目 录

一. 系统需求分析

1. 系统描述

这是一款由本人原创的第一人称射击游戏，采用 MCVS 架构，类似网络游戏具有登陆验证系统，玩家登录后需要选择存档，根据存档选择的角色将会有特定角色的武器和刚体属性，然后每个存档将对应一个背包系统，玩家可以在游戏里捡取道具，使用道具后将使游戏玩法更多样化。

2. 数据存储需求

玩家注册用户将会有唯一的 id 标识，保存玩家账户信息

每个玩家将有三个存档

每个存档具有唯一的 id 标识，保存角色数据

每个背包对应一个存档

每个背包有三个库存格子

每个库存格子具有唯一的 id 标识，每个库存格子可以存放一种道具

目前有 3 种角色和 5 种道具

3. 系统常做的查询与更新

经常做的查询，或许对创建索引有影响的：

- 通过玩家 id 查看检索玩家信息
- 通过玩家 id 查看存档信息
- 通过存档 id 查看背包信息
- 通过库存 id 查看库存信息

关于更新

- 更新玩家信息
- 更新存档信息
- 更新背包信息
- 更新库存信息

4. 应用程序功能

客户端主要功能如下：

1) 登录用户界面：
    a. 登陆按钮 验证登陆信息 是否允许跳转至玩家存档界面
    b. 注册按钮 跳转用户注册界面
2) 用户注册界面：
    a. 用户注册：注册用户名不能重复 密码前后要一致
    b. 确定按钮 跳转玩家存档界面
    c. 返回按钮 返回登录界面
3) 玩家存档界面：
    a. 初始选定第一个存档，显示第一个存档的信息
    b. 玩家选择存档：3 个存档点中哪个就显示对应存档信息
    c. 开始游戏 若存档为空跳转至建立存档界面 否则进入游戏场景
4) 建立存档界面：
    a. 默认选择第一个角色，显示角色介绍
    b. 玩家选择角色：选中哪个角色就显示对应角色信息
    c. 创建按钮：若角色名不为空就创建角色跳转至玩家存档界面
5) 游戏场景：
    a. 初始化角色信息
    b. 玩家按键可以射击、使用物品、跳跃、行走、奔跑
    c. 显示角色 UI：生命 背包库存情况
    d. 玩家在场景中可以捡起物体

服务端主要功能如下：

1) 连接 MySql 数据库
2) 发送 SQL 指令
3) 插入玩家数据到数据库
4) 从数据库得到玩家数据
5) 更新玩家数据

二. 数据库概念设计

1. 确定实体和属性

分析网上图书销售系统的系统需求，将系统中设计的人、物进行抽象，得到了系统的实体如下：

1) 用户信息实体集。属性包括：用户编号、用户名、登录密码、注册日期
2) 存档实体集。属性包括：存档编号、角色编号、保存日期、存档名、存档等级
3) 角色实体集。属性包括：角色编号、角色名称
4) 背包库存实体集。属性包括：库存编号、物品编号、物品数目
5) 物品实体集。属性包括：物品编号、物品名称、物品类型、效果数目

2. E-R 图

系统 E-R 图如图 2-1 所示：

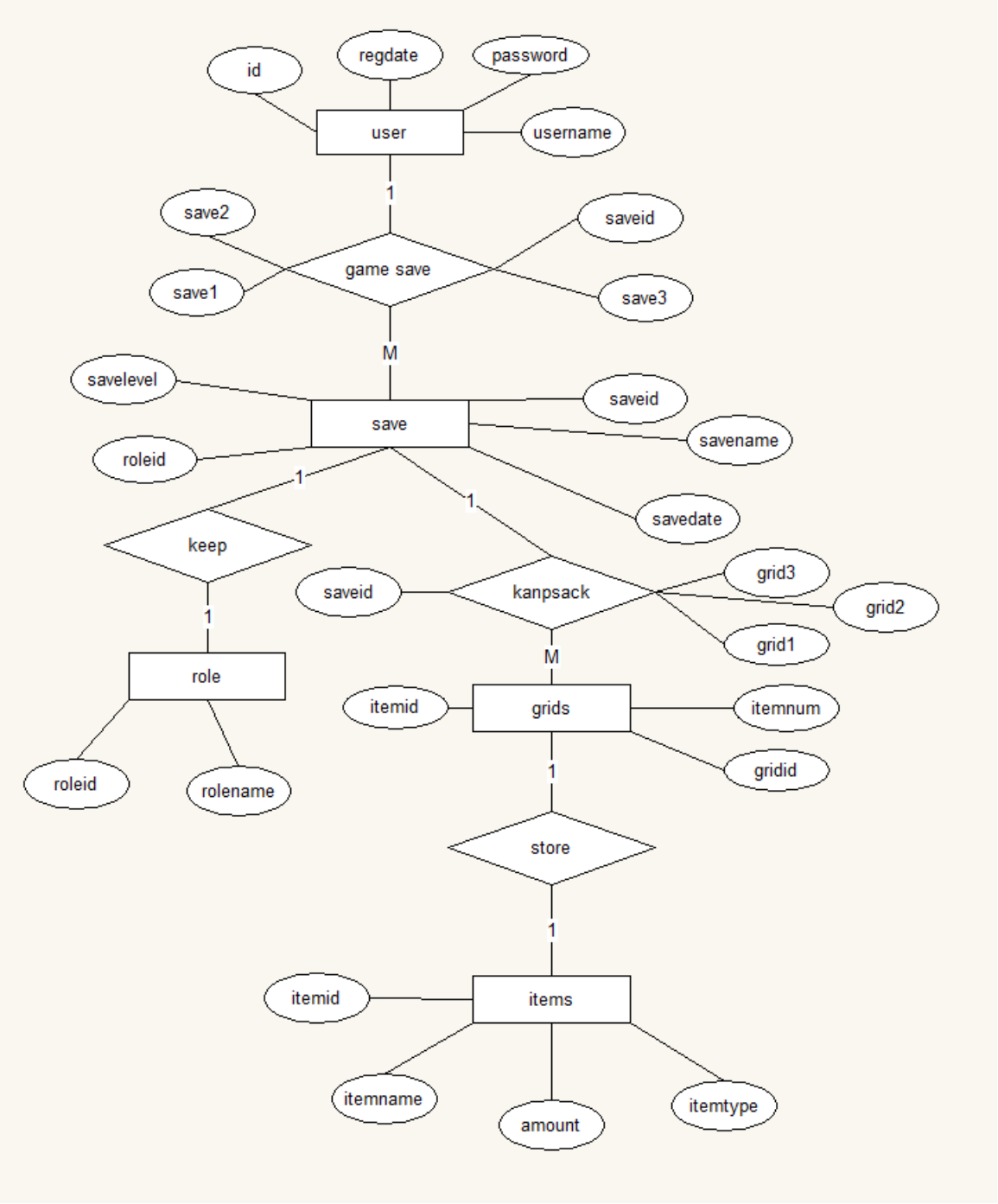
id
regdate
password
user
username
1
save2
saveid
game save
save1
save3
M
savelevel
saveid
save
savename
roleid
1
1
savedate
keep
saveid
kanpsack
grid3
grid2
grid1
1
M
role
itemid
grids
itemnum
roleid
rolename
gridid
1
store
1
itemid
items
itemname
amount
itemtype

图 2-1　E-R 图

三. 数据库逻辑结构设计

1. 关系模式设计

根据概念结构设计得到的 E-R 图和转换规则，得到如下关系模式（主键用下划线标出，外键予以说明）：

users(userid，username，password，regdate)

usersaves(userid，save1，save2，save3)　Foreign Key:save1 save2 save3

saves(saveid，roleid，savedate，savename，savelevel)　Foreign Key: roleid

roles(roleid，rolename)

knapsacks(saveid，grid1，grid2，grid3)　Foreign Key: grid1 grid2 grid3

grids(gridid，itemid，itemnum)　Foreign Key: itemid

items(itemid，itemname，itemtype，amount)

2. 基本表设计

users

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
userid	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
username	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
password	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
regdate	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

usersaves

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
userid	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
save1	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
save2	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
save3	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Saves

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
saveid	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
roleid	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
savedate	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
savename	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
savelevel	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

roles

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
roleid	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
rolename	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

knapsacks

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
saveid	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
grid1	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
grid2	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
grid3	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

grids

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
gridid	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
itemid	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
itemnum	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

items

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
itemid	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
itemname	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
itemtype	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
amount	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

四. 应用程序设计

1. 开发及运行环境介绍

MySql、Unity3D、C#.Net3.5、Mono2.0

2. 主要功能设计

只解释类和部分函数的作用，详情看具体代码，变量名和类名很直观的将不解释

动画类：

Splash.cs	RotateObj.cs	ComeOut.cs	StrenchObj.cs
开场 Logo 用	自动旋转物体	开场 Logo 用	自动伸展物体

界面交互类：

1.PageStart.cs

```
public class PageStart : MonoBehaviour
{
    public Text usernameText;
    public InputField passwordText;
    public Text tipText;

    public void RegButtonClick()   //跳转至 PageRegUser

    public void LoginButtonClick() //登录按钮

}
```

2.PageRegUser .cs

```
public class PageRegUser : MonoBehaviour
{
    public Text userNameText;
    public InputField passwordText;
    public InputField repetWordText;
    public Text tipText;
```

```
    public void RegistButtonClick()  //注册按钮


    public void ReturnButtonClick()   //返回 PageStart

}
```

3.PageUserSaves.cs

```
public class PageUserSaves : MonoBehaviour
{
    public static PageUserSaves instance;

    public SaveGrid[] saveGrids;
    public Text[] infoTexts;
    public SaveGrid currentGrid;
    public string[] roleInfos;

    public Slider loadingSlider;

    public void InitialPage()


    public void UpdateGridData(Save save) //更新 saveGridUI


    public void SetInfoTexts()


    public void ConfirmButtonClick() //跳转开始游戏 OR 创建角色


    public IEnumerator StartGame()
}
```

4.PageCreateRole .cs

```
public class PageCreateRole : MonoBehaviour
{
    public static PageCreateRole instance;
    public Text roleName;
    public string[] roleDescriptions;

    public int saveSign;
    public RoleGrid[] roleGrids;
```

```
    public RoleGrid currentRoleGrid;
    public Text roleDesText;

    public void InitialRoleGrids()

    public void SetRoleDesText()//设置角色描述信息

    public void CreateRoleButtonClick()
 }
```

数据库处理类：

MysqlController.cs

```
public class MysqlController : MonoBehaviour
{
    public static MysqlController instance;
    public string connectStr;

    public void LoadConnectConfig()//从配置文本读入连接设置

    void Awake()
    {
        instance = this;
        DontDestroyOnLoad(gameObject);
    }

    public User VerifyUser(string username, string passwo
rd)   //验证是否用户名存在以及密码是否正确

    public Save[] GetUserSaves(int userId)
```

```
    public Save GetSave(string saveName)

    public Save GetSave(int saveid)


    public Grid[] GetKnapsack(int saveId)

    public Grid GetGrid(int gridId)


    public void ExecuteScalar() //查看所有用户信息


    public void InsertSave(int roleId,string saveName)


    public int InsertGrid(int itemId,int gridSign)


    public void UpdateGrids()


    public void UpdateUserSaves(User user,int saveSign)
}
```

数据结构类： DataStructure.CS

```
[System.Serializable]
public class User
{
    public string username;
    public int userid;
```

```
    public Save[] saves;

    public User(int _userid,string _username)
    {
        username = _username;
        userid = _userid;
    }
}

[System.Serializable]
public class Save
{
    public int saveId;
    public string saveName;
    public int saveLevel;
    public string saveDate;
    public int roleId;

    public Save(int _saveId,int _roleId,string _saveDate,
string _saveName,int _saveLevel)
    {
        saveId = _saveId;
        saveName = _saveName;
        saveLevel = _saveLevel;
        saveDate = _saveDate;
        roleId = _roleId;
    }
}

[System.Serializable]
public class Role
{
    public int startHealth;
    public int jumpForce;
    public int forwardSpeed;
}


[System.Serializable]
public class Knapsack
{
    public int saveId;
    public Grid[] grids;
```

```
    public Knapsack(int _saveId)
    {
        saveId = _saveId;
    }
}

[System.Serializable]
public class Grid
{
    public int gridId;
    public int itemId;
    public int itemNum;

    public Grid(int _gridId,int _itemId,int _itemNum)
    {
        gridId = _gridId;
        itemId = _itemId;
        itemNum = _itemNum;
    }
}

public enum ItemType{Health,Speed,Jump,Link,Transfer};

[System.Serializable]
public class Item
{
    public ItemType itemType;
    public int itemId;
    public string itemName;
    public int amount;
}
```

数据库数据转数据类（面向关系转换面向对象处理）

DataController.CS

```
public class DataController : MonoBehaviour
{
    public User currentUser;
    public Save currentSave;
    public Knapsack currentKnapsack;

    public static DataController instance;
```

```
    void Awake()
    {
        instance = this;
        DontDestroyOnLoad(gameObject);
    }

    public void GetCurrentUser(User user)

    public void GetCurrentKnapsack()

}
```

游戏场景游戏核心数据控制：

GameController .CS

```
public class GameController : MonoBehaviour
{
    public static GameController instance;
    public Text healthText;
    public Sprite[] itemSprites;
    public ItemGrid[] itemGrids;
    public GameObject[] guns;
    public Role[] roles;
    int roleId;
    public RigidbodyFirstPersonController roleController;

    public Item[] items;

    public int itemCount;

    public GameObject exitGameMenu;
    public Transform transfer;

    public void UseItem(int itemId)

    public void UpdateItemGrid(int itemId)

    void UseItemUpdate(int gridIndex)

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Q))
        {
            UseItemUpdate(0);
        }
```

```
        if (Input.GetKeyDown(KeyCode.E))
        {
            UseItemUpdate(1);
        }

        if (Input.GetKeyDown(KeyCode.R))
        {
            UseItemUpdate(2);
        }

        if (Input.GetKeyDown(KeyCode.Escape))
        {
            exitGameMenu.SetActive(!exitGameMenu.activeSe
lf);
        }

        if (exitGameMenu.activeSelf)
        {
            if (Input.GetKeyDown(KeyCode.Return))
            {
                OnGameExit();
                print("UpdateGrid OK");

                Application.Quit();
            }
        }
    }
    public void OnGameExit()

    public int GetRoleStartHealth(Save save)

    void InitialGame()

    public void SetRoleForwardSpeed(float forwardSpeed)

    void SetRoleJumpForce(float jumpForce)

    void SetHealthText(int healthNum)
}
```

游戏枪支射击逻辑：

GunController.CS

```
public class GunController : MonoBehaviour
{

    public Transform[] projectileSpawns;    //枪口

    public GameObject bulletPrefab;

    public float timeBetweenAttack=.5f;
    float nextAttackTime;


    Vector3 startPosition;
    public float gunMoveLength=1f;
    MuzzleFlash[] muzzleFlashs;

    public float waitTime = 2f;
    public AnimationCurve animationCurve;

    public bool canLink;

    void DoShoot()      //射击


    void InstantiateBullet() //发射子弹


    IEnumerator MoveGun()  //枪支后坐力
}
```

MuzzleFlash .CS //枪支显示开火动画

```
public class MuzzleFlash : MonoBehaviour
{
    public GameObject flashHolder;
    public Sprite[] flashSprites;
    public SpriteRenderer[] spriteRenderers;

    public float flashTime;

    public void Activate()    //火花闪烁


    void Deactivate()      //火花隐藏
}
```

3. 主要界面

登录界面如图 5-1 所示。

图 4-1 登陆界面

图 4-2 注册界面

图 4-3 选择存档界面

图 4-4 创建角色界面

图 4-5 游戏中界面

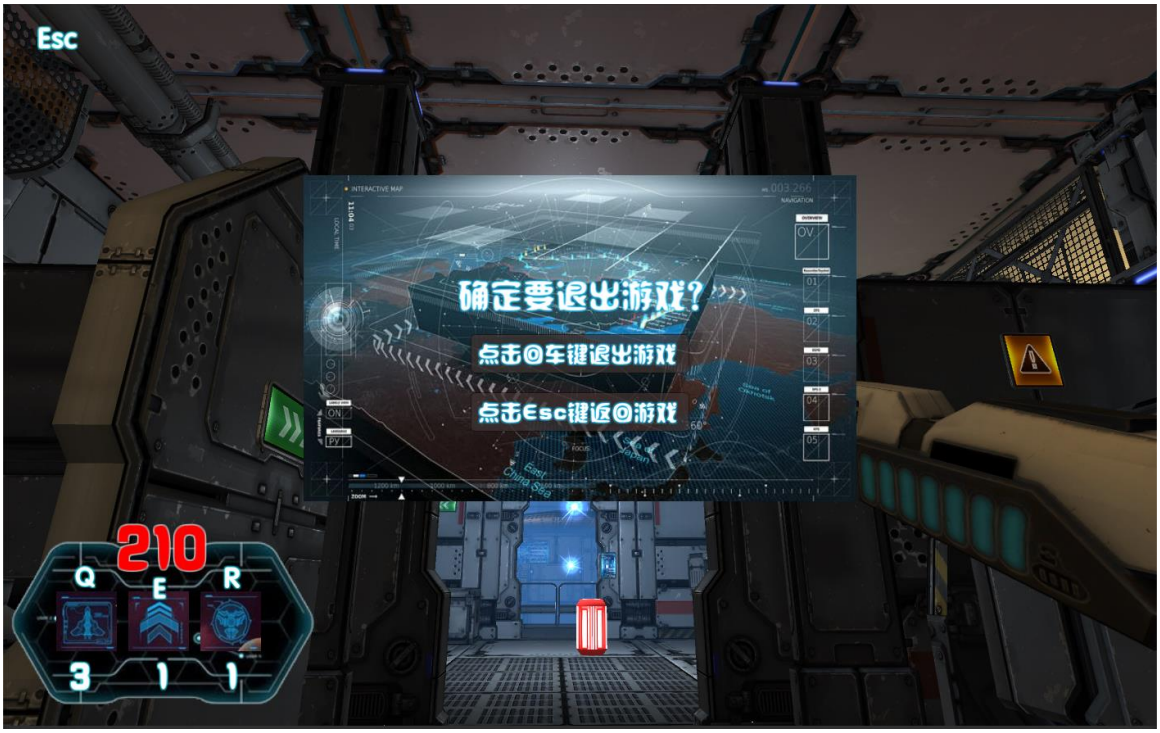

图 4-6 退出游戏界面

五. 心得体会

服务器端会用到数据库,但现在都是在搞面向对象的数据库~比如使用 Nhibernate 直接不用 SQL 指令就可以将面向关系的数据库 Mysql 或 SqlServer 转换为面向对象数据库处理，这样对程序员当作类来操作将会很方便。而且客户端数据基本是通过 json 存的（因为我在游戏公司的任务就是写界面、用户交互和配置数据表）所以就纯当练手了。