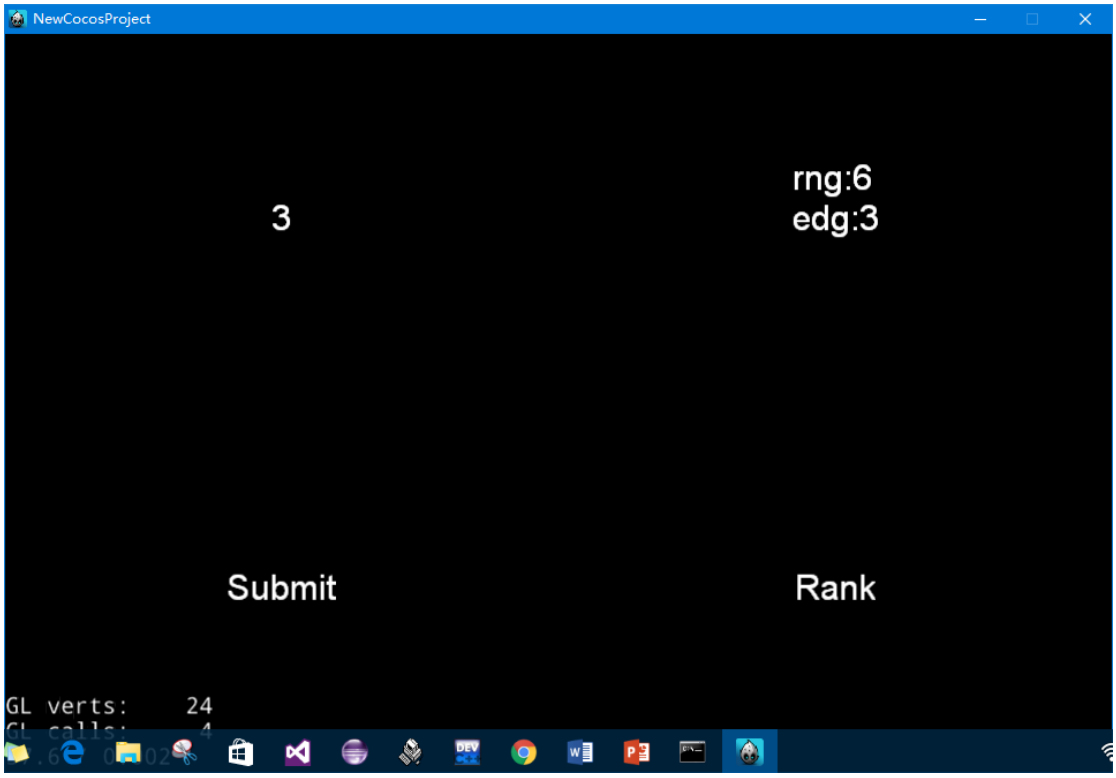
点击 rank 显示排行



输入分数，点击 Submit，再点击 Rank,可以看到分数更新了



重新启动程序，并换个用户名登陆



四．实验过程遇到的问题

问题一：在查看分数排行榜 Body 部分的返回值一直为 false，info 内容显示 no provided top，但代码中按如下图方式已设置好

```
string _postData = "top=10";
const char* postData = _postData.c_str();
request->setRequestData(postData, strlen(postData));
```

出错原因：因为该请求方式为 GET,所以与 POST 方法不同，设置参数方法不同。

```
request->setUrl("http://localhost:8080/rank?top=10");
```

五．思考与总结

实验总结:

本次实验其实比较简单，只需要处理下从服务器中的得到的数据，内部的操作服务器也已经全部弄好，甚至连出错原因服务器的返回值也有明确指示，调试起来很方便。

这次实验做的加分项是思考 HttpClient 的 enableCookies 函数的作用，看了一下网站

资料，然后应用下发现只需要 enableCookies 这一句话就可以实现这次功能，在 sumbit 和 rank 也就不需要在做 Gamesessionid 进行处理，着实方便，所以对于该函数的作用的认识可能不是很准确。