## 单元测试报告

测试模块：登陆模块，条件组合覆盖

测试工具：unity

测试人员：陈栩

#### 1程序结构：

前端：

if (www.text != "1")

{

panel.SetActive(false);

penal3.SetActive(true);

}

else //如果账号已经存在

{

panel.SetActive(false);

penal2.SetActive(true);

GameObject btnObjx = GameObject.Find("queding2");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(Onone);

}

后台：

$sql="select \* from namepas where name = '$name'";

$result=mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

$return=0;

// 插入新数据

if($row['name']==null)

{

$sql="insert into namepas(name) values('$name')";

mysqli\_query($myData,$sql);

$return =1;

}

$sql="update namepas set my=1";

mysqli\_query($myData,$sql);

$sql="update namepas set my='$return' where name='$name'";

mysqli\_query($myData,$sql);

//关闭数据库

mysqli\_close($myData);

#### 测试结果分析

测试1：

账户已存在



分析：输入框在得到用户输入的用户名后就将文本传入数据库看是否存在，如果存在就直接进入角色选择界面，并且将该用户置为当前用户（my=0）

测试2:

账户不存在



分析：输入框在得到用户输入的用户名后就将文本传入数据库看是否存在，如果不存在就跳出弹框，点击确定后再次进入输入账号界面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 登陆 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试在数据库中的用户名和不再数据库中的用户名能否正常登陆 | | | |
| 测试方法 | | 在游戏界面和数据库端进行条件组合覆盖测试 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | 用户名：cx | cx是已经在数据库中存在的用户名，并且不在游戏进行时 | 正常登陆，进入角色选择界面,并且将该用户置为当前用户（my=1） | 正常登陆 ，进入角色选择界面,并且将该用户置为当前用户（my=1） | |
|  | 用户名：xx | xx是在数据库中不存在的用户名 | 跳出弹框，显示该用户名不存在 | 跳出弹框，显示该用户名不存在 | |
|  | 用户名：cx | cx是已经在数据库中存在的用户名，并且正在游戏中 | 跳出弹框，显示该用户已被登录 | 数据库中将该用户为已登陆用户（my=1） | |
| 测试结论 | |  |  |  |  |
| 状态 | | 正常 |  |  |  |

测试模块：注册模块，条件组合覆盖

测试工具：unity

测试人员：陈栩

#### 1程序结构：

前端：

if (www.text == "0")

{

panel.SetActive(false);

penal3.SetActive(true);

}

else if(www.text == "2")

{

panel.SetActive(false);

penal4.SetActive(true);

GameObject btnObjx = GameObject.Find("queding3");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(Onone2);

Inp\_Username.text = "";

}

else if (www.text=="1")//如果账号已经存在

{

panel.SetActive(false);

penal2.SetActive(true);

GameObject btnObjx = GameObject.Find("queding2");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(Onone);

Inp\_Username.text = "";

}

后台：

<?php

$name=$\_POST["name"];

$number=$\_POST["number"];

// 连接数据库

$myData=mysqli\_connect( "localhost" ,"root" ,"123","name\_password" );

if ( mysqli\_connect\_errno())

{

echo mysqli\_connect\_error();

return;

}

// 选择数据库

mysqli\_query($myData,"set names utf8") ;

mysqli\_select\_db( $myData ,"name\_password" );

// 查讯

$sql = "select \* from namepas where name = '$name'";

$result = mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

$return=0;

// 准备发送数据到Unity

if($row["my"]==1&&$number==1)

{

$return=2;

}

if($row["name"]==null&&$number==2)

{

$sql="insert into namepas(name) values('$name')";

mysqli\_query($myData,$sql);

$return =1;

}

else if($row["name"]==null&&$number==1)

{

$return =1;

}

else if($row["name"]!=null&&$number==1)

{

$sql="update namepas set my=1 where name='$name' ";

mysqli\_query($myData,$sql);

}

$myData=mysqli\_connect( "localhost" ,"root" ,"123","mydscoresdb" );

if ( mysqli\_connect\_errno())

{

echo mysqli\_connect\_error();

return;

}

// 校验用户名是否合法(防止SQL注入)

$UserID=mysqli\_real\_escape\_string($myData,$UserID);

// 选择数据库

mysqli\_query($myData,"set names utf8") ;

mysqli\_select\_db( $myData ,"mydscoresdb" );

$sql="select \* from hiscores where username = '$name'";

$result=mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

// 插入新数据

if($row["username"]==null)

{

$sql="insert into hiscores(username,score) values('$name',-1)";

mysqli\_query($myData,$sql);

}

// 关闭数据库

mysqli\_close($myData);

// 发送

echo $return;

?>

#### 测试结果分析

测试1：

注册账户已存在



分析：注册输入框在得到用户输入的用户名后就将文本传入数据库看是否存在，如果存在就跳出弹框，显示注册账号已存在

测试2:

注册账户不存在



分析：输入框在得到用户输入的用户名后就将文本传入数据库看是否存在，如果不存在就跳出弹框，显示注册成功。

测试3：

注册账号已被登录



分析：输入框在得到用户输入的用户名后就将文本传入数据库看my是否为0，如果为1就代表在游戏中，就跳出弹框显示账号已被登录，请输入其他账号。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 注册 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试已经存在数据库中的用户名和不在数据库中的用户名能否正常注册 | | | |
| 测试方法 | | 在游戏界面和数据库端进行条件组合覆盖测试 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | 用户名：cx | cx是已经在数据库中存在的用户名 | 不能正常注册，显示该用户名已存在 | 不能正常注册，显示该用户名已存在 | |
|  | 用户名：xx | xx是在数据库中不存在的用户名 | 正常注册，并在数据库中添加改账户 | 正常注册，并在数据库中添加改账户 | |
| 测试结论 | |  |  |  |  |
| 状态 | | 正常 |  |  |  |

测试模块：选择角色模块，条件组合覆盖

测试工具：unity

测试人员：陈栩

#### 1程序结构：

前端：

public void OnClicks()

{

SceneManager.LoadScene("peopleloading");

MapCreation.gameChoice = 1; //选择人类角色

}

public void OnClicks()

{

SceneManager.LoadScene("wolfloading");

MapCreation.gameChoice = 0; //选择狼人角色

}

#### 测试结果分析

测试1：

选择人类



分析：跳转人类的故事情节的场景

测试2:

选择狼人



分析：跳转狼人的故事情节的场景

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 选择角色 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试选择不同的角色是否会跳转该角色的故事情节的场景 | | | |
| 测试方法 | | 在游戏界面进行条件组合覆盖测试 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | 选择人类角色 | 人类角色的 MapCreation.gameChoice = 1 | 跳转人类故事情节场景 | 跳转人类故事情节场景 | |
|  | 选择狼人角色 | 狼人角色的 MapCreation.gameChoice = 0 | 跳转狼人故事情节场景 | 跳转狼人故事情节场景 | |
| 测试结论 | |  |  |  |  |
| 状态 | | 正常 |  |  |  |

测试模块：排行榜模块，条件组合覆盖

测试工具：unity

测试人员：石梦韬

#### 1程序结构：

后台：

<?php

$myData=mysqli\_connect( "localhost" ,"root" ,"123456","name\_password" );

if ( mysqli\_connect\_errno())

{

echo mysqli\_connect\_error();

return;

}

// 校验用户名是否合法(防止SQL注入)

$UserID=mysqli\_real\_escape\_string($myData,$UserID);

// 选择数据库

mysqli\_query($myData,"set names utf8") ;

mysqli\_select\_db( $myData ,"name\_password" );

// 插入新数据

$sql="select \* from namepas where my='0'";

$result=mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

// 读入用户名和分数

//用户

$UserID=$row['name'];

//分数

$hiscore=$\_POST["score"];

// 连接数据库

$myData=mysqli\_connect( "localhost" ,"root" ,"123456","mydscoresdb" );

if ( mysqli\_connect\_errno())

{

echo mysqli\_connect\_error();

return;

}

// 校验用户名是否合法(防止SQL注入)

$UserID=mysqli\_real\_escape\_string($myData,$UserID);

// 选择数据库

mysqli\_query($myData,"set names utf8") ;

mysqli\_select\_db( $myData ,"mydscoresdb" );

// 插入新数据

$sql="select \* from hiscores where name='$UserID'";

$result=mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

$time = time();

if($row['name']==null)

{

$sql="insert into hiscores(name,score) values('$UserID','$hiscore')";

$result=mysqli\_query($myData,$sql);

$sql="select \* from hiscores where name='$UserID'";

$result=mysqli\_query($myData,$sql);

$row = mysqli\_fetch\_array($result ,MYSQLI\_ASSOC);

}

else

{

if($row['score']<$hiscore)

{

$sql="update hiscores set score =$hiscore where name='$UserID'";

$result=mysqli\_query($myData,$sql);

}

else

$hiscore=$row['score'];

}

//关闭数据库

mysqli\_close($myData);

$id=$row['id'];

$time=strtotime($row['time']);

$arr =array();

$arr[$id]['id']=$row['id'];

$arr[$id]['name']=$UserID;;

$arr[$id]['score']=$hiscore;

$arr[$id]['time']=date("Y-m-d", $time);

echo json\_encode($arr);

?>

测试结果分析

测试1：

游戏当前分数并不是玩家游戏最高分

分析：玩家当前分数与数据库中玩家最高分进行比较，若游戏当前分数小于数据库中玩家分数，则不改变数据库中玩家分数

测试2:

游戏当前分数是玩家游戏最高分

分析：玩家当前分数与数据库中玩家最高分进行比较，若游戏当前分数大于数据库中玩家分数，则改变数据库中玩家分数，将其置为玩家当前分数，并将时间置为当前时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 排行榜 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试玩家获得不同分数时数据库进行的操作 | | | |
| 测试方法 | | 在后台进行条件组合覆盖测试 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | Userid=”cx’  Score=20 | 当前玩家的用户名为cx，分数为20 | 不改变当前玩家的分数 | 不改变当前玩家的分数 | |
|  | Userid=”cx’  Score=1000 | 当前玩家的用户名为cx，分数为1000 | 将当前玩家的分数改为1000 | 将当前玩家的分数改为1000 | |
| 测试结论 | |  |  |  |  |
| 状态 | | 正常 |  |  |  |