

学 号:	0122008910419 0122009360912 0122008890201 0122007780122 0122011350612
------	---

武汉理工大学

软件工程实践（二）

学 院	计算机与人工智能学院
专 业	软件工程
班 级	软件 2001
小 组	momo
成 员	肖峰、张忠瑾、戚家辉、李涵、姜明昆
指导教师	唐祖锴

2023 年 6 月 24 日

目录

1	任务概述	1
2	程序设计与开发.....	1
3	开发版本计划与任务指派	3
4	开发分支模型与代码合并	4
5	开发规范与测试.....	5
6	集成与版本发布.....	7
7	实施情况	8
8	问题记录与分析.....	19
9	任务总结	19
10	参考文献.....	20

1 任务概述

1.1 任务目的

- 巩固强化软件编程规范
- 提高面向对象软件建模与抽象能力
- 培养小组协同开发能力
- 掌握基于 Maven 的软件项目管理机制
- 掌握基于 Github 的小组协同开发工具和平台
- 了解 DevOps 软件开发流程

1.2 任务内容

- 创建开发小组
- 确定开发目标
- 分配开发任务
- 分支开发与继承
- 自动规范检查与测试
- 自动打包与发布

2 程序设计与开发

2.1 技术和工具

Git 版本控制工具

Github 小组协作平台

Maven 项目工程管理

FXGL 游戏开发框架

Junit 单元与集成测试

Mangodb 数据库

MarkDown 文档编写工具

DevOps 软件开发过程

2.2 项目框架

本次关于"World of Zuul"小组任务的项目框架包括使用 Maven 作为项目管理工具和 FXGL 作为游戏引擎。以下是对这两个框架的简要介绍：

1. Maven:

Maven 是一个开源的项目管理和构建工具，用于管理 Java 项目的构建、依赖管理和项目生命周期。它使用基于 XML 的配置文件（pom.xml）来定义项目的结构、依赖关系和构建步骤。

在"World of Zuul"项目中，Maven 用于管理项目的依赖项、编译源代码、运行测试和构建可执行的游戏包。通过定义和管理依赖关系，Maven 能够自动下载所需的库和工具，简化了项目的配置和构建过程。此外，Maven 还支持持续集成和部署，可以轻松地将项目打包成可分发的格式。

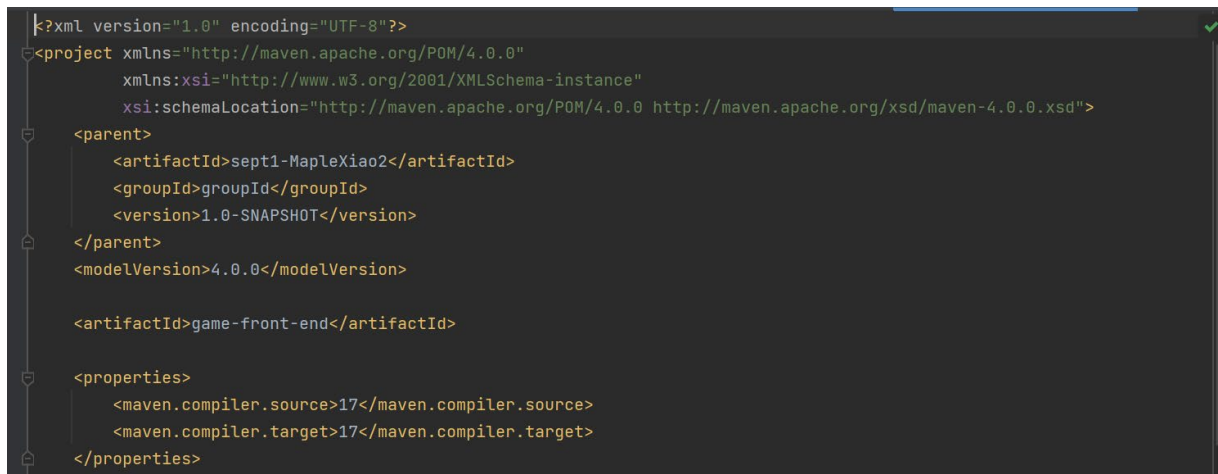
2. FXGL:

FXGL 是一个基于 JavaFX 的游戏开发框架，提供了丰富的功能和工具，使开发者能够快速构建 2D 游戏。它提供了场景管理、实体组件系统、碰撞检测、输入处理、动画和 UI 组件等功能。

在"World of Zuul"项目中，FXGL 用作游戏引擎，负责处理游戏的渲染、用户交互和物理模拟等方面。它提供了易于使用的 API 和一套强大的工具，使开发者能够专注于游戏逻辑和设计，而无需过多关注底层的实现细节。

FXGL 还支持基于 Tiled Map Editor 的地图编辑器，可用于创建游戏场景和关卡。此外，FXGL 还提供了网络功能和多人游戏支持，适用于需要实现多人交互的游戏项目。

综上所述，"World of Zuul"小组任务中的项目框架使用了 Maven 作为项目管理工具，用于管理依赖和构建过程；使用 FXGL 作为游戏引擎，提供丰富的功能和工具，使开发者能够快速构建 2D 游戏。这些框架的使用可以提高开发效率、简化配置，并为团队成员提供一致的开发环境。



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <parent>
    <artifactId>sept1-MapleXiao2</artifactId>
    <groupId>groupId</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>

  <artifactId>game-front-end</artifactId>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>
</project>
```

2.3 开发计划

本次任务的开发计划如下：

阶段一：确定游戏模块和功能，任务分工，制定开发计划

1. 确定游戏的各个模块和主要功能，包括场景管理、角色控制、物体交互、关卡设计等方面。
2. 进行任务分工，根据团队成员的技能和兴趣，将各个模块分配给相应的开发人员。
3. 制定开发计划，确定每个阶段的目标和行程安排，包括开发、测试和代码审查等。

阶段二：并行开发，前后端编码和接口功能实现

1. 小组成员在各自的分支上独立进行开发，开始编写前端代码和后端接口功能。
2. 前端开发人员使用 FXGL 和相关工具，实现游戏场景、用户界面和交互功能。
3. 后端开发人员使用 IOGame 和 MongoDB，编写游戏的后端逻辑和数据存储功能。
4. 进行 action 的配置编写，确保游戏的行为和事件能够正确触发和响应。

阶段三：代码审查、单元测试和分支合并

1. 完成各个模块的开发后，进行代码审查和单元测试，确保代码质量和功能正确性。
2. 每个开发人员将自己的分支提交到 dev 总分支，并进行合并操作。

3. 进行代码合并时，解决可能出现的冲突，并确保各个模块之间的协调和一致性。

阶段四：发布可执行版本软件

1. 在代码审查和测试无错误并且无冲突的情况下，进行最终的发布准备。
2. 打包和构建可执行版本的软件，包括前端资源和后端服务。
3. 进行终端测试，确保游戏在不同环境中能够正确运行和展示预期功能。
4. 发布可执行版本软件，可以是可执行文件、安装包或 Web 应用等形式，供用户使用。

3 开发版本计划与任务指派

3.1 开发版本

基于任务要求，以下是预期的开发版本计划：

版本 1.0（MVP）：

1. 完成基本的游戏框架搭建，包括场景管理、角色控制和基本物体交互功能。
2. 实现游戏的主要功能，例如玩家移动、物体碰撞、基本游戏规则等。
3. 创建至少两个关卡，并确保玩家能够在这些关卡中进行游戏。
4. 实现基本的用户界面，包括开始菜单、游戏设置和得分显示等。
5. 进行基本的单元测试和功能测试，确保游戏的基本功能正常运行。

版本 2.0（功能增强）：

1. 增加更多的关卡和游戏内容，提供更丰富的游戏体验。
2. 添加更多的角色和物体交互功能，例如道具收集、敌人 AI 等。
3. 增加游戏中的音效和背景音乐，提升游戏的声音效果。
4. 完善用户界面，包括游戏设置的扩展、关卡选择和存档等功能。
5. 进行进一步的单元测试和集成测试，确保新功能的正确性和稳定性。

版本 3.0（优化和扩展）：

1. 对游戏进行性能优化，提高游戏的运行效率和响应速度。
2. 增加更多的游戏特性，例如成就系统、排行榜等社交功能。
3. 实现多人游戏模式，允许玩家进行在线对战或合作。
4. 引入更复杂的关卡设计，增加谜题和挑战性元素。
5. 进行全面的测试，包括压力测试和兼容性测试，确保游戏的稳定性和可玩性。

版本 4.0（最终发布）：

1. 进行代码审查和修复潜在的问题，确保代码质量和可维护性。
2. 进行最终的用户界面优化，确保游戏的可用性和易用性。
3. 执行最终的全面测试，包括用户体验测试和多平台测试。
4. 修复剩余的问题和漏洞，并进行最后的性能调优和优化。
5. 准备发布版本，包括生成可执行文件或安装包，并发布到目标平台或应用商店。

注意，这只是一个大致的版本计划，具体的版本迭代和时间安排应根据实际开发进展和团队资源来进行调整。

3.2 任务指派

小组分工如下：

- 肖 峰（组长）：统筹规划、任务分配、系统架构设计
- 张忠瑾：视觉和交互评审、页面构建和开发、前后端联调
- 戚家辉：数据库设计、业务逻辑实现、前后端联调
- 李 涵：代码测试、后台管理及维护、报告撰写
- 姜明昆：代码规范化、ppt 制作、项目跟进

<input type="checkbox"/>	7 Open	0 Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	讨论打包和部署方式	#7 opened yesterday by MapleXiao2						
<input type="checkbox"/>	规范检查和测试安排	#6 opened yesterday by MapleXiao2						
<input type="checkbox"/>	技术实现方案讨论	#5 opened yesterday by MapleXiao2						3
<input type="checkbox"/>	项目分支和具体任务	#4 opened yesterday by MapleXiao2						2
<input type="checkbox"/>	分配各个成员的工作任务	#3 opened yesterday by MapleXiao2						3
<input type="checkbox"/>	讨论功能扩充需求点	#2 opened yesterday by MapleXiao2						2
<input type="checkbox"/>	项目任务大纲	#1 opened yesterday by MapleXiao2						

4 开发分支模型与代码合并

4.1 分支模型

1. master 分支：

- master 分支作为主分支，用于存放稳定的、可发布的代码版本。
- 只有经过充分测试和审查的代码才能合并到 master 分支。
- 直接从其他分支合并或通过 Pull Request 合并到 master 分支。

2. server 分支：

- server 分支用于后端服务器的开发和维护。
- 后端开发人员在该分支上独立进行后端逻辑和接口功能的编写。
- 后端开发完成并经过测试后，可以将代码合并到 master 分支。

3. client 分支：

- client 分支用于前端客户端的开发和维护。
- 前端开发人员在该分支上独立进行游戏界面、用户交互和前端逻辑的编写。
- 前端开发完成并经过测试后，可以将代码合并到 master 分支。

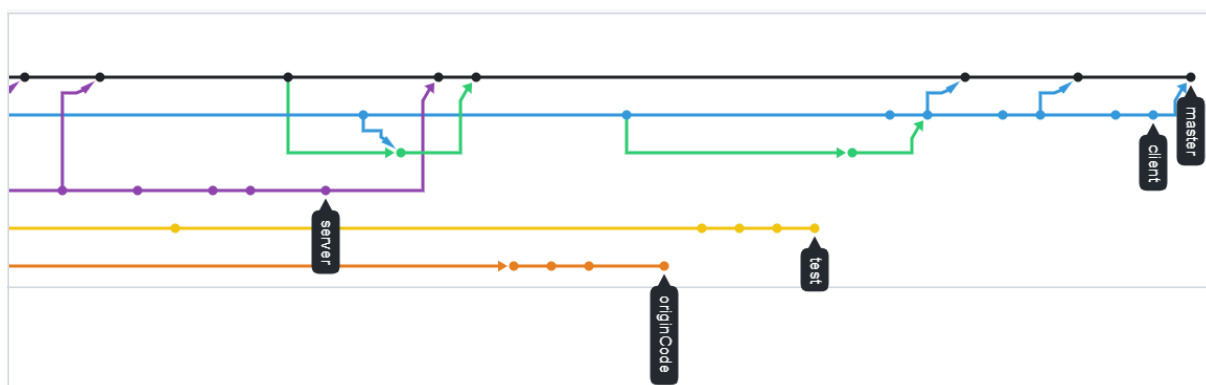
4. test 分支：

- test 分支用于测试团队进行功能测试、集成测试和其他测试相关工作。
- 测试团队在该分支上独立进行测试脚本编写、测试用例执行和错误修复。
- 完成测试并确认无错误后，可以将代码合并到 master 分支。

All branches					
master	Updated 40 minutes ago by GrayFluex		Default		
client	Updated 14 minutes ago by GrayFluex	20 1		New pull request	
test	Updated 52 minutes ago by fshoes	18 11		New pull request	
originCode	Updated 1 hour ago by 7WaveZ	26 4		New pull request	
server	Updated 2 hours ago by GrayFluex	19 0		New pull request	

4.2 合并策略

- 开发人员各自在分支上独立开发，确保代码功能正确并通过单元测试。
- 当功能开发完成后，开发人员将自己的分支与 **master** 分支进行合并前，先合并最新的 **master** 分支到自己的分支，以保持代码同步。
- 对于 **server**、**client** 和 **test** 分支，可以通过 Pull Request 方式进行合并，由相关负责人进行代码审查和合并操作。
- 在进行合并之前，必须确保代码没有冲突，以及通过必要的测试，包括单元测试和功能测试等。
- 确认合并后的代码在 **master** 分支上运行稳定且没有明显问题后，即可发布或准备发布版本。



5 开发规范与测试

5.1 开发规范

开发规范的重要性无法被低估。开发规范是确保代码质量、提高可维护性和促进团队协作的关键因素。通过一致的编码风格、命名规范和代码结构，开发规范使团队成员能够更轻松地理解和维护代码，减少潜在的错误和 bug。规范化的开发流程和工具使用提高了开发效率，并为团队协作提供了框架。同时，开发规范还对跨团队合作起到重要作用，确保不同团队之间的代码一致性和整体质量。遵循开发规范是确保项目成功和团队协作高效的关键要素，它们提供了一个统一的标准，使开发人员能够以一致的方式构建高质量的软件。以下，是此次任务中，我们小组遵循的开发规范：

5.1.1 命名规范

- 使用有意义且符合约定的命名方式，使代码易于理解和维护。
- 使用驼峰式命名法（**camelCase**）或下划线命名法（**snake_case**）来命名变量、函数、类、文件等。

- 避免使用过于简单或含糊不清的命名，尽量使用具有描述性的命名。

5.1.2 注释规范

- 在代码中使用清晰、有用的注释，解释代码的意图、实现细节和注意事项等。
- 使用自然语言编写注释，确保注释易于理解，并与代码保持一致。
- 注释应该简洁明了，不过多注释显而易见的代码，但要注明复杂或不明显的部分。

5.1.3 编码规范

- 遵循统一的编码风格，使代码具有一致性和可读性。
- 使用适当的缩进和空格，使代码结构清晰。
- 根据编程语言的惯例和最佳实践，选择合适的代码结构和命名风格。
- 避免使用魔术数字和硬编码的常量，使用有意义的变量和常量名称。
- 尽量保持代码简洁、可维护和可扩展，避免冗余代码和过于复杂的逻辑。

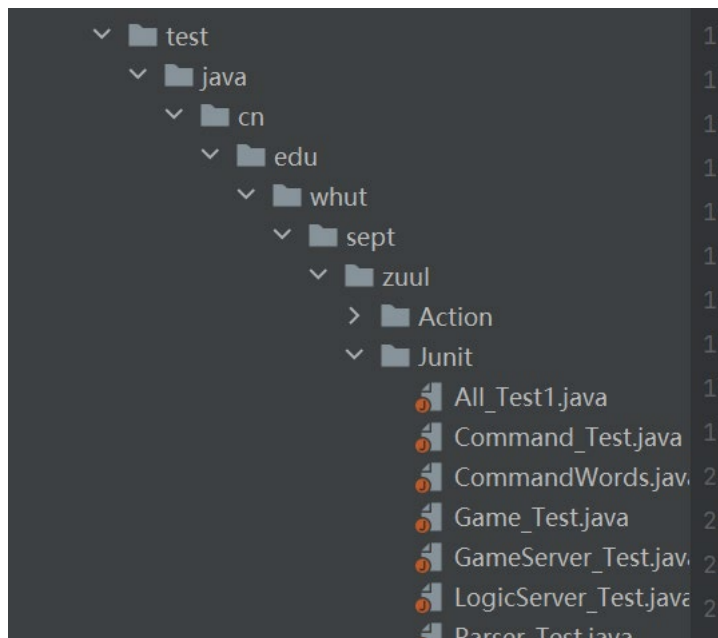
5.1.4 分支管理规范

- 使用版本控制系统（如 Git）进行代码管理和分支管理。
- 主分支通常包括 **master**（主要发布版本）和 **dev**（开发版本）。
- 每个开发人员在自己的功能开发上创建独立的分支，分支命名可基于特性或任务。
- 在进行分支合并之前，确保通过合适的测试，解决代码冲突，并进行代码审查。

5.2 测试计划

5.2.1 测试工具说明

使用 Junit4 进行单元测试。所有测试文件在根目录的 **test** 文件夹下的 **Junit** 目录。



5.2.2 单元测试设计原则

单元测试设计原则是为了确保单元测试的有效性和可靠性而制定的一系列指导原则和最佳实践。这些原则包括独立性、单一职责、可重复性、全面性、及早测试、可读性和高效性。独立性原则确保每个单元测试都能独立执行，不受其他测试或外部环境的影响。单一职责原则确保每个单元测试专

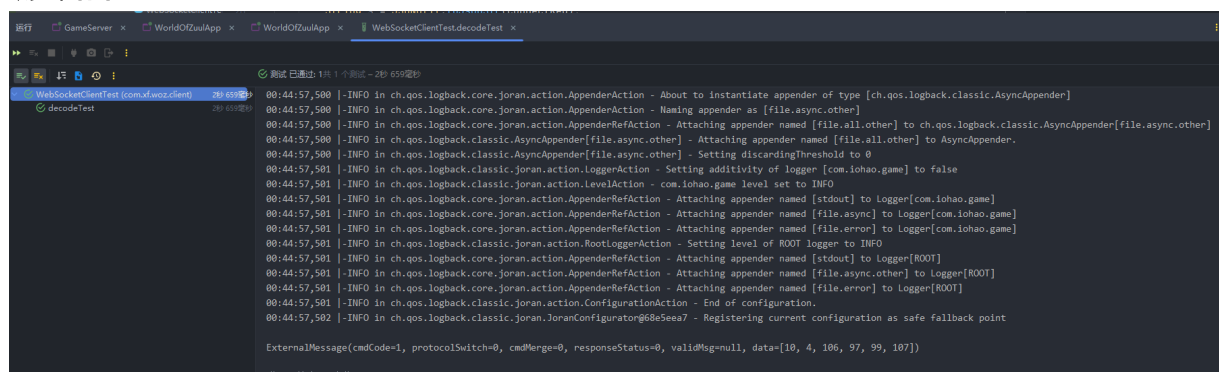
注于测试特定功能或行为。可重复性原则强调单元测试应该是可重复的，不依赖于外部资源或状态。全面性原则要求覆盖被测单元的各种情况和路径。及早测试原则鼓励在代码早期进行单元测试，以尽早发现和解决问题。可读性原则强调测试代码的可读性，方便他人理解和维护。高效性原则要求测试代码高效运行，避免冗长和不必要的重复。遵循这些原则可以提高单元测试的质量和效率，确保代码的正确性和稳定性。

5.2.3 单元测试对象和测试用例编写

测试用例编写

```
public class WebSocketClientTest {  
    @Test  
    public void decodeTest(){  
        int cmd = 0;  
        int subCmd = 0;  
        ConnectVerify connectReq = new ConnectVerify();  
  
        connectReq.setMsg("jack");  
        String s = JSONUtil.toJsonStr(connectReq);  
        ExternalMessage externalMessage = ExternalKit  
            // 路由、子路由、业务数据  
            .createExternalMessage(cmd, subCmd, connectReq);  
        System.out.println(externalMessage);  
  
        // System.out.println("CommentWebSocketClient.onOpen");  
        // 游戏框架内置的协议， 与游戏前端相互通信的协议  
    }  
}
```

测试结果



```
运行 GameServer x WorldOfZuulApp x WorldOfZuulApp x WebSocketClientTest.decodeTest x  
测试 已通过: 1秒 1个测试 - 2秒 659毫秒  
WebSocketClientTest [com.id.meow.client] 20/699B  
decodeTest 38/403B  
00:44:57,500 | -INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logback.classic.AsyncAppender]  
00:44:57,500 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Naming appender as [file.async.other]  
00:44:57,500 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [file.all.other] to ch.qos.logback.classic.AsyncAppender[file.async.other]  
00:44:57,500 | -INFO in ch.qos.logback.classic.AsyncAppender[file.async.other] - Attaching appender named [file.all.other] to AsyncAppender.  
00:44:57,500 | -INFO in ch.qos.logback.classic.AsyncAppender[file.async.other] - Setting discardingThreshold to 0  
00:44:57,501 | -INFO in ch.qos.logback.classic.joran.action.LoggerAction - Setting additivity of logger [com.iohao.game] to false  
00:44:57,501 | -INFO in ch.qos.logback.classic.joran.action.LevelAction - com.iohao.game level set to INFO  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [stdout] to Logger[com.iohao.game]  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [file.async] to Logger[com.iohao.game]  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [file.error] to Logger[com.iohao.game]  
00:44:57,501 | -INFO in ch.qos.logback.classic.joran.action.RootLoggerAction - Setting level of ROOT logger to INFO  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [stdout] to Logger[ROOT]  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [file.async.other] to Logger[ROOT]  
00:44:57,501 | -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [file.error] to Logger[ROOT]  
00:44:57,501 | -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - End of configuration.  
00:44:57,502 | -INFO in ch.qos.logback.classic.joran.JoranConfigurator@08e5ea7 - Registering current configuration as safe fallback point  
ExternalMessage(cmdCode=1, protocolSwitch=0, cmdMerge=0, responseStatus=0, validMsg=null, data=[10, 4, 106, 97, 99, 107])  
测试已结束 退出代码0
```

6 集成与版本发布

集成和版本发布是确保项目成功的关键步骤。通过合理的集成流程和版本管理，可以确保各个

模块的代码正确集成，并生成稳定的发布版本。持续改进和反馈的循环将有助于项目的长期发展和优化。

6.1 集成开发分支

- 将各个开发分支（如 `server`、`client` 等）的代码合并到主分支（`master` 分支）中。
- 执行代码合并和冲突解决，确保各个模块的代码能够正确地集成到主分支中。

6.2 构建和测试

- 在集成完成后，进行整体的构建和编译，生成可执行文件或软件包。
- 运行全面的单元测试、集成测试和系统测试，确保代码的质量和功能的稳定性。

6.3 代码审查

- 进行代码审查，团队成员相互检查代码，确保代码质量和一致性。
- 检查代码规范、注释、命名等方面，发现潜在的问题并提出改进建议。

6.4 版本号管理

- 根据项目的版本管理规范，为发布的版本分配一个唯一的版本号。
- 可以使用语义化版本控制（`Semantic Versioning`）等方式进行版本号命名。

6.5 发布版本

- 创建一个可执行的、稳定的版本软件或游戏包。
- 将生成的可执行文件或软件包进行打包、压缩等操作。

6.6 文档更新

- 更新项目文档（`readme.md`）。

[Releases](#) / pre

ZUUL-v1.0 Latest

MapleXiao2 released this 23 minutes ago pre b9a124d ✓

By Momo

▼ Assets 2

Source code (zip)

2 hours ago

Source code (tar.gz)

2 hours ago

7 实施情况

7.1 扩展功能举例

游戏初始化

```
@Override
protected void initGame() {
    FXGL.getGameWorld().addEntityFactory(new PlayerFactory());
    getGameWorld().addEntity(Objects.requireNonNull(RoomFactory.createEntity(RoomEntityType.OUTSIDE)));
    playerEntity = spawn(entityName: "player");
    playerEntity.addComponent(new PlayerView());
    // spawn("outside");

    FXGL.runOnce() -> {
        List<String> lines = getAssetLoader().loadText(name: "login.txt");
        Cutscene cutscene = new Cutscene(lines);
        getCutsceneService().startCutscene(cutscene, () -> {
            getDialogService().showInputDialog(s: "请输入用户名",
                username -> {
                    getDialogService().showInputDialog(s: "请输入密码来登录游戏(用户名不存在则自动注册)",
                        password -> {
                            this.player = new Player();
                            this.player.setName(username);
                            this.player.setPwd(password);
                            playerView = playerEntity.getComponent(PlayerView.class);
                            playerView.login(player);
                        });
                });
        });
    });
    }, Duration.ONE);
}
```

展示对话框

```
//展示对话框的方法
11 个用法
public synchronized void show(List<Talk> talkList) {
    FXGL.getSceneService().pushSubScene(this);
    ArrayList<Talk> arrayList = new ArrayList<>(talkList);
    arrayList.forEach(t -> {
        int length = t.getText().length();
        if (length > 100) {
            for (int i = 0; i < length / 100; i++) {
                String substring = t.getText().substring(0, Math.min(t.getText().length(), 100));
                Talk talk = new Talk();
                talk.setText(substring);
                talk.setAddresserImg(t.getAddresserImg());
                list.add(talk);
                if (t.getText().length() > 100) {
                    t.setText(t.getText().substring(100, t.getText().length() - 100));
                } else {
                    list.add(t);
                    break;
                }
            }
        } else {
            list.add(t);
        }
    });
    nextTalk();
}
```

UI 界面

```
protected void initUI() {
    String[] ui = {"help", "go", "quit", "look", "back", "take", "drop", "items"};
    Image image = new Image(s: "assets/textures/ui/button.png");

    Rectangle r1 = new Rectangle(v: 180, v1: 64);
    r1.setFill(new ImagePattern(image));
    Text t1 = FXGL.getUIFactoryService().newText(ui[0]);
    StackPane help = new StackPane(r1, t1);
    List<String> helpTalks = List.of("You are lost. You are alone.",
        "You wander around at the university.",
        "Your command words are left.");
    List<Talk> talks1 = TalkFactory.buildTalkList(helpTalks);
    help.setOnMouseClicked(mouseEvent -> Platform.runLater(() -> TalkScene.getInstance().show(talks1)));
    addUINode(help, x: 0, y: 30);

    Rectangle r2 = new Rectangle(v: 180, v1: 64);
    r2.setFill(new ImagePattern(image));
    Text t2 = FXGL.getUIFactoryService().newText(ui[1]);
    StackPane go = new StackPane(r2, t2);

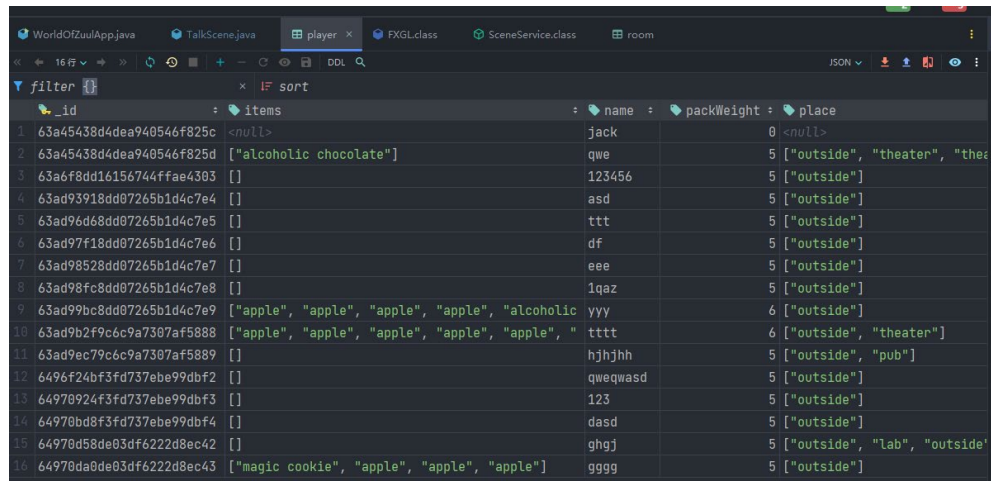
    go.setOnMouseClicked(mouseEvent -> Platform.runLater(() -> {
        List<String> place = FXGLUtils.nowPlayer.getPlace();
        nowPlace = place.subList(place.size() - 1, place.size()).get(0);

        List<String> exits = geto(nowPlace);
        List<String> goTalks = List.of("Go where?",
            "Exits:" + exits);
        List<Talk> talks2 = TalkFactory.buildTalkList(goTalks);
        TalkScene.getInstance().show(talks2);

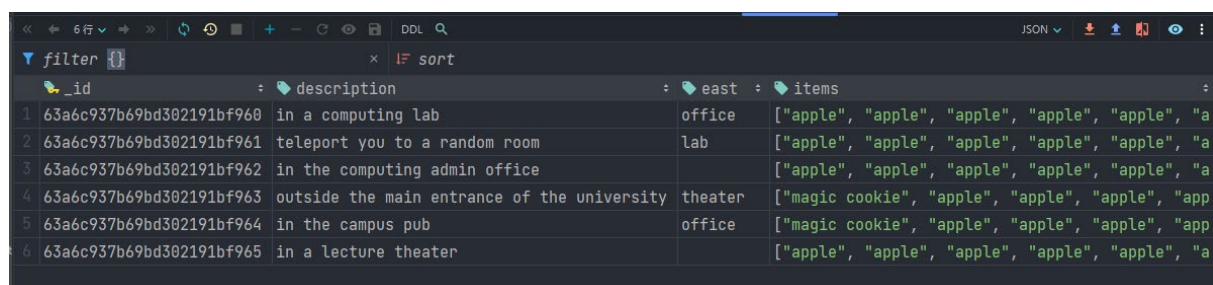
        List<StackPane> dirNodes = new ArrayList<>();

        for (int i = 0; i < exits.size(); i++) {
            Rectangle rect = new Rectangle(v: 180, v1: 64);
            rect.setFill(new ImagePattern(image));
            Text text = FXGL.getUIFactoryService().newText(exits.get(i));
            StackPane stackPane = new StackPane(rect, text);
            int finalI = i;
            stackPane.setOnMouseClicked(mouseEventDir -> {
                dirNodes.forEach(FXGL::removeUINode);
                try {
                    String s = FXGLUtils.newRoom(nowPlace, exits.get(finalI));
                    FXGLUtils.updateRoom(s);
                } catch (NoSuchFieldException | IllegalAccessException e) {
                    e.printStackTrace();
                }
            });
        }
    }));
}
```

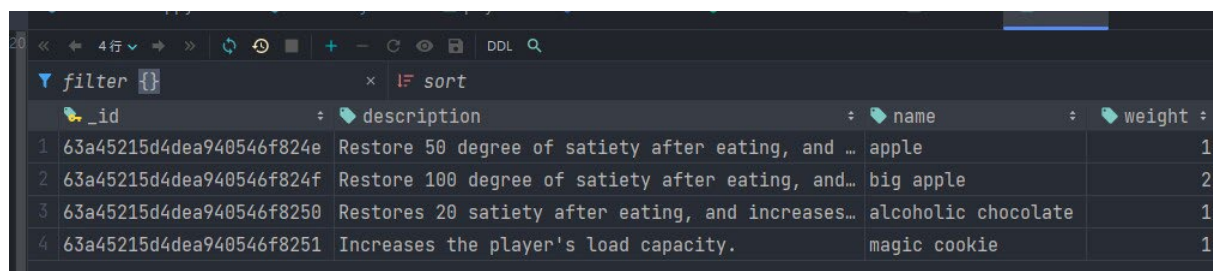

Mangodb 数据库



	_id	items	name	packWeight	place
1	63a45438d4dea940546f825c	<null>	jack	0	<null>
2	63a45438d4dea940546f825d	["alcoholic chocolate"]	qwe	5	["outside", "theater", "theater"]
3	63a6f8dd16156744ffae4303	[]	123456	5	["outside"]
4	63ad93918dd07265b1d4c7e4	[]	asd	5	["outside"]
5	63ad96d68dd07265b1d4c7e5	[]	ttt	5	["outside"]
6	63ad97f18dd07265b1d4c7e6	[]	df	5	["outside"]
7	63ad98528dd07265b1d4c7e7	[]	eee	5	["outside"]
8	63ad98fc8dd07265b1d4c7e8	[]	1qaz	5	["outside"]
9	63ad99bc8dd07265b1d4c7e9	["apple", "apple", "apple", "apple", "alcoholic"]	yyy	6	["outside"]
10	63ad9b2f9c6c9a7307af5888	["apple", "apple", "apple", "apple", "apple", ""]	tttt	6	["outside", "theater"]
11	63ad9ec79c6c9a7307af5889	[]	hjhjhh	5	["outside", "pub"]
12	6496f24bf3fd737ebe99dbf2	[]	qweqwasd	5	["outside"]
13	64970924f3fd737ebe99dbf3	[]	123	5	["outside"]
14	64970bd8f3fd737ebe99dbf4	[]	dasd	5	["outside"]
15	64970d58de03df6222d8ec42	[]	ghgj	5	["outside", "lab", "outside"]
16	64970da0de03df6222d8ec43	["magic cookie", "apple", "apple", "apple"]	gggg	5	["outside"]



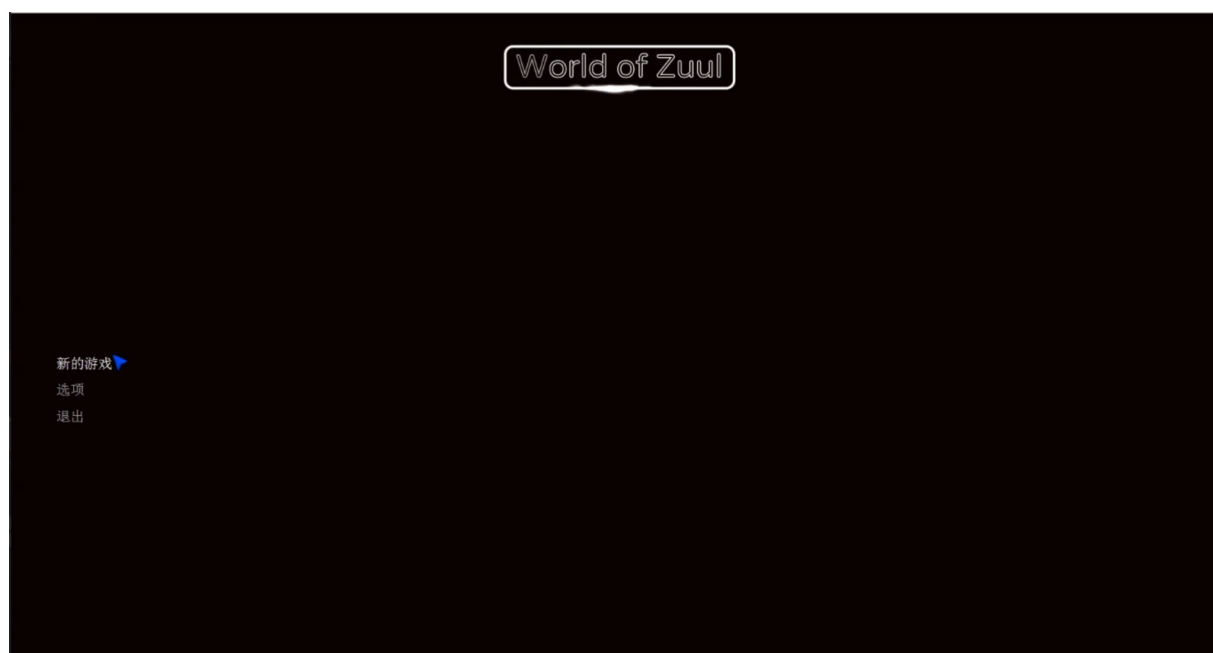
	_id	description	east	items
1	63a6c937b69bd302191bf960	in a computing lab	office	["apple", "apple", "apple", "apple", "apple", "a"]
2	63a6c937b69bd302191bf961	teleport you to a random room	lab	["apple", "apple", "apple", "apple", "apple", "a"]
3	63a6c937b69bd302191bf962	in the computing admin office		["apple", "apple", "apple", "apple", "apple", "a"]
4	63a6c937b69bd302191bf963	outside the main entrance of the university	theater	["magic cookie", "apple", "apple", "apple", "app"]
5	63a6c937b69bd302191bf964	in the campus pub	office	["magic cookie", "apple", "apple", "apple", "app"]
6	63a6c937b69bd302191bf965	in a lecture theater		["apple", "apple", "apple", "apple", "apple", "a"]

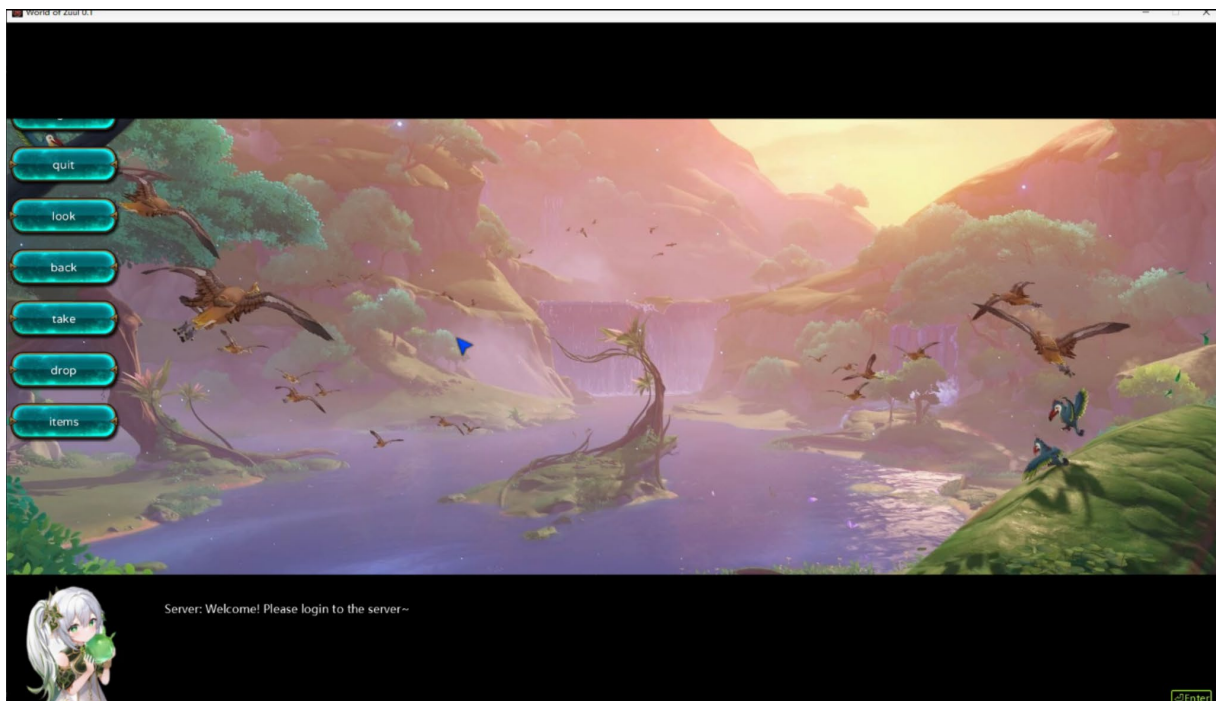


	_id	description	name	weight
1	63a45215d4dea940546f824e	Restore 50 degree of satiety after eating, and ...	apple	1
2	63a45215d4dea940546f824f	Restore 100 degree of satiety after eating, and...	big apple	2
3	63a45215d4dea940546f8250	Restores 20 satiety after eating, and increases...	alcoholic chocolate	1
4	63a45215d4dea940546f8251	Increases the player's load capacity.	magic cookie	1

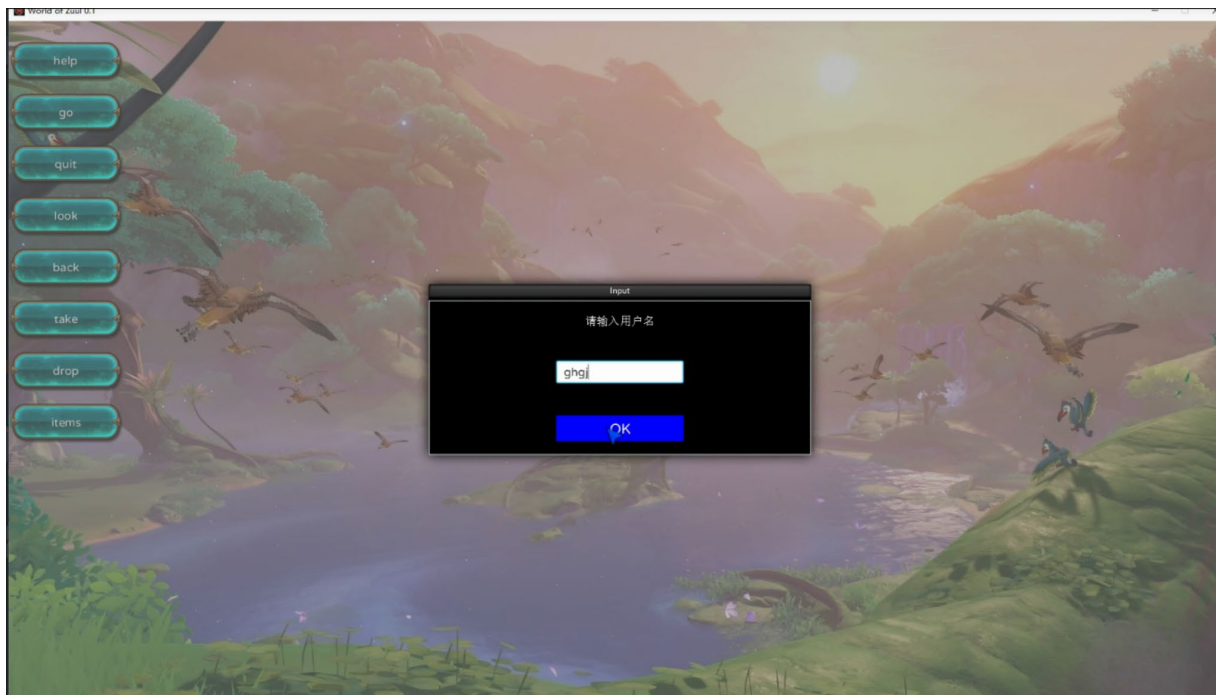
7.2 游戏展示

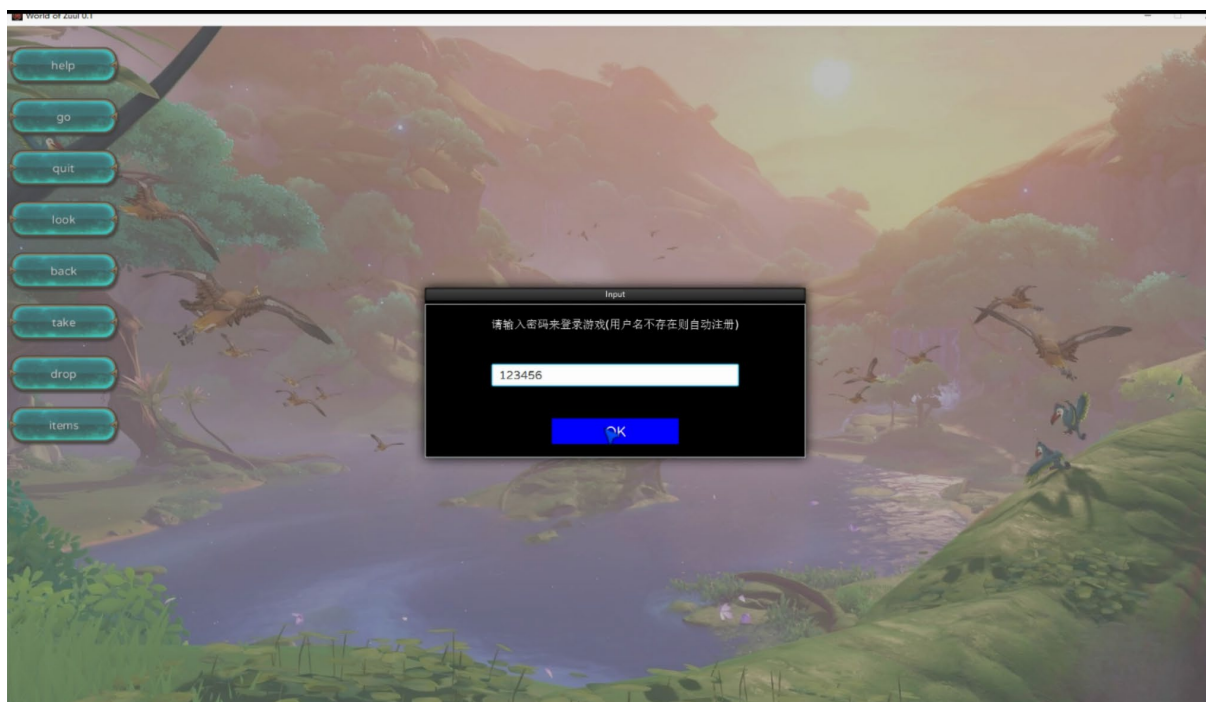
初始界面



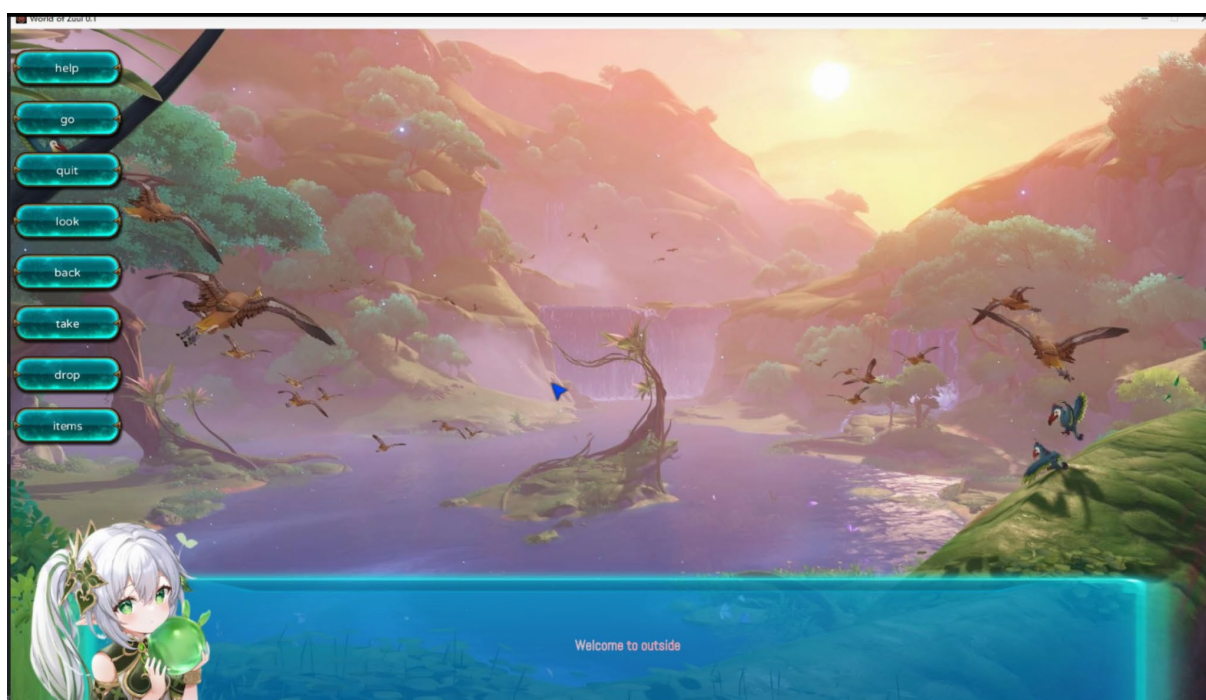


用户登录





登录成功（初始地点为 outside）



Help 功能



Go 功能



前往 south, outside 的 south 为 lab



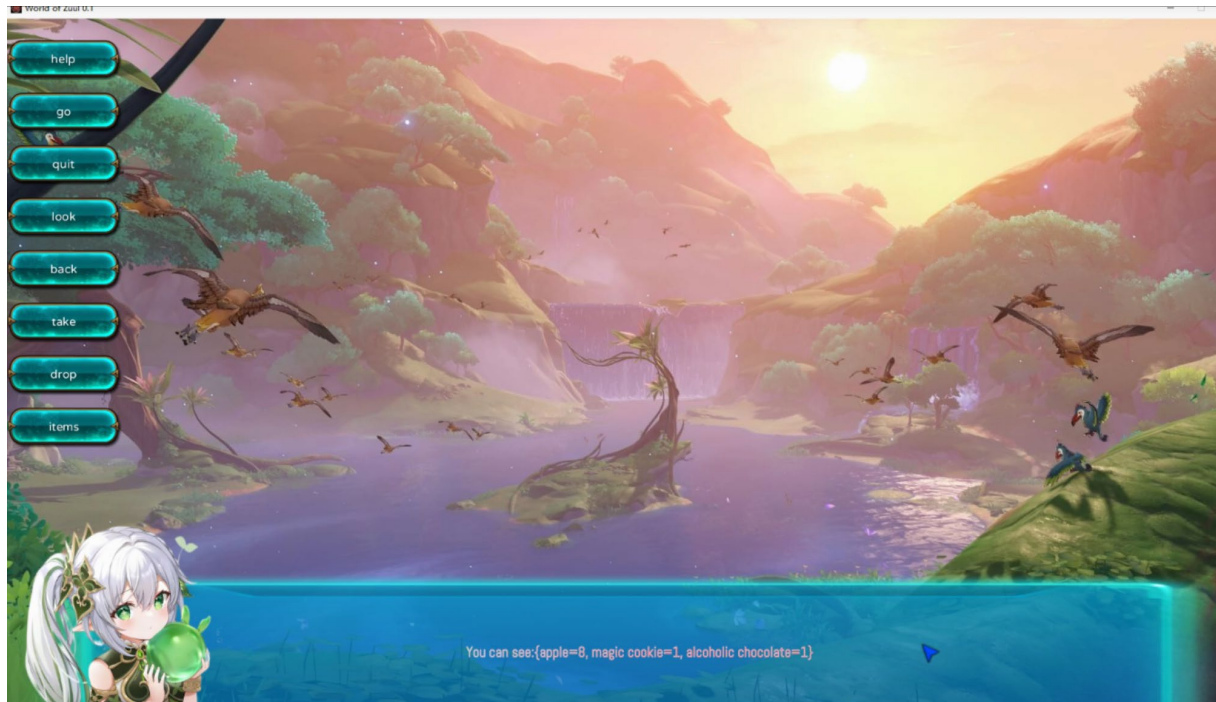
Back 功能



Quit 功能



Look 功能



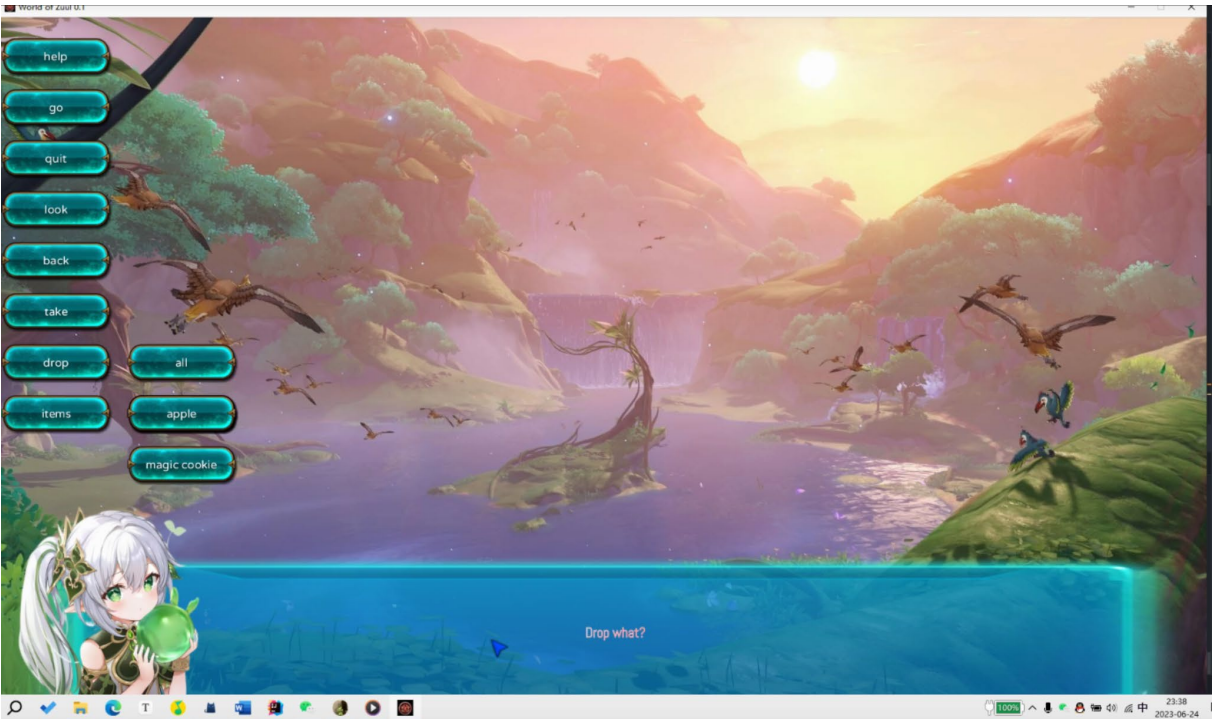
Take 功能



Items 功能



Drop 功能



8 问题记录与分析

以下是本次任务中出现的一些问题和相应的分析：

1. 功能不完整或缺失：

- 问题：某些游戏功能可能没有实现或存在缺陷。
- 分析：可能是因为在设计阶段没有充分考虑所有功能，或者在开发过程中出现了一些问题。
- 解决方案：重新审查游戏功能需求，确保所有功能都得到正确实现，并修复已知的缺陷。

2. 性能问题：

- 问题：游戏在某些情况下可能出现性能问题，例如卡顿或响应延迟。
- 分析：可能是由于算法不优化、资源管理不当或代码效率低下引起的。
- 解决方案：使用合适的算法和数据结构，优化代码逻辑，进行性能测试和调优，确保游戏在各种情况下都能流畅运行。

通过记录和分析这些问题，小组可以识别潜在的风险和改进的机会，从而改进游戏质量和用户体验。在解决问题时，应采取适当的措施，如修改代码、优化算法、增加测试覆盖率等，以确保任务顺利完成以达到预期目标。

9 任务总结

在基于样例工程"World of Zuul"的软件工程实践任务中，我们采用了 FXGL、IOGame、MongoDB 和 Maven 等工具和技术进行 GitHub 小组游戏开发。以下是我们的任务总结：

1. 团队组建和沟通：我们首先组建了一个合适的小组，确保每个成员具有适当的技能和角色分工。
2. 游戏设计和规划：我们仔细分析了"World of Zuul"样例工程，并进行了游戏设计和规划。我们确定了游戏的目标、玩法机制、关卡设计和用户界面等方面的要素，并创建了相应的游戏设计文档。
3. 技术选择：根据游戏需求，我们选择了 FXGL 作为游戏引擎，这为我们提供了创建 2D 游戏所需的必要功能。此外，我们使用 IOGame 作为服务器框架来支持多人游戏和网络功能。为了存储和管理游戏数据，我们使用了 MongoDB 作为数据库。
4. 任务分配和并行开发：我们根据成员的技能 and 兴趣，将任务进行了合理的分配。每个成员负责特定的功能模块或特定任务，以便能够并行开发。我们使用版本控制工具（如 Git）和 GitHub 作为代码管理和协作平台，确保代码的合并和更新能够高效进行。
5. 周期性会议和评审：我们定期举行会议，以进行项目进展更新、解决问题和讨论决策。我们进行了代码评审，以确保代码质量和一致性，并提供反馈和改进建议。
6. 软件测试和调试：我们进行了单元测试、集成测试，以验证游戏功能的正确性和稳定性。我们还进行了调试和错误修复，以解决发现的问题和漏洞。
7. 文档编写和演示：我们编写了适当的文档，如 readme 等。我们进行了游戏演示，以展示游戏的功能和特性，并接受反馈和建议。

8. 持续集成和部署：我们使用 **Maven** 进行项目构建和持续集成，确保代码的稳定性和可靠性。我们设置了自动化的部署流程，以便将游戏部署到合适的环境中进行测试和发布。

通过这些任务和实践，我们的小组成功地开发了一个基于"World of Zuul"样例的游戏，并获得了宝贵的软件工程实践经验。我们充分利用了 **FXGL**、**IOGame**、**MongoDB** 和 **Maven** 等工具和技术，实现了一个功能完善、可玩性高的游戏，并通过 **GitHub** 进行版本控制和团队协作。我们的小组成员在这个过程中不断学习和成长，同时也面临了挑战和解决了各种技术和协作上的问题。总体而言，这次实践任务为我们提供了一个宝贵的学习机会，加强了我们的团队合作和软件开发技能。

10 参考文献

[1] ioGame 网络游戏服务器框架 <https://www.yuque.com/iohao/game>

[2] 吴川,王琢琦. 软件单元测试及测试用例设计方法研究[C]//.第十七届中国航空测控技术年会论文集.[出版者不详],2020:246-248.

[3] Javafx 游戏框架 fxgl <https://github.com/AlmasB/FXGL/wiki>

《软件工程实践（二）》成绩评定表

姓 名	肖峰、张忠瑾、戚家辉、李涵、姜明昆	学 号	0122008910419 0122009360912 0122008890201 0122007780122 0122011350612	
专业、班级	软件 2001			
成绩评定：				
评价内容		满分	实得分	
			得分	小计
实践任务 完成情况	软件项目设计、改进与扩充	20		
	个人软件过程与项目管理	15		
	代码版本管理	25		
	代码注释与编码规范	25		
	单元测试	15		
实践报告 总评情况	学习态度与考勤	10		
	报告格式的规范性	10		
	报告的逻辑结构与语言表达	15		
	实践内容的正确性与合理性	60		
	文献引用及标注	5		
总分		100		
最终评定成绩（以优、良、中、及格、不及格评定）				

指导教师签字：

年 月 日