JavaCraft

Full Project Specification

Friday, October 20, 2023

Table of contents

1 Introduction

Welcome to JavaCraft, a multifaceted text-based Java game. This project is a holistic academic exercise in computer science, logical thinking, and collaboration. Working in teams of four, you will use your creativity, analytical skills, and technical skills. Please see the Project Overview for more information on the intended learning goals of the project.

2 Tasks

2.1 JavaCraft's WorkFlow

JavaCraft is a (*relatively*) complex Java program that brings together over 35 functions to create a diverse gameplay experience. This task is your chance to understand the core functionalities of JavaCraft, putting them together to form the game's overarching structure.

2.1.1 Sub-Tasks

- **Create a Flowchart**: Develop a detailed flowchart that broadly explains the gameplay. This flowchart should contain the main components of the game, illustrating how they interact and connect to create the gaming experience.
- **Write a Pseudocode**: Complement the flowchart with a pseudocode that outlines the main game logic, providing a textual representation of the game's flow and structure.

2.1.2 Deliverables:

- **Flowchart & Pseudocode Submission**: Include the flowchart and pseudocode in the final PDF document, showcasing your understanding of the game's overall architecture[1].

2.2 Functionality Exploration

Dive deep into the codebase of JavaCraft by exploring the 35+ functions that constitute the game's core:

[1]There are two main submissions for this project, you can see the details of which task is part of which submission on the Submissions page.

2.2.1 Sub-Tasks

For each function, perform the following detailed analysis:

1. **Identify Key Components**: Recognize the essential parts of the function, focusing on what makes each function unique and how it contributes to the overall game.
2. **Recognize Patterns**: Delve into the code of each function to find patterns or repetitions, understanding the patterns.
3. **Break Down the Function**: Divide the function into logical steps or subproblems, providing a step-by-step analysis that reveals the function's inner workings.
4. **Represent the Function**: Develop a flowchart and pseudocode for each function, translating the textual analysis into visual and algorithmic representations[2].

2.2.2 Deliverables:

- **Project Documentation**: Prepare a project documentation that includes descriptions of **at least 35** functions within the code. Create flowcharts and pseudocodes for **at least 15** of these functions, showcasing your analytical skills and understanding of the game's mechanics. This documentation will be a part of the provisional and final PDF submission.

2.3 Finite State Automata (FSA) Design

Your next task is to design a Finite State Automata (FSA) for the secret (hidden) door logic:

2.3.1 Sub-Tasks

1. **Study the Secret Door Logic**: Examine the secret door's functionality in the provided code, understanding how the secret door operates within the game's context.
2. **Create an FSA**: Design an FSA (DFA or NFA) that mirrors the behavior of the secret door, ensuring that the FSA accurately captures every aspect of the secret door's functionality.
3. **Illustrate the FSA**: Draw a clear and well-structured FSA diagram, labeling states and transitions, making sure that the diagram is in line with what was covered in the lectures.
4. **Describe the FSA**: Write an explanation of the FSA's operation, detailing the conditions that lead to state changes, and how the FSA represents the secret door logic within the game.

[2] In 2.1, you designed a flowchart and pseudocode for the overall program, here you have to do it for individual functions in the program.

2.3.2 Deliverables:

- **FSA Documentation**: Include a detailed explanation of the secret door logic, a well-designed FSA, and concise documentation of the FSA's functioning and trigger conditions. This documentation will be part of the provisional and final PDF document, reflecting your ability to apply theoretical concepts to practical game elements.

2.4 Extending the Game Code

In this task, you apply your creativity and technical skills by extending JavaCraft's existing code:

2.4.1 Sub-Tasks

1. **Create New Block Types**: Design at least 2 unique block types, considering their defining attributes, interactions, appearances, and how they will integrate into the existing game dynamics.
2. **Formulate a New Craft Recipe**: Develop at least 1 crafting recipe that creatively integrates existing blocks to yield a new and distinct block.
3. **Integrate the Extension**: Incorporate the new block types and craft recipe into the existing game code, ensuring that the new elements work with the existing gameplay mechanics.

2.4.2 Deliverables:

- **Extension Documentation**: Provide a detailed and comprehensive blueprint of the new block types and craft recipe. Include this in the final PDF document, showcasing your ability to innovate within an existing codebase.

2.5 Git Usage

As a part of the project, you will learn Git, a powerful tool for version control, and apply it to the project.

2.5.1 Sub-Tasks

1. **Set Up a Git Repository**: Create a Git repository for your project, all the team members and the TA should have access to it.
2. **Assign Tasks and Branches**: Distribute tasks among team members, creating specific branches for each task, ensuring that each branch serves a clear purpose within the project's development.
3. **Handle Changes and Conflicts**: Use Git's capabilities to manage modifications, merge branches, and resolve conflicts, maintaining a smooth and efficient development process.
4. **Submit the Final Code**: Contribute the completed code, including 2 new block types and 1 craft recipe, to the TA's Git repository[3], adhering to the guidelines and expectations set by your TA.

2.5.2 Deliverables:

- **Git Summary**: Include a summary of your Git usage in the final PDF document. Detail the repository setup, task allocation, change management, and final submission.

2.6 Interacting with Flags API

Once you are done with unlocking the secret door and your intermediary submission[4], you can move on to the last part. The challenge in this part is interacting with the Flags API (flag.ashish.nl). You use this API to know which flag you need to plot. You can choose different levels of difficulty.

2.6.1 Sub-Tasks

1. **Explore the Flags API**: Understand the Flags API (documentation at flag.ashish.nl), grasping how it can be utilized within the game.
2. **Draw the Flag**: Configure the difficulty (easy, medium, and hard) and draw the flag on a 50x30 grid, either manually or through automation[5].

[3] More instructions on this will be provided by your TA.
[4] Please see the submission guidelines for more information on this.
[5] To get you started, I have plotted a flag in the hidden door. You can see how I have automated the process of plotting the flag.

2.6.2 Deliverables:

Flags Documentation: Detail the Flags API's functionality and your flag drawing process in the final PDF document. Include the chosen difficulty level and how the flag drawing integrates into the game, reflecting your ability to work with external APIs.

3 Final Deliverable

Compile a single PDF document, not exceeding 10 pages, detailing all the above steps in a structured and coherent format[6]. Include pseudocode and flowcharts for part 3 in the appendix, as lengthy as needed. Ensure that all deliverables mentioned are included in this document, except for the Git submission, which will be submitted separately to the TA's repository.

[6]You can use the template provided at bcs1110.ashish.nl.