**Project Template**

**Project Report: Group 66**

Friday, October 20, 2023

## Table of contents

[**1 Introduction** **2**](#_Toc3329)

[**2 JavaCraft’s Workflow** **2**](#_Toc3330)

[**3 Functionality Exploration** **2**](#_Toc3331)

[**4 Finite State Automata (FSA) Design** **3**](#_Toc3332)

[**5 Git Collaboration & Version Control** **3**](#_Toc3333)

[**6 Extending the Game Code (For Final Submission)** **3**](#_Toc3334)

[**7 Interacting with Flags API (For Final Submission)** **3**](#_Toc3335)

[**8 Conclusion (For Final Submission)** **3**](#_Toc3336)

[**9 Appendix** **3**](#_Toc3337)

[**10 References** **3**](#_Toc3338)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | | Details | | |
| Group Name | | [Insert Group Name] | | |
| Group Number | | [group 66] | | |
| TA | | [Pavel Detvan] | | |
| Student Name | | Student ID |
| [Filippo] | | [Student ID] |
| [Giannis Fourlas] | | [Student ID] |
| [Maarten Koji] | | [Student ID] |
| [Alexandra Plishkin] | | [Student ID] |

# Introduction

[Provide a brief introduction to the report here.]

# JavaCraft’s Workflow

* Flowchart For Game: [attach image]
* Pseudocode For Game: [Provide pseudocode here]

# Functionality Exploration

List of key functionalities explored:

|  |  |  |
| --- | --- | --- |
| No. | Function Name | Description |
| 1 | [initGame] | [generates the square] |
| 2 | [generateWorld] | [generates the squares with blocks] |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | [displayWorld]  [getBlockSymbol]  [getBlockChar]  [startGame]  [fillInventory]  [resetWorld]  [generateEmpty  World]  [clearScreen]  [lookAround]  [movePlayer]  [mineBlock]  [placeBlock]  [getBlockType  FromCraftedItem]  [getCraftedItem  FromBlockType]  [displayCrafting  Recipes]  [craftItem]  [craftWooden  Planks]  [craftStick]  [craftIronIngot]  [inventoryContains]  [inventoryCointains]  [removeItemsFrom  Inventory]  [addCraftedItems]  [interactWithWorld]  [saveGame]  [loadGame]  [getBlockName]  [displayLegend]  [displayInventory]  [getBlockColor]  [waitForEnter]  [getCraftedItem  Name]  [getCraftedItem  Color]  [getCountryAnd  QuoteFromServer] | [creates the square with the blocks]  [gives color to the blocks]  [gives a name to each block]  [controls the actions in the game]  [allows to keep the items in the inventory]  [resets the world and generates a new one, opening the secret door]  [generates the secret room]  [clears the terminal]  [if you type “look” it shows which blocks are around you]  [defines the keys and words to move the player in certain directions]  [it allows the player to mine the blocks]  [it allows the player to place the mined blocks in the inventory]  [it allows you to get the block type from a crafted item]  [getting a crafted item from a block type]  [tells you what you need to craft something]  [initiates the process of crafting]  [completes the crafting for wooden planks]  [completes the crafting for wooden planks]  [completes the crafting for iron ingot]  [it says what you have in your inventory]  [it says how many of each object you have in your inventory]  [it allows you to remove the items from your inventory]  [adds items to a new category called “Crafted Items” below your inventory]  [it tells to you what you have done]  [saves your progress]  [allows you to load your previous save]  [gives the name to the blocks  [writes the blocks legend]  [writes the inventory]  [gives color to different blocks in the legend]  [the program waits for you to type “Enter”]  [it gives a name to the item you crafted]  [gives a color to the item you crafted]  [connects you to the database and the server location] |

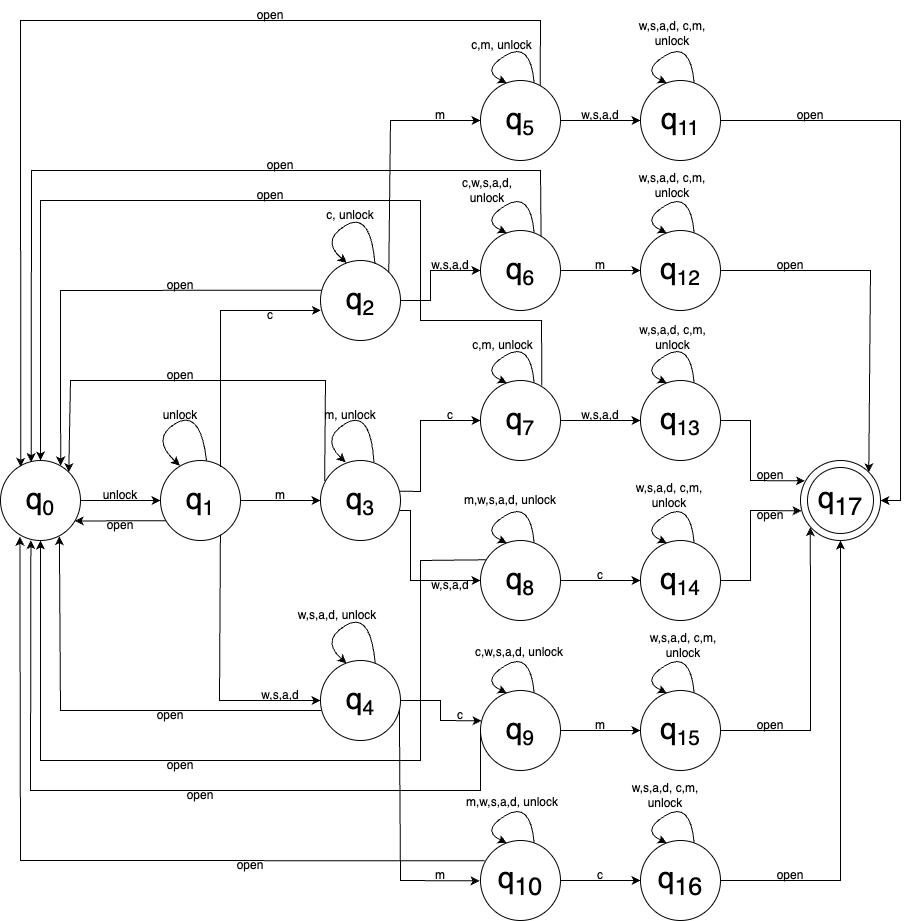
Detailed description for each function:

1. [initGame]: Key variables are declared and initialized. These include the world width and height, the size of the world, player X and Y, and the inventory. The method declares static variables, with the integer parameters of the world width and world height.
2. [generateWorld]: Generates a random value, and relies on probability density to set each coordinate (using two for loops) in the world grid to a particular block (wood, leaves, stone, iron ore, and air), with air being the most prominent, occurring 70% of the time.
3. [displayWorld]: Displays world borders through an equation that uses the world width and height values, as well as printing “World map:” in cyan. Also sets the character “p” to represent the player, with the color of the character changing based on if the player is in the secret area or not, which is accomplished through if-else-if statements.
4. [getBlockSymbol]: Sets the character of air to a hyphen, assigns colors to all the blocks through the use of a switch statement. Calls the getBlockChar method, and returns it with the parameter blockType of type int.
5. [getBlockChar]: Sets the characters of all the other blocks, by assigning them a specific unicode character, again with a switch statement, considering 4 cases and a default.
6. [startGame]: Calls a multitude of static public methods to produce the backbone of the game. Also includes the code to unlock the secret door and enter the secret area, which are discussed in detail in a further section.
7. [fillInventory]: Clears the inventory, adds each blockType (1-4) to the inventory through a for loop.
8. [resetWorld]: Calls the static generateEmptyWorld method to print the secret area, and initializes the player’s X and Y coordinates.
9. [generateEmptyWorld]: Prints the layout of the secret area, by dividing the grid in 3, with each division being assigned the color of red, white, and blue blocks in order, with double for loops to fill the 2D grid.
10. [clearScreen]: This function clears the user's screen, making a distinction between windows and non-windows systems. Therefore, it checks for the type of system and uses specific commands to clear the screen, for windows: (cmd, cls, /c) and for another operating system: (\033,[H\033[2J), which are ANSI escape codes. So, it allows you to clear the screen of a device with any operating system.
11. [lookAround]: This function is used to look around in the game. So, if the player types “look”, it would tell the player which are the blocks around. For this, assigns *x* and *y* as integers values related to the world’s disposition, after this, those values are determined to playerY and playerX. And uses a loop with some mathematical equations to determine exactly the position in the world and what’s around the player.
12. [movePlayer]: It’s what allows the player to move through the map of the game. For this, it defines the keys and words to move the player in certain directions. It defines a string variable “direction”, which is therefore used in a switch case with all the possible movements for the player. Also, in each case, there’s an if statement that determines if the player is allowed to move to that direction or if he/she already arrived at the border of the map.
13. [mineBlock]: Allows the player to mine the blocks into the game. It uses an integer variable “blockType”, which is related to the player position in the world. It determines if the type of block the player is trying to mine is not air, and if it’s not air, the player is able to mine it and the block is removed from the map and substituted by air; on the other hand, if it’s air, a message will pop up saying that there’s no block to mine there.
14. [placeBlock]: Allows the player to place the mined blocks and crafted items the player has in the inventory. It uses the integer variable “blockType” and determines what’s the integer value of the block the player is trying to place. If the value is bigger or equal to zero and smaller or equal to seven, there are three different possibilities; this is determined by if-else statements. In case of being smaller or equal to 4, it checks if the inventory contains the type of block, and in case of being there, it’s removed from the inventory and placed in the world, determining the place with playerX/playerY; in case of not having the block in the inventory, it would tell the player that there’s no block there. If the block is not smaller or equal to 4, it would generate an integer value “craftedItem”, which is related to the block type, as you get the crafted item from a block type; if that crafted item is one of the items you have in the inventory designated as “Crafted items”, the item will be removed from the inventory and placed in the world; on the other side, a message will pop up saying that there’s no crafted item. If the integer typed wasn’t any of the previous options, the program will say that it’s invalid and will provide the player with a list of the block numbers information.
15. [getBlockTypeFromCraftedItem]: It’s used to get the block type from a crafted item. It uses the integer variable “craftedItem” and a switch-case to return integer values that are designated to each “blockType”.
16. [getCraftedItemFromBlockType]: It’s used to get a crafted item from a block type. It uses the integer variable “blockType” and a switch-case to return integer values that are designated to each “craftedItem”.
17. [displayCraftingRecipes]: This function displays the crafting recipes on a list to inform the player of what’s allowed to be crafted and with which materials. It simply uses the print output to display the list.

Note: Provide flowchart and pseudocode for at least 15 functions in the Appendix.

# Finite State Automata (FSA) Design

* Secret Door Logic Analysis: [Describe the secret door’s functionality]
* FSA Illustration & Description: [Attach FSA diagram]



# Git Collaboration & Version Control

* Repository Link: https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft.git
* Branch Details: Branch group66

List branch names and corresponding members • Changes & Conflicts: Discuss how changes and conflicts were handled.

# Extending the Game Code (For Final Submission)

[Provide details on the new block types, craft recipes, and their integration into the game. Include code snippets where appropriate]

[Provide Java code here]

# Interacting with Flags API (For Final Submission)

[Details on Flags API exploration and flag rendering on the grid.]

# Conclusion (For Final Submission)

[Provide a summary of achievements, challenges, and learnings.]

# Appendix

Include any additional pseudocode, flowcharts, or supplementary material.

1. Flowchart & pseudocode displayInventory:

A diagram of a block diagram

Description automatically generatedIf the inventory is empty, print ‘Empty’ in *yellow.*

Create an array that lists the number of blocks in the inventory.

For all elements in the inventory,

Set the integer ‘block’ to the value of that element,

add 1 to the block type specified by ‘block’ in the array ‘blockCounts’.

For all elements in ‘blockCounts’,

Set the integer ‘occurrences’ to the element,

If ‘occurrences’ >0, print the name of the block along with the occurrences.

Print “Crafted Items:”

If there are none, print “None” in *yellow.*

Else, for all items in the ‘craftedItems’ array, print the color and name of the item.

Print 2 empty lines.

End

1. Flowchart & pseudocode mineBlock:

A diagram of a block type flow

Description automatically generated

Void mineBlock

            Define the blockType within the playerX and playerY

            If the block is not air

                           Add the blockType

                           Substitute the block type into air

                           Print (“Mined (blockType) .”

            Else

                           Print “No block to mine here.”

            Wait for the player to press Enter

            End

1. Flowchart & pseudocode placeBlock:

A diagram of a computer

Description automatically generatedVoid placeBlock

int blockType

If blockType>=0 and <=7

If blockType <=4

If inventory contains blockType

Remove blockType from inventory

Place blockType in the world within the playerX and playerY

Print (“Placed (blockType) at your position.”)

Else

Print (“You don’t have (blockType) in your inventory)

Else

int craftedItem (get the crafted item from the blocktype)

If craftedItems contains the craftedItem

Place blockType in the world within the playerX and playerY

Print (“Placed (craftedItem name) at your position.”)

Else

Print (“You don’t have (craftedItem name) in your crafted items.”)

Else

Print (“Invalid block number. Please enter a valid block number.”)

Print block numbers info

Wait for the player to press Enter

End

1. Flowchart & pseudocode displayCraftingRecipes:

Void displayCraftingRecipes

A diagram of a craft

Description automatically generated Print (“Crafting Recipes.”)

Print (“1. Craft Wooden Planks: 2 Wood”)

Print (“2. Craft Stick: 1 Wood)

Print (“3. Craft Iron Ingot: 3 Iron Ore”)

End

1. Flowchart & pseudocode getBlockName:

A diagram of a wood processing process

Description automatically generated

String getBlockName

int blockType

Switch (blockType)

Case AIR:

Return “Empty Block”

Case WOOD:

Return “Wood”

Case STONE:

Return “Stone”

Case IRON\_ORE:

Return “Iron Ore”

Default:

Return “Unknown”

End

1. Flowchart & pseudocode generateEmptyWorld:

A diagram of a diagram

Description automatically generatedfunction generateEmptyWorld():

    create an empty 2D array called world with dimensions NEW\_WORLD\_WIDTH by NEW\_WORLD\_HEIGHT

    set redBlock to 1

    set whiteBlock to 4

    set blueBlock to 3

    set stripeHeight to NEW\_WORLD\_HEIGHT divided by 3  // Divide the height into three equal parts

    // Fill the top stripe with red blocks

    for y from 0 to stripeHeight - 1:

        for x from 0 to NEW\_WORLD\_WIDTH - 1:

            set world[x][y] to redBlock

    // Fill the middle stripe with white blocks

    for y from stripeHeight to (stripeHeight \* 2) - 1:

        for x from 0 to NEW\_WORLD\_WIDTH - 1:

            set world[x][y] to whiteBlock

    // Fill the bottom stripe with blue blocks

    for y from (stripeHeight \* 2) to NEW\_WORLD\_HEIGHT - 1:

        for x from 0 to NEW\_WORLD\_WIDTH - 1:

            set world[x][y] to blueBlock

1. A diagram of a flowchart

   Description automatically generatedFlowchart & pseudocode craftItem:

void CraftItem

Switch: four possible cases

Case 1:  call craftWoodenPlanks function

Case 2: call craftSticks function

Case 3: call craftIronIngot

default: print “invalid recipe number”

Wait for the player to write Enter

End

A flowchart of a wood product

Description automatically generated

1. Flowchart & pseudocode craftWoodenPlanks:

If the inventory does not contain 2 wood, print "Insufficient resources to craft Wooden Planks."

Else, remove the necessary items from inventory,

Add the crafted wooden plank,

Print "Crafted Wooden Planks."

End

1. Flowchart & pseudocode initGame;

**A diagram of a program

Description automatically generated**

function initGame(worldWidth: int, worldHeight: int):

    Set JavaCraft.worldWidth to worldWidth

    Set JavaCraft.worldHeight to worldHeight

    Create a new 2D array JavaCraft.world with dimensions worldWidth by worldHeight

    Set playerX to worldWidth / 2

    Set playerY to worldHeight / 2

    Create a new empty Array list

1. Flowchart & pseudocode fill\_Inventory;

A diagram of a flowchart

Description automatically generatedfillInventory()

     Clear the contents of the inventory list

     loop for blockType from 1 to 4

        loop for i from 1 to INVENTORY\_SIZE

            add blockType to the inventory list

        End loop

    End loop

End

1. Flowchart & pseudocode resetWorld:

A diagram of a game

Description automatically generated

resetWorld()

    Call generateEmptyWorld()

    Set playerX to worldWidth/2

    Set playerY to worldHeight/2

End

1. Flowchart & pseudocode movePlayer:

A diagram of a flowchart

Description automatically generated

Take user input

Capitalize user input

5 cases:

Case 1: Input is W / UP

If Y position>0, decrease Y position by 1. Terminate.

Case 2: Input is S / DOWN

If Y position<WorldHeight-1, increase Y position by 1. Terminate.

Case 3: Input is A / LEFT

If X position>0, decrease X position by 1. Terminate.

Case 4: Input is D / RIGHT

If X position<WorldWidth-1, increase X position by 1. Terminate.

Case 5: Any other inputs

Terminate.

End

1. Flowchart & pseudocode waitForEnter:

A flowchart of a printer

Description automatically generated

Print “Press Enter to continue…”

Read user input, type: String.

End.

1. Flowchart & pseudocode craftWoodenPlank:

A diagram of a wood plank

Description automatically generated

If the inventory contains two items of WOOD

remove two items of WOOD from the inventory

add to the inventory one items of CRAFTED\_WOODEN\_PLANKS

print “Crafted Wooden Planks.”

else

print “Insufficient resources to craft Wooden Planks.”

1. Flowchart & pseudocode craftSticks:

A diagram of a process flow

Description automatically generated

If the inventory contains one items of WOOD

remove one items of WOOD from the inventory

add to the inventory one items of CRAFTED\_STICK

print “Crafted Stick.”

Else

print “Insufficient resources to craft Stick.”

1. Flowchart & pseudocode craftIronIngot:

A flowchart of a process

Description automatically generated

If the inventory contains two items of IRON\_ORE

remove three items of IRON\_ORE from the inventory

add to the inventory one items of CRAFTED\_IRON\_INGOT

print “Crafted Iron Ingot.”

Else

print “Insufficient resources to craft Iron Ingot.”

1. Flowchart & pseudocode interactWithWorld:

A diagram of a process

Description automatically generated

define the type of the block that the player is on

switch: six possible cases depending on the type of block

case WOOD:

print “you gather wood from the tree”

add to the inventory one item of LEAVES

break

case STONE:

print “you gather stone from the ground.:”

add to the inventory one item of STONE

                break

case IRON\_ORE:

print “you gather iron ore from the ground.:”

add to the inventory one item of IRON\_ORE

break

case AIR:

print “Nothing to interact with here.:”

break

case DEFAULT:

print “Unrecognized block. Cannot interact.:”

Wait for the player to type “ENTER”.

# References

1. Source Name - Description
2. …