现代操作系统应用开发实验报告

学号： 14331098　　**班级：** 周四班

姓名： 黄建武　　**实验名称：** HW14

实验平台：**win32**

一．参考资料

本周课件

二．实验步骤

① 阅读 demo 的源代码，理解代码的思路。

② 阅读实验要求，将实验分成几部分，逐步完成；

③ 实现登录功能

```
// 登录按钮
auto loginLabel = Label::createWithTTF("L o g i n", "fonts/Marker Felt.ttf", 48);
loginLabel->setColor(Color3B(247, 147, 0));
auto loginBtn = MenuItemLabel::create(loginLabel, CC_CALLBACK_1(LoginScene::loginClick,
 this));
Menu* login = Menu::create(loginBtn, NULL);
login->setPosition(visibleWidth / 2 + 30, visibleHeight / 2);
this->addChild(login);
```

```
// 点击登录按钮
void LoginScene::loginClick(Ref * pSender) {
  if (userName->getString() != "") {
    HttpRequest* req = new HttpRequest();
    req->setRequestType(HttpRequest::Type::POST);
    req->setUrl("localhost:8080/login");
    string data = "username=" + userName->getString();
    req->setRequestData(data.c_str(), data.size());
    req->setResponseCallback(CC_CALLBACK_2(LoginScene::onLoginCompleted, this));
    HttpClient::getInstance()->send(req);
    req->release();
  }
}
```

在提取响应头部时，因为返回的是一个 vector<char>*，TA 的 demo 中特意写了一个函数就行转换，而 C++的 STL 中的 string 的构造函数就可以直接用一个 vector<char>进行构造，所以删了 TA 的转换函数，直接使用 string 的构造函数。

```
// 登录响应
void LoginScene::onLoginCompleted(HttpClient * sender, HttpResponse * res) {
  if (res == nullptr) return;

  // 登录成功，获取ResponseHeader,提取gameSessionId,并保存用户名,下次可直接登录
  // 跳转到游戏界面
  if (res->isSucceed()) {
    string resHeader(res->getResponseHeader()->begin(), res->getResponseHeader()->end());
    Global::gameSessionId = Global::getSessionIdFromHeader(resHeader);
    database->setStringForKey("username", userName->getString());
    Director::getInstance()->replaceScene(TransitionCrossFade::create(0.3f,
     Breakout::createScene()));
  }
  else {
    log("Login failed: %s", res->getErrorBuffer());
  }
}
```

④ 实现提交分数功能；

```
// 提交按钮
label = Label::createWithTTF("Submit", "fonts/Marker Felt.ttf", 40);
label->setColor(Color3B(247, 147, 0));
auto submitBtn = MenuItemLabel::create(label, CC_CALLBACK_1(GameOver::submitCallback,
 this));
Menu* submit = Menu::create(submitBtn, NULL);
submit->setPosition(visibleSize.width / 2 - 80, visibleSize.height / 2 - 100);
this->addChild(submit);
```

在提交分数的 post 请求时，一开始的 post 数据错误导致提交失败，这个在后面有详细说明。

```
// 点击提交按钮
void GameOver::submitCallback(Ref* sender) {
  HttpRequest* req = new HttpRequest();
  req->setRequestType(HttpRequest::Type::POST);
  req->setUrl("localhost:8080/submit");
  req->setResponseCallback(CC_CALLBACK_2(GameOver::onsubmitCompleted, this));
  vector<string> header;
  header.push_back("Cookie: " + Global::gameSessionId);
  req->setHeaders(header);
  string data = "score=" + score->getString().substr(12);
  req->setRequestData(data.c_str(), data.size());
  HttpClient::getInstance()->send(req);
  req->release();
}
```

```
// 提交响应函数,如果成功提交则更新排名
void GameOver::onsubmitCompleted(HttpClient* sender, HttpResponse* res) {
  if (res == nullptr) return;
  if (res->isSucceed())
    rankCallback(nullptr);
  else {
    log("Login failed: %s", res->getErrorBuffer());
  }
}
```

⑤ 实现查询前 n 名的功能；

```
// 排名按钮
label = Label::createWithTTF("Rank", "fonts/Marker Felt.ttf", 40);
label->setColor(Color3B(247, 147, 0));
auto rankBtn = MenuItemLabel::create(label, CC_CALLBACK_1(GameOver::rankCallback, this));
Menu* rank = Menu::create(rankBtn, NULL);
rank->setPosition(visibleSize.width / 2 + 45, visibleSize.height / 2 - 100);
this->addChild(rank);
```

```
// 点击排名按钮,最多查询前10名,默认查前十名,
void GameOver::rankCallback(Ref * sender) {
  if (topx->getString() == "0")
    topx->setString("");
  string top = (topx->getString() != "" ? topx->getString() : "10");
  HttpRequest* req = new HttpRequest();
  req->setRequestType(HttpRequest::Type::GET);
  req->setUrl(("localhost:8080/rank?top=" + top).c_str());
  req->setResponseCallback(CC_CALLBACK_2(GameOver::onrankCompleted, this));
  vector<string> header;
  header.push_back("Cookie: " + Global::gameSessionId);
  req->setHeaders(header);
  HttpClient::getInstance()->send(req);
  req->release();
}
```

因为返回的数据是用'|'做分隔符，将|替换成换行，实现一个排名一行。

```
// 排名响应函数,成功则显示前n名排名
void GameOver::onrankCompleted(HttpClient* sender, HttpResponse* res) {
  if (res == nullptr) return;
  if (res->isSucceed()) {
    string resData(res->getResponseData()->begin(), res->getResponseData()->end());
    rapidjson::Document json;
    json.Parse(resData.c_str());
    if (!json.HasParseError() && json.HasMember("result") && json["result"].GetBool()) {
      string rankRes = json["info"].GetString();
      rankRes.erase(0, 1);
      for (size_t i = 0; i < rankRes.length(); ++i)
        if (i && rankRes[i] == '|')
          rankRes[i] = '\n';
      rank_field->setString(rankRes);
    }
  }
  else {
    log("Login failed: %s", res->getErrorBuffer());
  }
}
```

⑥ 实现自动登录；

实验要求是储存 GAMESESSIONID 实现自动登录，可是存储了 GAMESSIONID 后我发现可以自动登录可是无法得知当前用户名，而我在游戏失败的页面要显示当前用户名和分数，所以改用储存用户名，下次运行就直接发送 post 请求实现自动登录。

```
// 记住登录,如果已登录直接登录
if (database->getStringForKey("username") != "") {
  userName->setString(database->getStringForKey("username"));
  loginClick(nullptr);
}
return true;
```

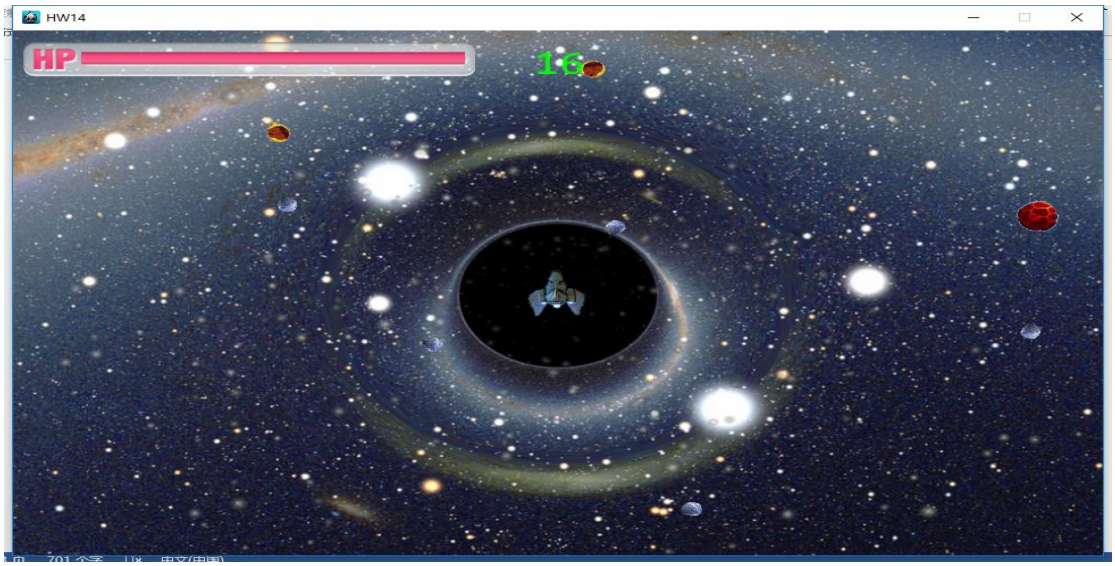
⑦ 将本次作业与上次作业结合，组成一个游戏；

⑧ 进行最后的测试。

三．实验结果截图

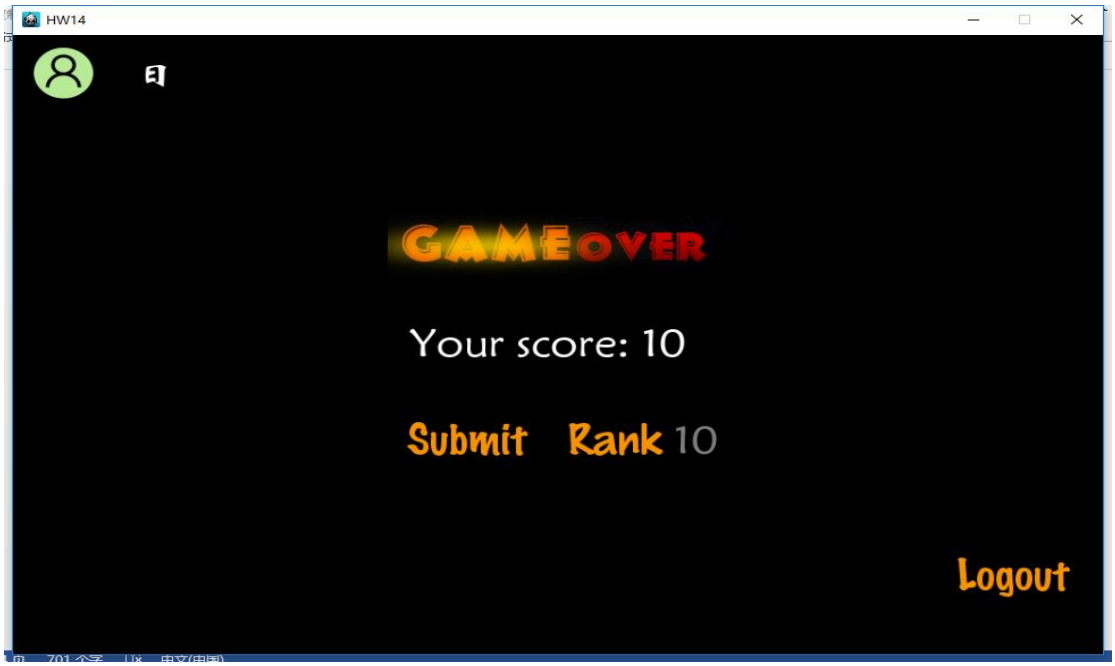
登录界面，用户名最多 8 位字符，可输入中文

登陆后进入游戏界面

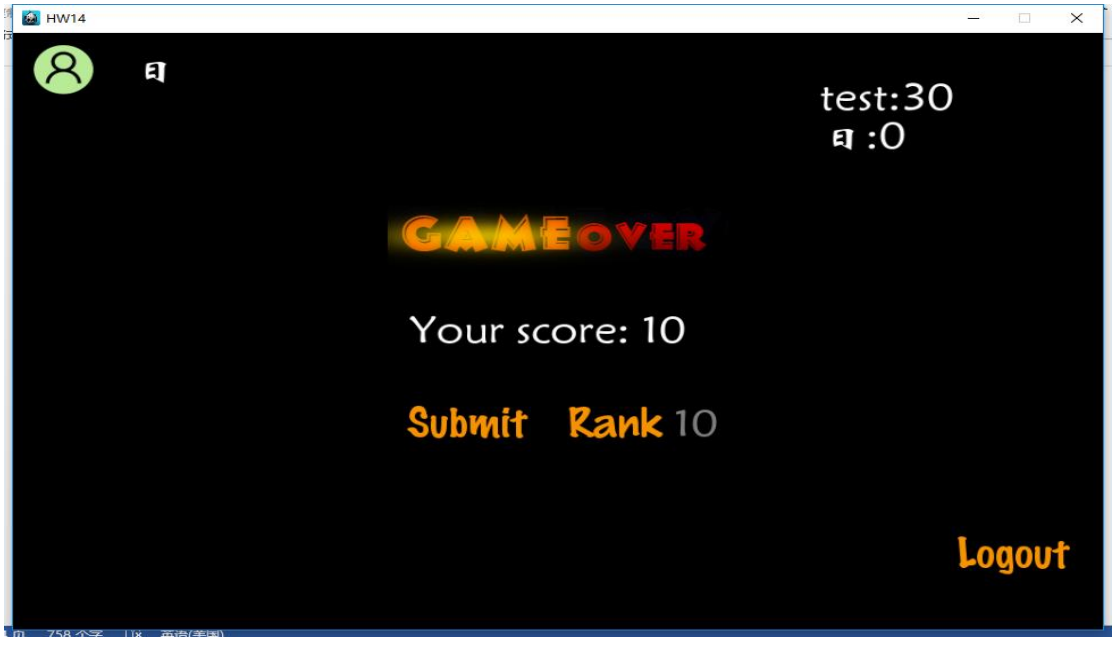


通过一关后显示当前分数，可选择继续下一关也可退出，退出不会清除用户信息，下次运行可自动登录并继续下一关。

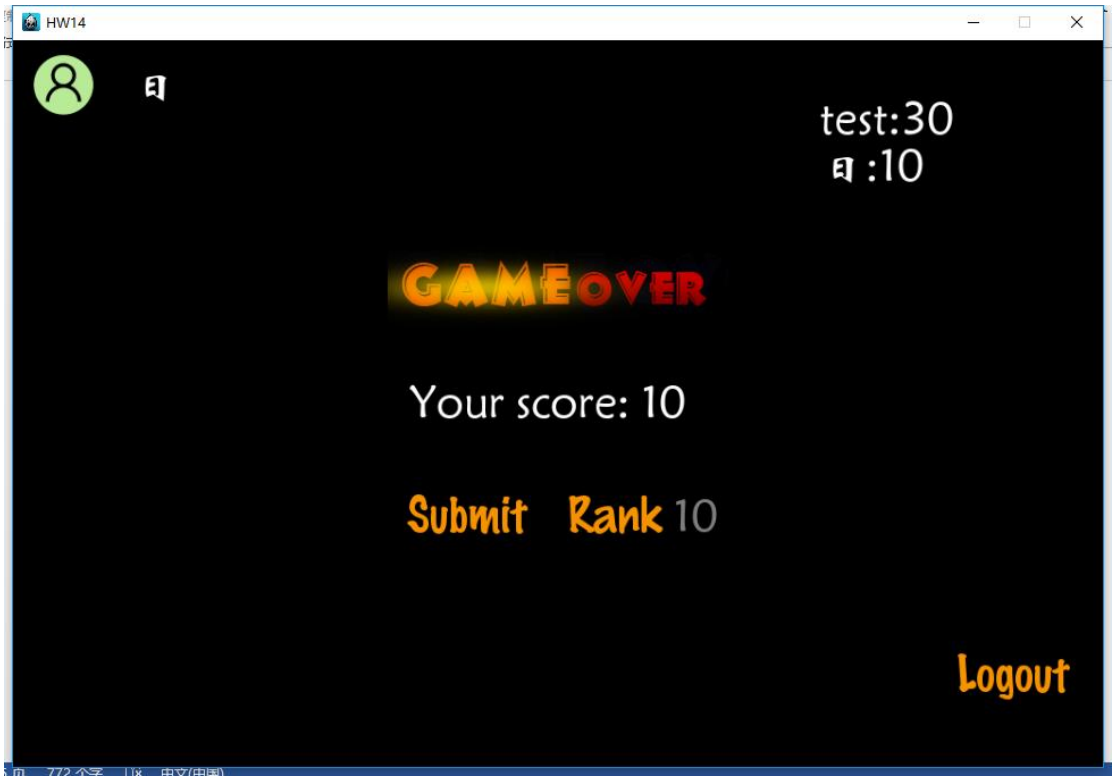
游戏失败界面，左上角显示当前用户，可提交分数，最多可查询前 10 名成绩，默认查询前 10 名。Logout 退出登录，退出后清除用户信息，回到登录界面



还没提交成绩



点击 submit 提交成绩后，排名自动更新



四．实验过程遇到的问题

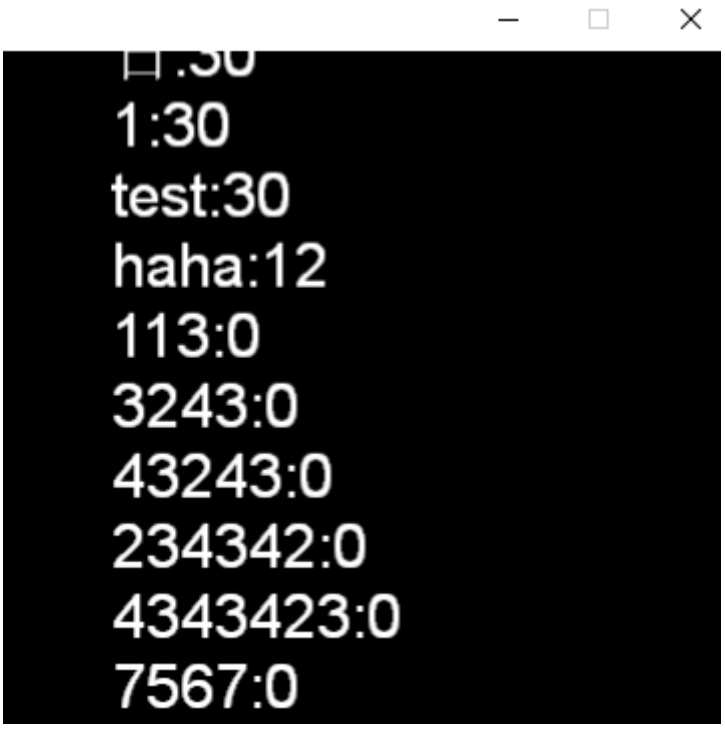
① 运行 java 服务器报出一大堆错，仔细查看错误信息后发现有一个错误信息是地址已被使用

```
java.net.BindException: Address already in use: bind
        at sun.nio.ch.Net.bind0(Native Method) ~[na:1.8.0_40]
        at sun.nio.ch.Net.bind(Net.java:437) ~[na:1.8.0_40]
        at sun.nio.ch.Net.bind(Net.java:429) ~[na:1.8.0_40]
        at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:223) ~[na:1.8.0_40]
        at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74) ~[na:1.8.0_40]
        at org.apache.tomcat.util.net.NioEndpoint.bind(NioEndpoint.java:340) ~[tomcat-embed-core-8.
        at org.apache.tomcat.util.net.AbstractEndpoint.start(AbstractEndpoint.java:765) ~[tomcat-em
```

仔细回想了一下才想起我的本地 8080 端口已经在 web 服务器中使用了，于是关闭了本地的 web 服务器，成功运行了 java 服务器。

② Submit 的时候一开始老是错误，说没有提供 gamesessionid，可是请求头中已经添加了，反复看课件后发现请求头中应该是”Cookie: GAMESESSIONID=xxxxx” ，原来缺少了“Cookie：”，这才想到请求头中的数据应该是键值对的形式，太粗心了。

③ 发现 demo 中的显示排名的信息用的是 TextField，排名信息可被编辑，故改为 Label 显示，排名信息就不可编辑。但是当排名的人数变多，显示的信息会超出屏幕边缘



观察后发现是因为锚点默认为(0.5,0.5),内容都是从中间向两边扩散，将锚点设为(0.5,1)成功解决了问题。

```
rank_field = Label::createWithTTF("", "fonts/STXINWEI.TTF", 40);
rank_field->setPosition(visibleWidth / 5 * 4, visibleHeight - 50);
rank_field->setContentSize(Size(visibleWidth / 4, visibleHeight / 5 * 3));
rank_field->setAnchorPoint(Vec2(0.5, 1));
this->addChild(rank_field);
```

五．思考与总结

① 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。

② cocos2d 和 UWP 进行网络编程有点麻烦，处理 json 也不是很方便，没有 python 和 JavaScript 等脚本语言简单方便。