Project Template

Project Report: Group73

Friday, October 20, 2023

Table of contents

Attribute	Details
Group Name	Group73
Group Number	73
TA	Tom

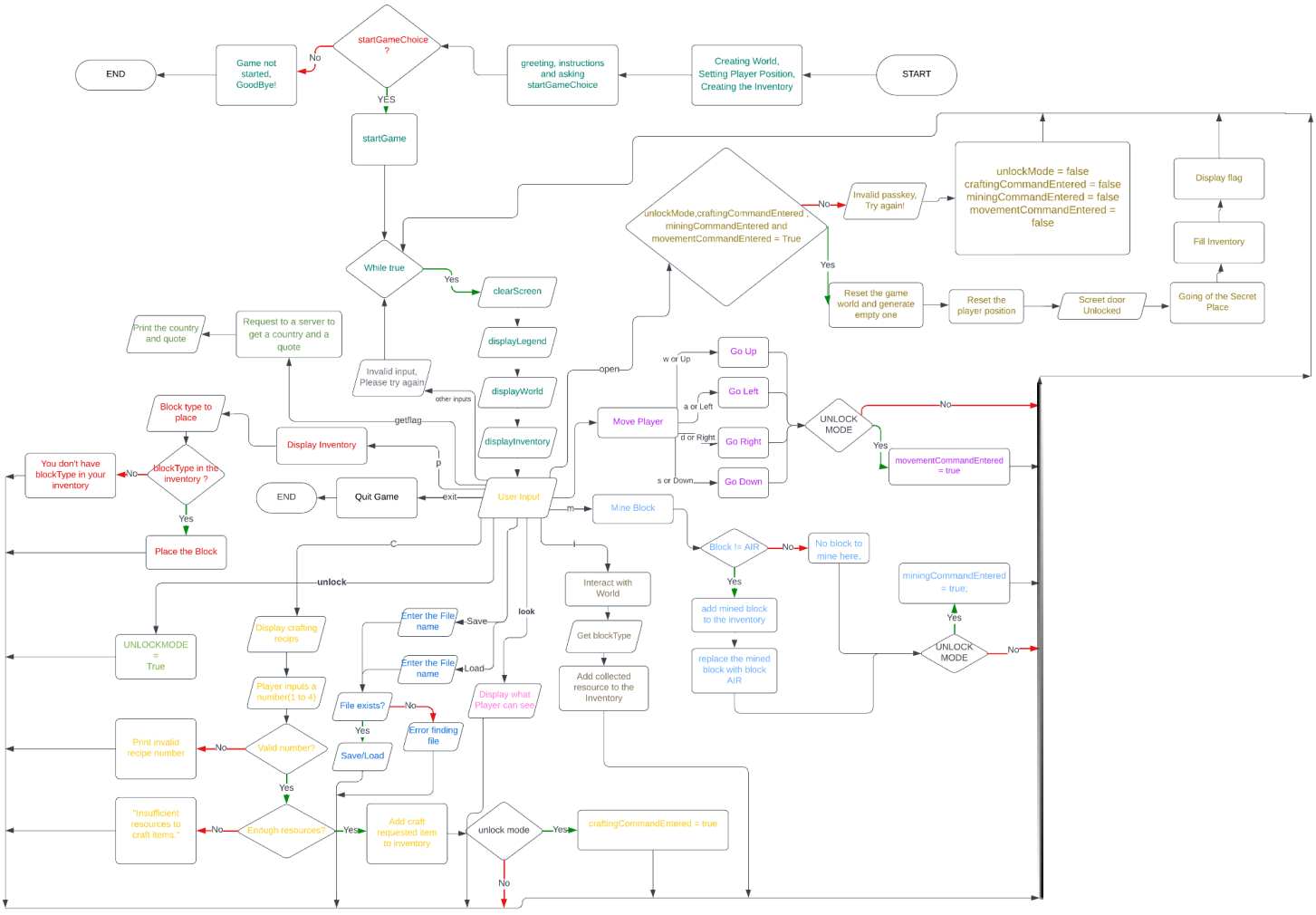
Student Name	Student ID
Adrian Rusu	i6359689
Abdulla Al Fayyumi	i6253594
Mehmet Osman Acar	i6361732
Amir Kalantarzadeh	i6363421

1 Introduction

Adrian Rusu - description for the functions, made the new blocks/recipe, secret door FSA, organized the pdf file.
Abdulla Al Fayyumi - the game's pseudocode, secret door FSA, 5 pseudocodes and flowcharts for functions, FSA transition table.
Mehmet Osman Acar - 50% of the game flowchart, 5 pseudocodes and flowcharts for functions, secret door logic analysis
Seyedamirreza Kalantarzadeh - 50% of the game flowchart, 5 pseudocodes and flowcharts for functions, secret door logic analysis.

2 JavaCraft's Workflow

- Flowchart For Game: [attach image]



- Pseudocode For Game:

```
 function initGame(worldWidth, worldHeight):
      Set JavaCraft.worldWidth to worldWidth
      Set JavaCraft.worldHeight to worldHeight
      Create a new empty world of size worldWidth x worldHeight
      Set playerX to worldWidth / 2 Set playerY to worldHeight / 2
      Create an empty inventory list
 end function

function generateWorld():
     For each y from 0 to worldHeight:
         For each x from 0 to worldWidth:
              Generate a random number randValue
              if randValue is less than 30:
                   Set world[x][y] to WOOD
              else if randValue is less than 40: Set world[x][y] to LEAVES
              else if randValue is less than 60: Set world[x][y] to STONE
              else if randValue is less than 80: Set world[x][y] to
              IRON_ORE else if randValue is less than 95: Set world[x][y]
              to GOLD_ORE else if randValue is less than 105:
                   Set world[x][y] to DIAMOND_ORE
              else: Set world[x][y] to AIR

end function

    function displayWorld():
     Print "World Map:"
     For each y from 0 to worldHeight:
          Print "‖"
          For each x from 0 to worldWidth:
               If (x equals playerX) and(y equals playerY) and(not in
                   SecretArea):
                    Print "P "
              Else If (x equals playerX)and(y equals
              playerY)and(inSecretArea):
                    Print "P "
              Else:
```

```
                    Print the symbol for the block at world[x][y]
          Print "╟"
          Print "╚═══" + "═".repeat(worldWidth * 2 - 2) + "╝"
end function

function handleInput(input):
     If input is "exit":
          Print "Exiting the game. Goodbye!"
          Exit the game loop
     Else If input is in movements: Call handleMovement(input)
    Else If input is "m": Call handleMining()
    Else If input is "p": Call handlePlacing()
    Else If input is "c": Call handleCrafting()
    Else If input is "i": Call interactWithWorld()
    Else If input is "save": Call handleSaving()
    Else If input is "load": Call handleLoading()
    Else If input is "look": Call lookAround()
    Else If input is "unlock": Set unlockMode to true
    Else If input is "getflag": Call getCountryAndQuoteFromServer()
    Else If input is "open": Call handleSecretDoor()
    Else: Print "Invalid input. Please try again."
end function

    function handleSecretDoor()
          if (unlockMode is true and craftingCommandEntered is true and
          miningCommandEntered is true and movementCommandEntered is true):
               set openCommandEntered to true set secretDoorUnlocked to true
               resetWorld()
               display "Secret door unlocked!"
          else
               display "Invalid passkey. Try again!"
               set unlockMode to false
               set craftingCommandEntered to false
               set miningCommandEntered to false
               set movementCommandEntered to false
               set openCommandEntered to false
    end function
```

```
function handleLoading()
     display "Enter the file name to load the game state: "
     read fileName
     loadGame(fileName)
end function

function handleSaving()
     display "Enter the file name to save the game state: "
     read fileName
     saveGame(fileName)
end function

function handleCrafting()
     if unlockMode is true
          set craftingCommandEntered to true
     displayCraftingRecipes()
     display "Enter the recipe number to craft: "
     read recipe craftItem(recipe)
end function

function handlePlacing()
     displayInventory()
     display "Enter the block type to place: "
     read blockType
     placeBlock(blockType)
end function

function handleMining()
     if unlockMode is true
          set miningCommandEntered to true
     mineBlock()
end function

function handleMovement(input)
     if unlockMode is true
          set movementCommandEntered to true
      movePlayer(input)
end function
```

3 Functionality Exploration

List of key functionalities explored:

1. initgame	Sets the dimensions of the game world, creates a 2D array to represent the world, position the player in the center, and initializes an empty inventory list.
2. generateWorld	Generates the 2D array using random values between 0 and 100. Randomly assigns blocks as wood if the random number generated is less than 20, stone if less than 50, iron if the random number is less than 70, leaves if the random number is less than 35 and the rest of the world is filled with air.
3. displayWorld	Displays the world and outlines it with a border. If the player is in the secret area, the “P” will be blue, else it will be green.
4. getBlocksymbol	Returns a string representing that block type with associated ANSI color codes.
5. getBlockChar	Returns a character representing that block type. It uses a switch statement to specific block types (e.g., WOOD, LEAVES, STONE, IRON_ORE) to corresponding Unicode characters (e.g., '\u2592', '\u00A7', '\u2593', '\u00B0').
6. fillInventory	Ensures the inventory is empty and makes each block type have a maximum storage of 100 blocks.
7. resetWorld	Resets the world and positions the player in the center of the world.
8. generateEmptyWorld	Generates a new world and it fills in the top, middle and bottom stripes with white, red and blue blocks.
9. clearScreen	Checks if the player's operating system is Windows, if yes then it clears the terminal using the “cls” command. If not, it uses ANSI escape codes ‘’\033[H\033[2J’’ to clear the screen.
10. lookAround	Displays a 3x3 array with the player in the middle and the blocks that surround the player.
11. movePlayer	Moves the player in the 2D world (x and y axis).
12. mineBlock	If the block is different from Air, it adds the block in the inventory and tells what you have mined. If the block is Air, it tells that there is nothing to be mined.
13. placeBlock	Waits for the user’s input to be a number between 0 (inclusive) and 7 (inclusive) the number representing the block type if the input <= 4, else representing a crafted item. After the number was entered, it places the said

	block/Item on the player's position.
14. getBlockTypeFromCraftedItem	Converts a block type to a crafted item. It uses the integer variable "blockType" and a switch-case to return integer values that are designated to each "craftedItem". It uses the integer variable "craftedItem" and a switch-case to return integer values that are designated to each "blockType".
15. getCraftedItemFromBlockType	Converts a crafted item to a block type.It uses the integer variable "craftedItem" and a switch-case to return integer values that are designated to each "blockType".
16. displayCraftingRecipes	Displays all possible crafting recipes.
17. craftItem	Crafts an item based on the player's input, 1 being for wooden planks, 2 for sticks and 3 for iron ingots.
18. craftWoodenPlanks	If there are at least 2 pieces of wood in inventory, remove the 2 pieces of wood from inventory and replace them with a wooden plank.
19. craftSticks	If there is at least 1 piece of wood in inventory, remove the 1 piece of wood from inventory and replace them with a stick.
20. craftIronIngots	If there are at least 3 pieces of iron ore in inventory, remove the 3 pieces of n from inventory and replace them with an iron ingot.

21. inventoryContains	Is a boolean function which checks if the player has enough blocks of the necessary block type so he can craft, if yes, it returns true, if not, it returns false.
22.removeItemsfromInventory	Removes the number of an item from inventory. This is used after crafting.
23. addCraftedItem	If the player didn't have an item before, it creates a new ArrayList named craftedItems before adding the item in the inventory.
24. interactWithWorld	When the player input = "I", checks if there is any block Type on the player's coordinates. If there is, it adds the block which was interacted with in the inventory.
25. saveGame	Serialize game state data and write to the file. Saves data such as NEW_WORLD_WIDTH, NEW_WORLD_HEIGHT, world, playerX, playerY, inventory, craftedItems, unlockMode. Also tells you if the file was saved (if yes, where?) or there was an error.
26. loadGame	Read the file in which the world was saved. reads data such as

	NEW_WORLD_WIDTH, NEW_WORLD_HEIGHT, world, playerX, playerY, inventory, craftedItems, unlockMode. Also tells you if the file was loaded(if yes, from where?) or there was an error.
27. getBLockname	Takes the parameter blockType and returns a name for each block.
28. displayLegend	Displays the legend of the World, shows the characters and color that are assigned to each block.
29. displayInventory	Displays the player's inventory and the count of items
30. getBlockColor	Gives the blocks color using the ANSI escape code method, using a switch statement, the case defines the block type and gives the said block its color.
31. waitForEnter	Waits for the user's "Enter" input so the code can continue
32. getCraftedItemName	Takes the parameter craftedItem and returns a name for each crafted item.
33. getCrafteditemcolor	Takes the parameter craftedItem and assigns the color brown to all crafted items.
34. getCountryanQuotefromServer	It establishes a connection to a remote server using HTTP POST. It sends an empty JSON payload to the server and reads the server's response as a JSON string. It extracts and prints a "country" and a "quote" from specific positions in the JSON response.
35. startGame	Starts the game, sets multiple boolean functions to false, prints a text showing are the player's possible moves. It calls different existing functions in the code depending on what the user input is.

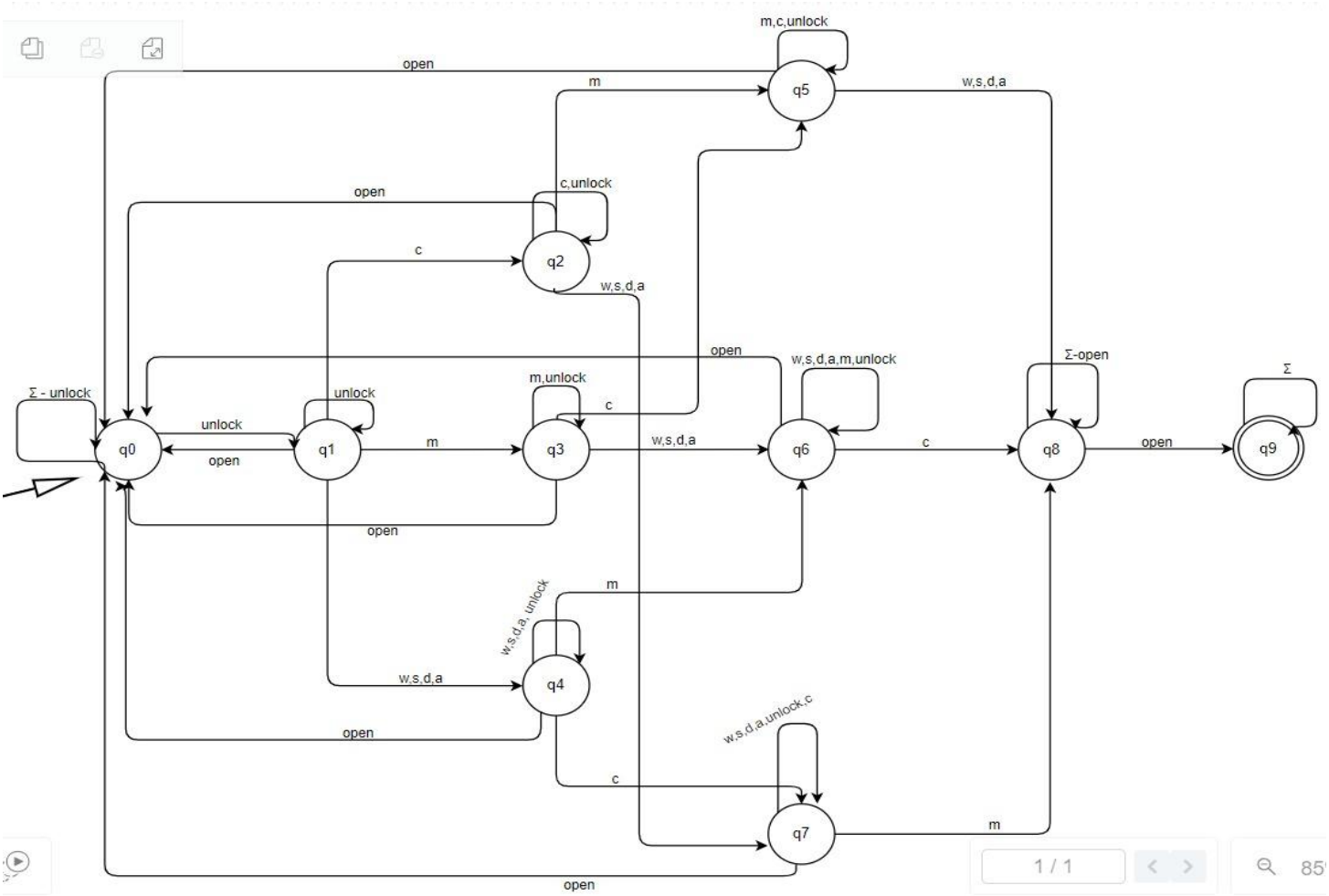
Note: Provide flowchart and pseudocode for at least 15 functions in the Appendix.

4 Finite State Automata (FSA) Design

- Secret Door Logic Analysis:

Within JavaCraft's source code lies a concealed sequence for accessing a secret door area. This functionality relies on several functions, starting with 'unlockMode,' initially set to false. To activate it, the player must type 'unlock.' Subsequently, 'c' (for craft), 'm' (for mine), and a movement command ('w,' 'a,' 's,' or 'd') must be entered, with no specific order. However, 'open' must be the last command; entering it prematurely resets the boolean. On success, the player unlocks the secret door, clearing the screen to reveal the Dutch flag and messages proclaiming entry into the secret area and the game board with the flag.

FSA Illustration & Description:



	unlock	w,s,d,a	open	c	m
→ q0	q1	q0	q0	q0	q0
q1	q1	q4	q0	q2	q3
q2	q2	q7	q0	q2	q5
q3	q3	q6	q0	q5	q3
q4	q4	q4	q0	q7	q6
q5	q5	q8	q0	q5	q5
q6	q6	q6	q0	q8	q6
q7	q7	q7	q0	q7	q8
q8	q8	q8	q9	q8	q8
*q9	q9	q9	q9	q9	q9

5 Git Collaboration & Version Control

- Repository Link: https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft/-/tree/group73?ref_type=heads
- Branch Details:
  -Members: Adrian Rusu, Abdulla Al Fayyumi, Mehmet Osman Acar, Amir Kalantarzadeh
  -Documents in the branch: FlowCharts, JavaCraft.java, JavaCraft35functions.docx, README.md, GameFlow.

In the branch we added the report and uploaded our code. We managed to handle the conflicts and upload our changes to the files. Each member committed at least once.

6 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]
```

7 Interacting with Flags API (For Final Submission)

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

8 Conclusion (For Final Submission)

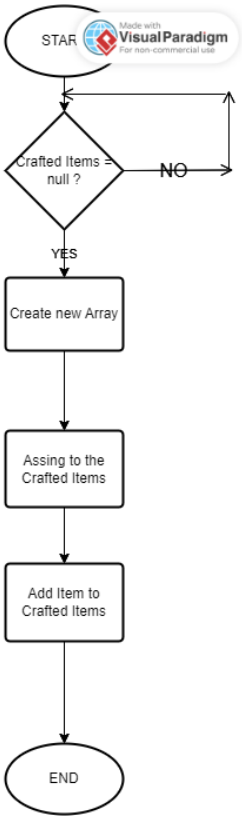
[Provide a summary of achievements, challenges, and learnings.]

9 Appendix

Include any additional pseudocode, flowcharts, or supplementary material.

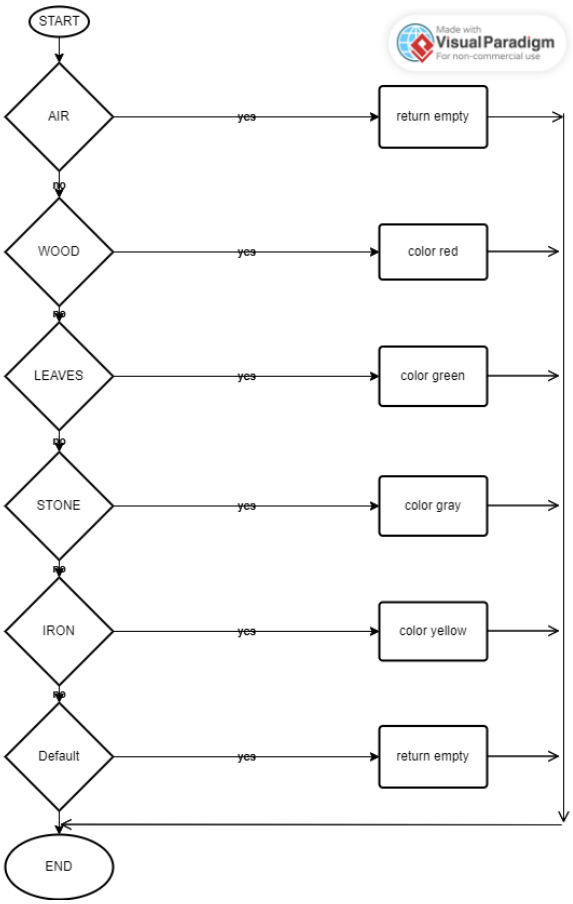
AddCraftedItems:

START
IF
Crafted items equal to null
Create new Array and assign to craftedItems
Add craftedItem to craftedItems
END



GetBlockColor:

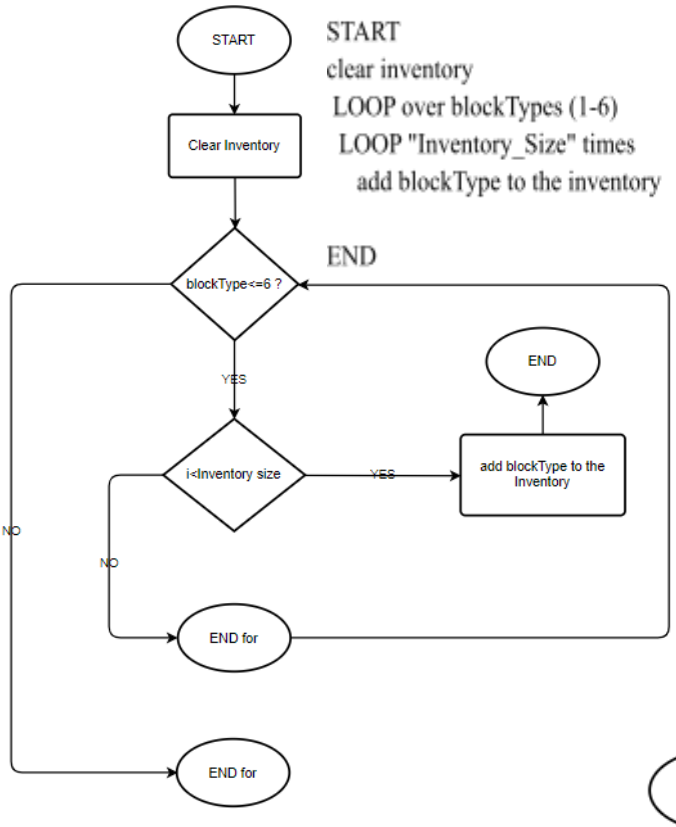
START
Switch on blockType
case AIR
return empty
case WOOD
return color RED
case LEAVES
return color GREEN
case STONE
return color GRAY
case IRON
return color YELLOW
default
return empty



Fill Inventory:

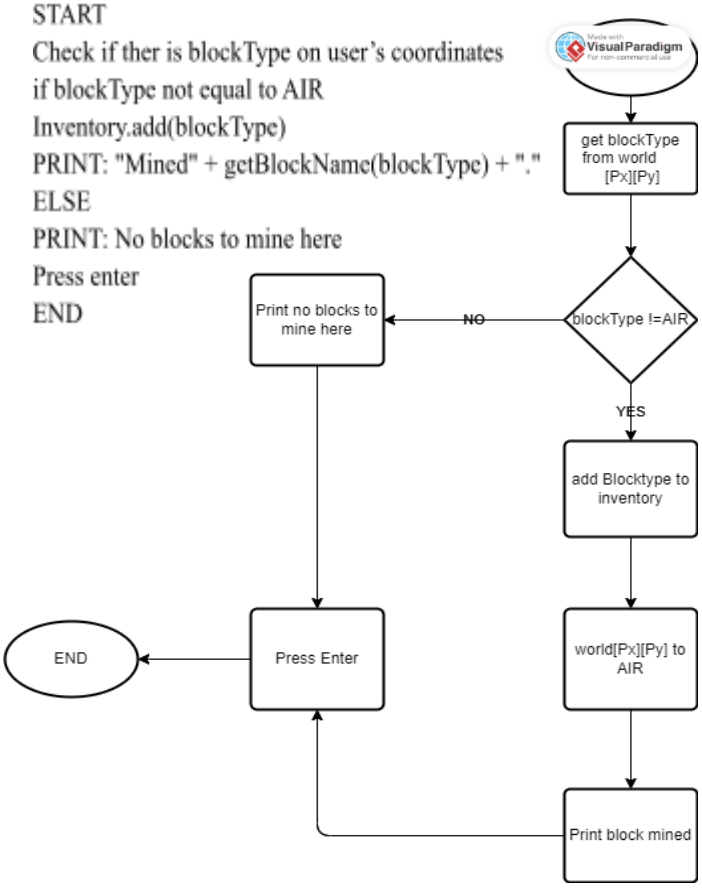
START
clear inventory
LOOP over blockTypes (1-6)
LOOP "Inventory_Size" times
add blockType to the inventory

END



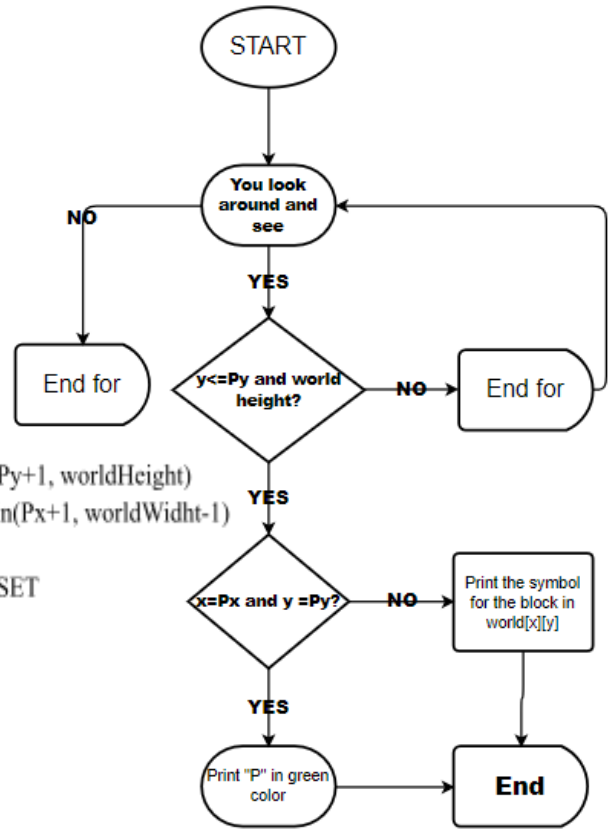
mineBlock:

START
Check if ther is blockType on user's coordinates
if blockType not equal to AIR
Inventory.add(blockType)
PRINT: "Mined" + getBlockName(blockType) + "."
ELSE
PRINT: No blocks to mine here
Press enter
END



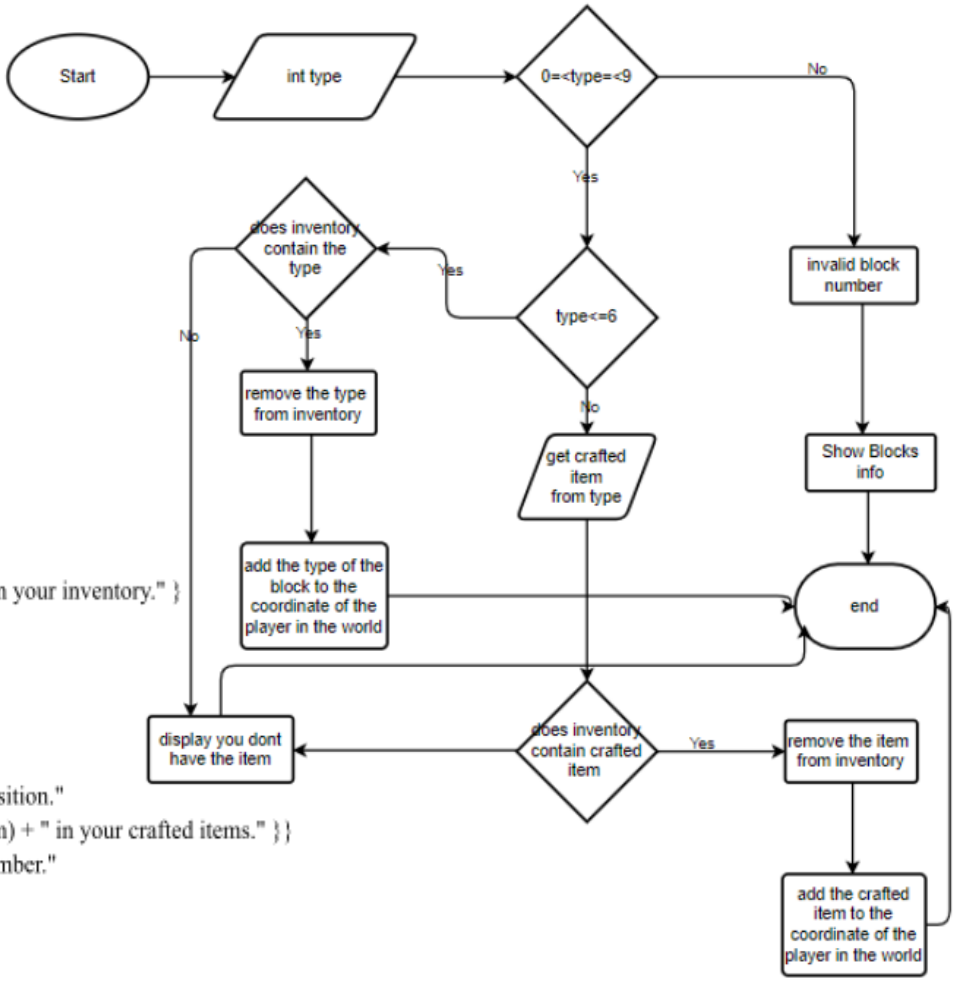
lookAround:

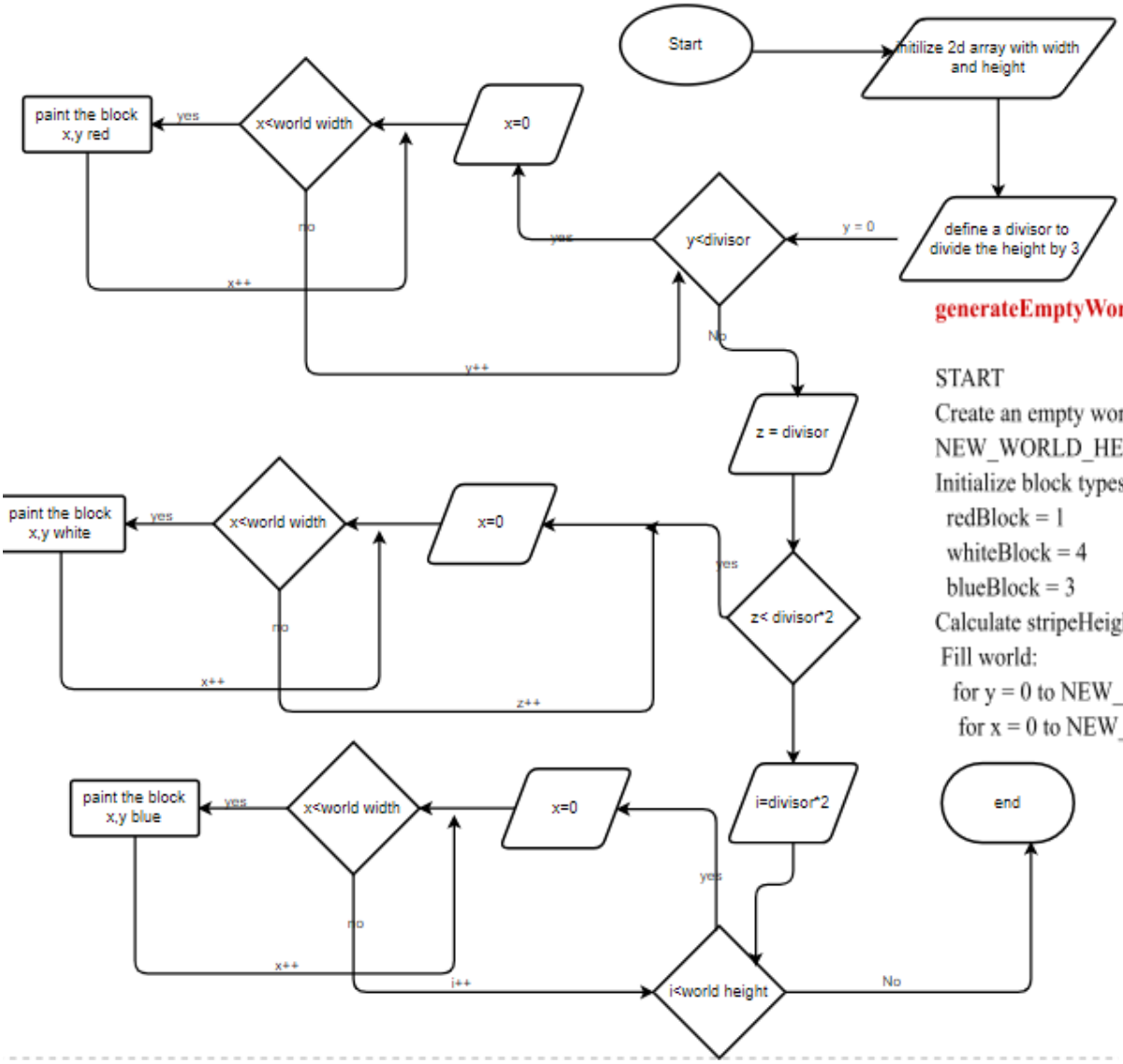
START
PRINT: You look around and see
LOOP over "y" from max(0, Py-1) to min(Py+1, worldHeight)
LOOP over "x" from max(0, Px-1) to min(Px+1, worldWidht-1)
IF x = Px and y = Py
PRINT: ANSI_GREEN + "P" + ANSI_RESET
ELSE
PRINT: getBlockSymbol(world[x][y])
END



placeBlock:

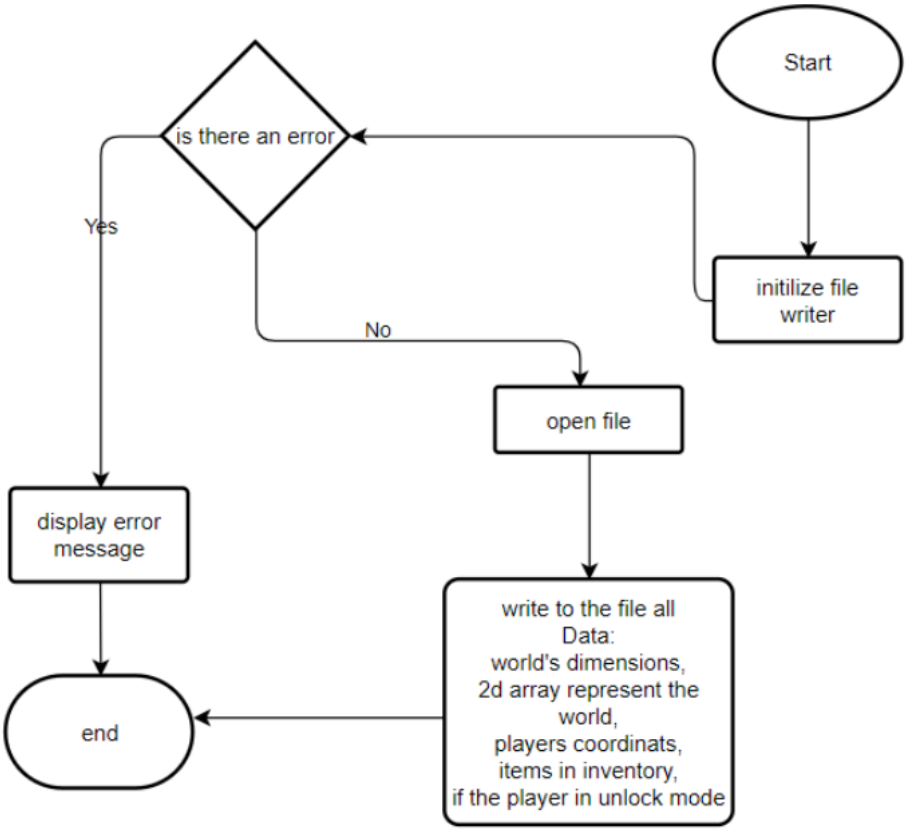
```
 if (blockType is a valid block number) {
if (blockType is a basic block) {
if (inventory contains blockType) {
 Remove blockType from inventory
Place blockType at player's position
Print "Placed " + getBlockName(blockType) + " at your position."
} else { Print "You don't have " + getBlockName(blockType) + " in your inventory." }
} else {
Calculate craftedItem from blockType
if (craftedItems contains craftedItem) {
Remove craftedItem from craftedItems
Place blockType at player's position
Print "Placed " + getCraftedItemName(craftedItem) + " at your position."
} else { Print "You don't have " + getCraftedItemName(craftedItem) + " in your crafted items." }}
} else { Print "Invalid block number. Please enter a valid block number."
Print BLOCK_NUMBERS_INFO }
```





generateEmptyWorld:

START
Create an empty world with NEW_WORLD_WIDTH and NEW_WORLD_HEIGHT
Initialize block types:
 redBlock = 1
 whiteBlock = 4
 blueBlock = 3
Calculate stripeHeight := NEW_WORLD_HEIGHT / 3
 Fill world:
  for y = 0 to NEW_WORLD_HEIGHT - 1 do
   for x = 0 to NEW_WORLD_WIDTH - 1 do
                if y < stripeHeight
                world[x][y] = redBlock
                 else if y < 2 * stripeHeight
                 world[x][y] = whiteBlock
                  else
                 world[x][y] = blueBlock
                 END

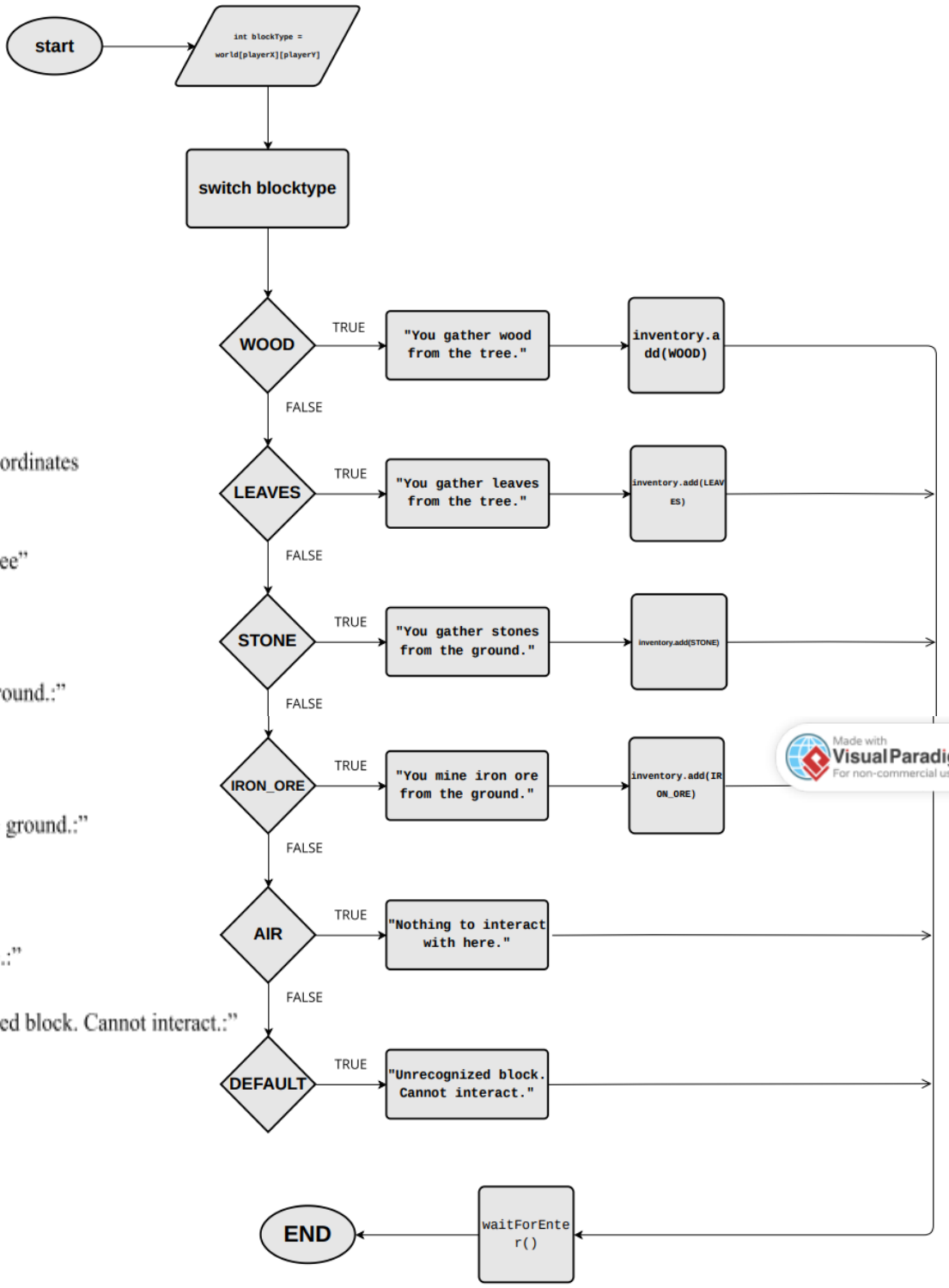


saveGame:

```
START
try
outputStream := createObjectOutputStream(fileName)
outputStream.writeInt(NEW_WORLD_WIDTH)
outputStream.writeInt(NEW_WORLD_HEIGHT)
outputStream.writeObject(world)
outputStream.writeInt(playerX)
outputStream.writeInt(playerY)
outputStream.writeObject(inventory)
outputStream.writeObject(craftedItems)
outputStream.writeBoolean(unlockMode)
Print "Game state saved to file: " + fileName
catch  IOException as e
Print "Error while saving the game state: " + e.getMessage()
end try
END
```

interactWithWorld:

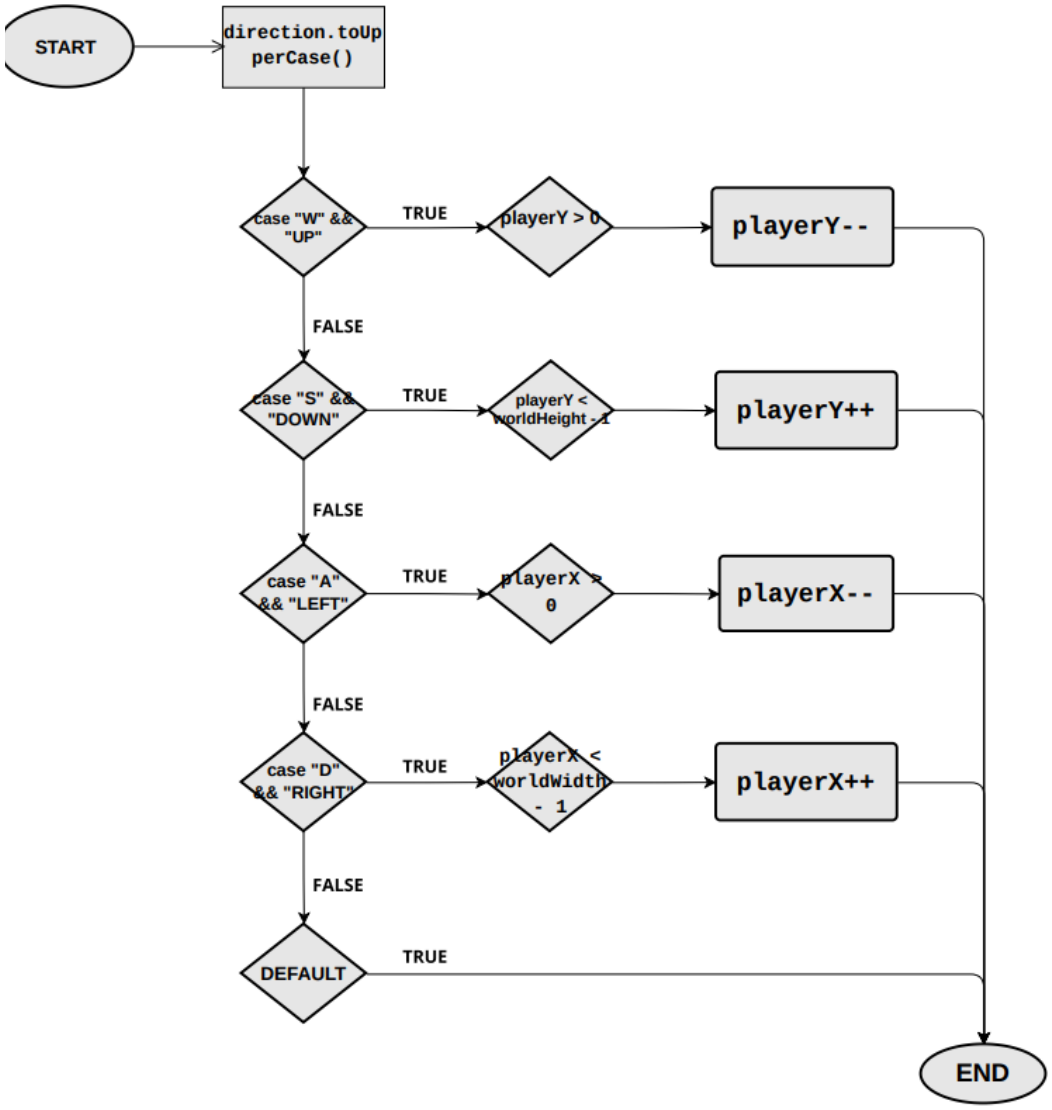
START
check for blockType on player's coordinates
switch:blockType
case WOOD:
print "you gather wood from the tree"
Add 1 Wood in inventory
break
case STONE:
print "you gather stone from the ground.:"
Add 1 Stone in inventory
break
case IRON_ORE:
print "you gather iron ore from the ground.:"
Add 1 Iron_Ore in inventory
break
case AIR:
print "Nothing to interact with here.:"
break
case DEFAULT: print "Unrecognized block. Cannot interact.:"
END



movePlayer:

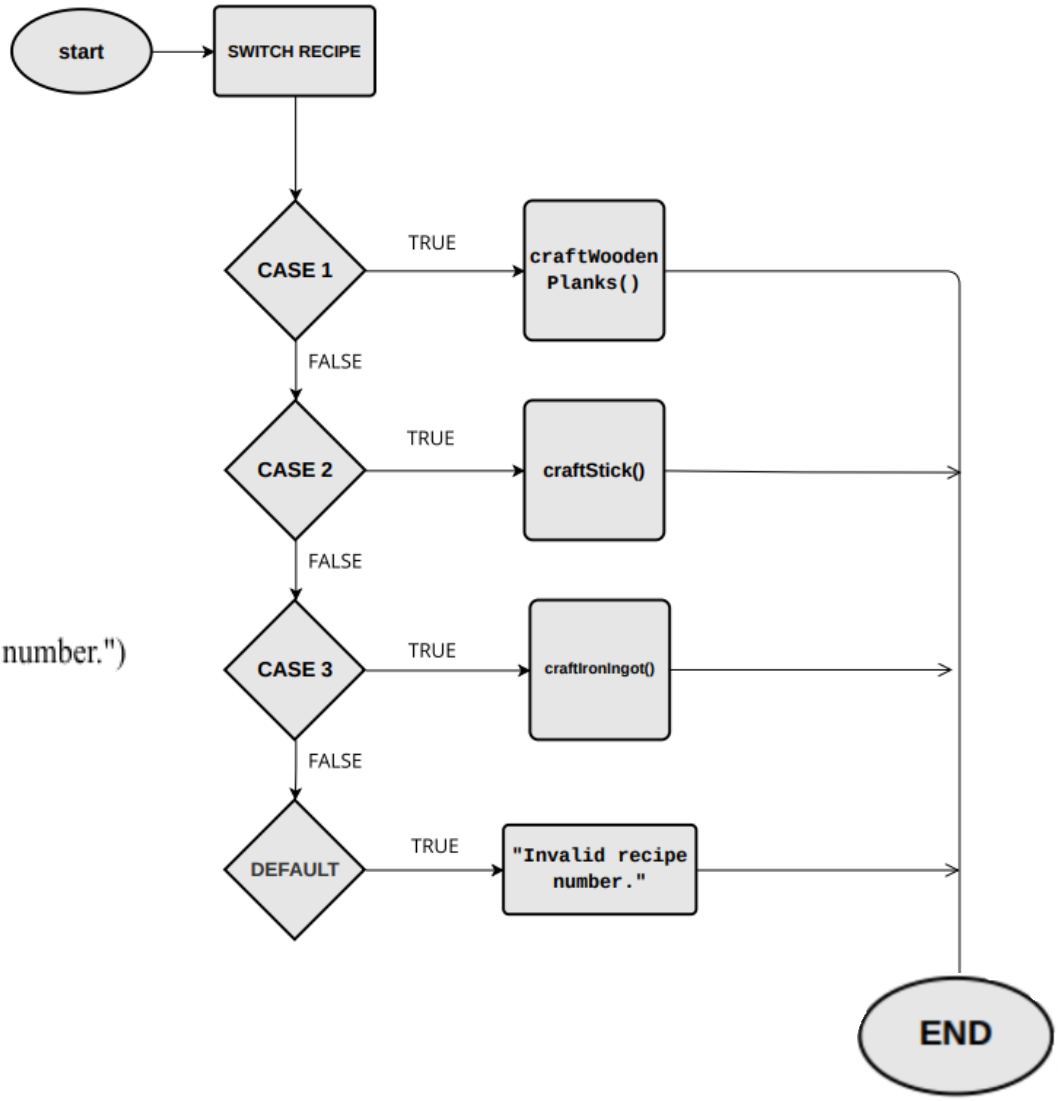
START

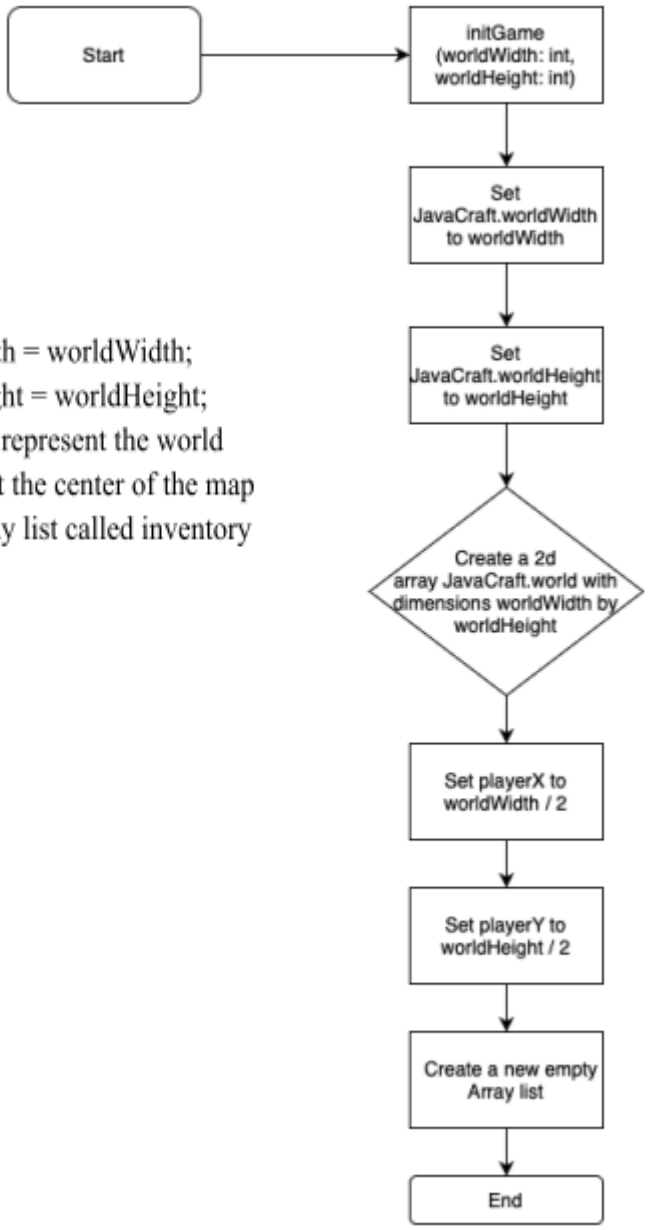
```
switch uppercasedDirection
case "W", "UP":
  if playerY > 0
   playerY = playerY - 1
  break
 case "S", "DOWN":
  if playerY < worldHeight - 1:
    playerY = playerY + 1
     break
 case "A", "LEFT":
    if playerX > 0
      playerX = playerX - 1
     break
 case "D", "RIGHT":
    if playerX < worldWidth - 1
     playerX = playerX + 1
 break
```



craftItem:

```
START
 switch recipe:
case 1:          craftWoodenPlanks()
  break
case 2:           craftStick()
  break
case 3:          craftIronIngot()
break
default:          print("Invalid recipe number.")
END
```



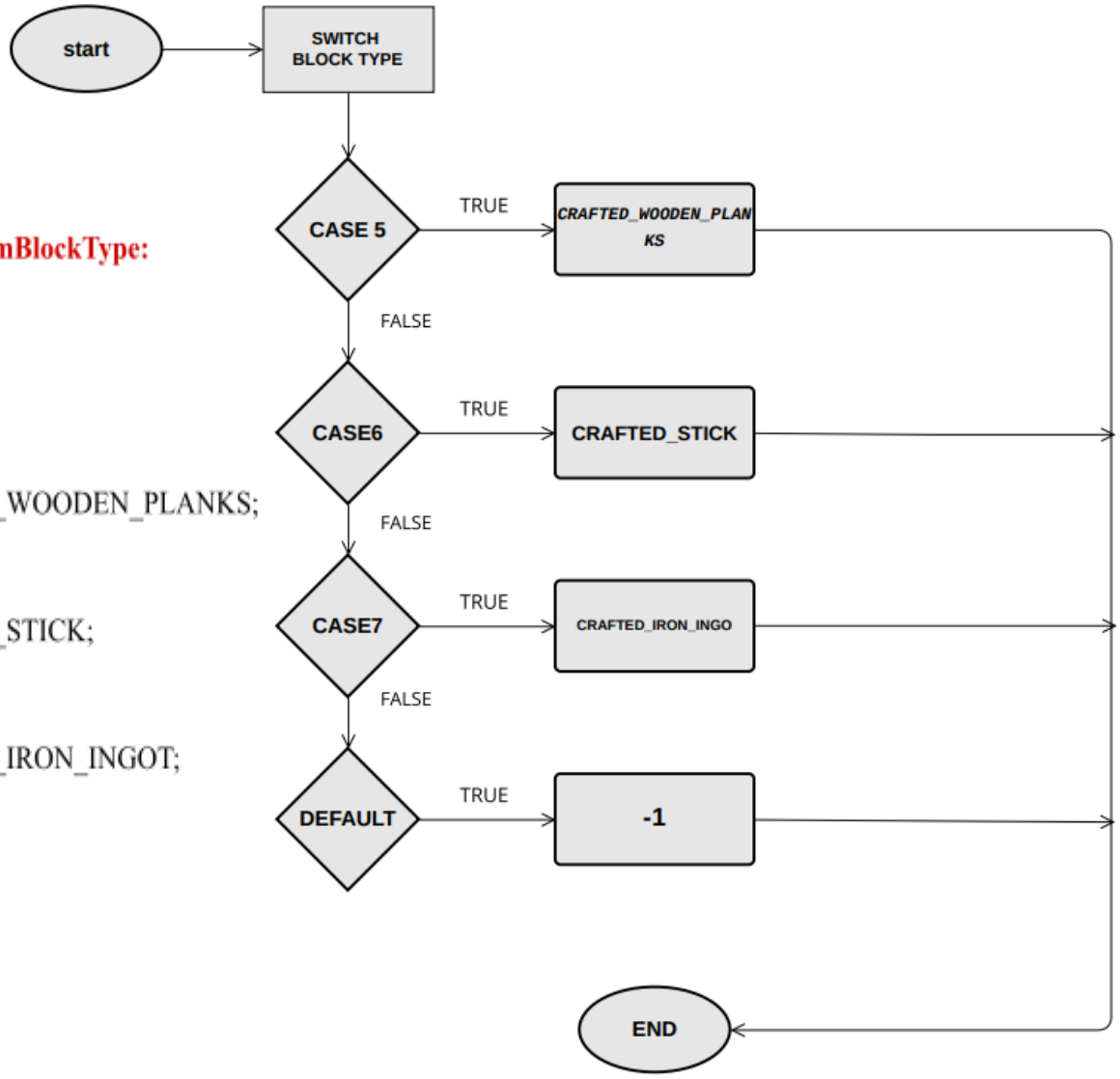


initGame:

JavaCraft.worldWidth = worldWidth;
JavaCraft.worldHeight = worldHeight;
Create a 2D array to represent the world
Position the player at the center of the map
Create an empty array list called inventory

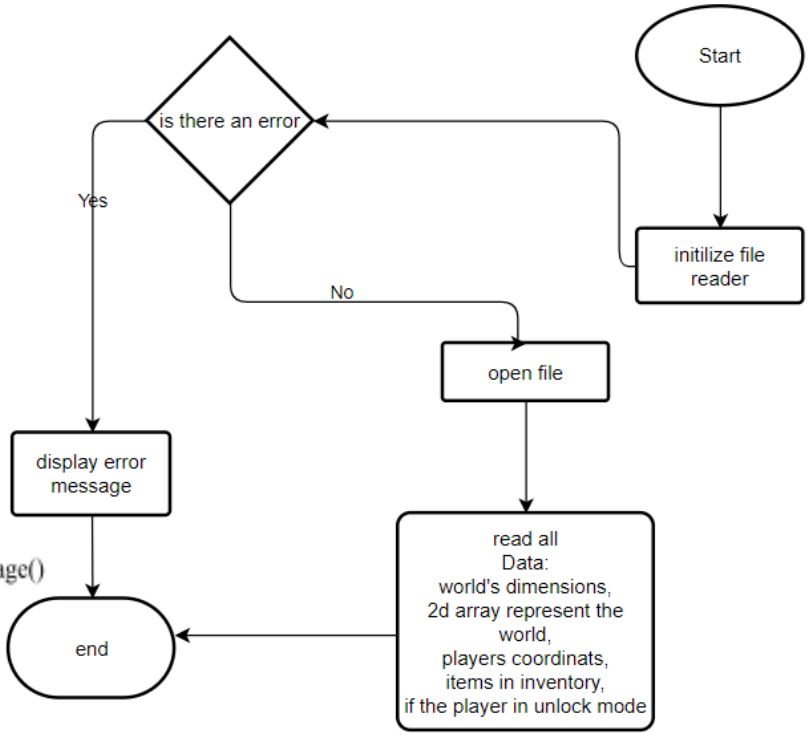
getCraftedItemFromBlockType:

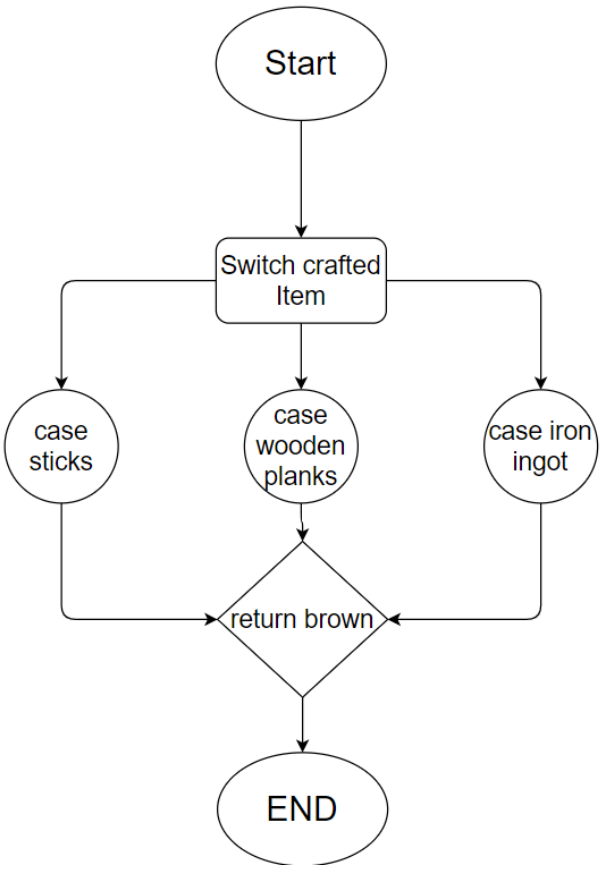
```
START
switch (blockType) {
    case 7:
   return CRAFTED_WOODEN_PLANKS;
 break
   case 8:
   return CRAFTED_STICK;
break
   case 9:
   return CRAFTED_IRON_INGOT;
break
   default:
return -1;
END
```



loadGame:

START
try
inputStream = createObjectInputStream(fileName)
inputStream.writeInt(NEW_WORLD_WIDTH)
inputStream.writeInt(NEW_WORLD_HEIGHT)
inputStream.writeObject(world)
inputStream.writeInt(playerX)
inputStream.writeInt(playerY)
inputStream.writeObject(inventory)
inputStream.writeObject(craftedItems)
inputStream.writeBoolean(unlockMode)
Print "Game state loaded from file: " + fileName
catch IOException file not found
Print "Error while loading the game state: " + e.getMessage()
end try
END





getCraftedItemColor:

START
make a switch case for every crafted item
return the color brown to all cases of crafted items
END

10 References

1. https://online.visual-paradigm.com/diagrams/features/flowchart-tool/ - For creating flowcharts.