Design Document

ISS Docking Simulation

James Robbie 2022-05-04

# Table of Contents

[**Table of Contents 2**](#_heading=h.omocq13judqa)

[**Description**](#_heading=h.so5y7r5tjaz2) **3**

[**Milestone #1 - 2/14/22 3**](#_heading=h.fr52j9j7oq81)

[**Milestone #2 - 2/28/22 6**](#_heading=h.ijgwp78peihx)

[**Milestone #3 - 3/28/22 13**](#_heading=h.9img6xao7w3a)

[**Milestone #4 - 4/18/22 16**](#_heading=h.qft9cerqkfyj)

[**Milestone #5 - 5/4/22 19**](#_heading=h.imzbgp5neioi)

[**Program Files 23**](#_heading=h.41m5uc8aa4yj)

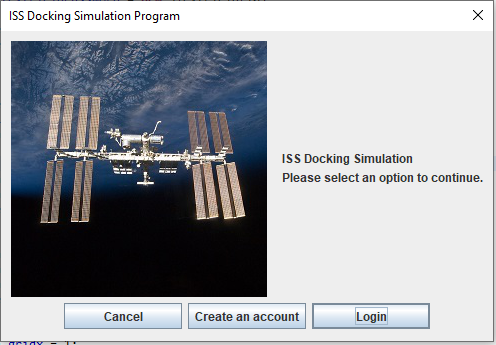
## **Description**

This is a program based in Java 1.6 in the Eclipse IDE. The goal of this project was to create a game based on a hypothetical docking procedure for the ISS Space Station. The user starts with a random approach distance, thus having a limited number of inputs.

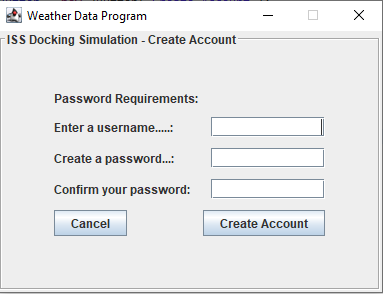
The user must use these limited options to “align” their vessel to the ISS Dock. There will be a feedback window showing how close the vessel is to alignment. If alignment is achieved before the approach distance runs out, the game is won.

## Milestone #1 - 2/14/22

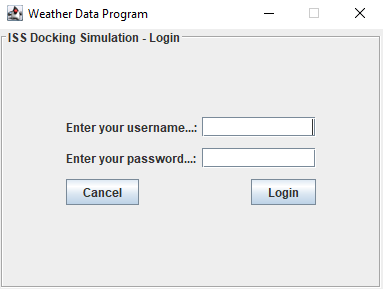
The goal of Milestone 1 is to develop an initial gui of the program as well as create a three option panel with the choices: Cancel, Create Account, and Login. Each of these buttons are meant to be functional, with Create Account and Login bringing up another panel respectively, and Cancel closing the programs.



Upon first launching the program, an interface is brought up with the options: Cancel, Create an account, and Login.



If the user selects “Create an account” a window will be brought up. Password requirements is present on the panel but nothing on it; will be updated later. The buttons are currently non-functional.



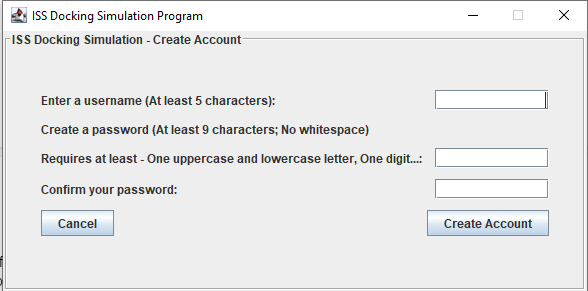
If the user selects “Login” the login window will be brought up. The buttons are non-functional.

Issues to fix:

* Update password requirements
* Make the buttons on Login and Create Account functional
* Make the “Cancel” buttons on the create account and login panels bring you back to the initial options menu.

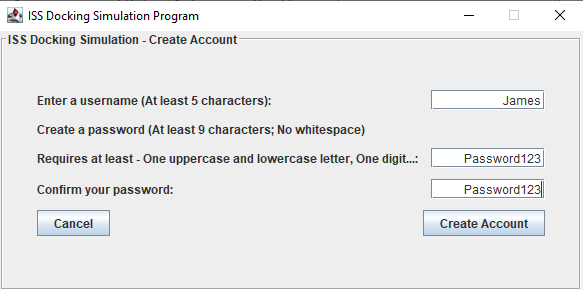
## Milestone #2 - 2/28/22

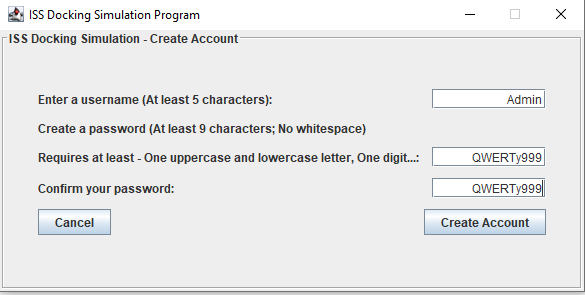
The goal of Milestone #2 is to develop a functional Create Account and Login sequence that allows the user to create an account and have the credentials stored for future logins. It is building upon the initial GUI created in Milestone 1. The GUI of the main interface of the program is created as well.

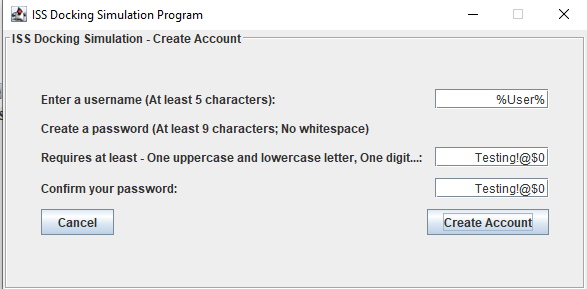


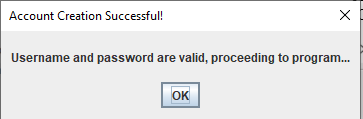
Updated Account Creation Window

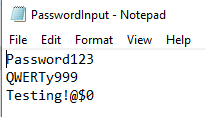
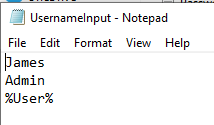
Successful Account Creation is Written to Username and Password Files





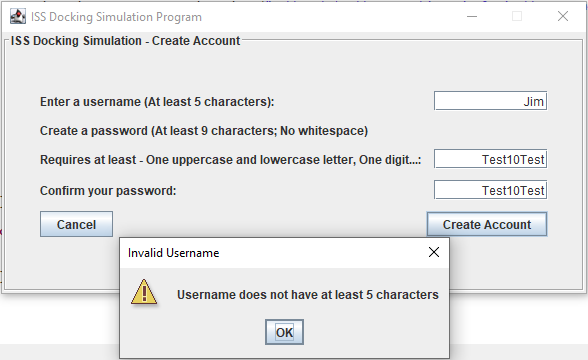


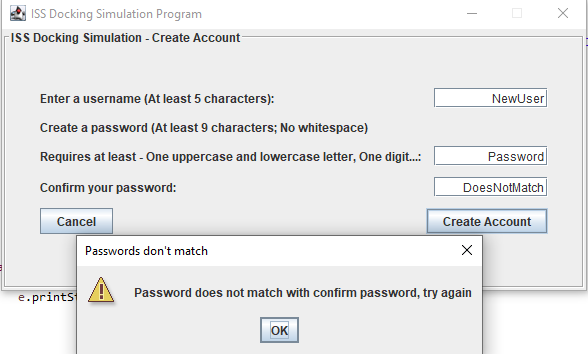


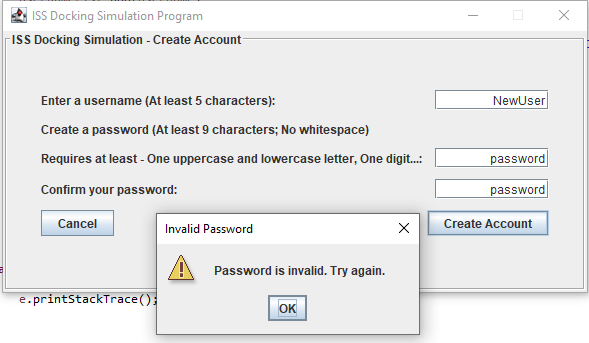


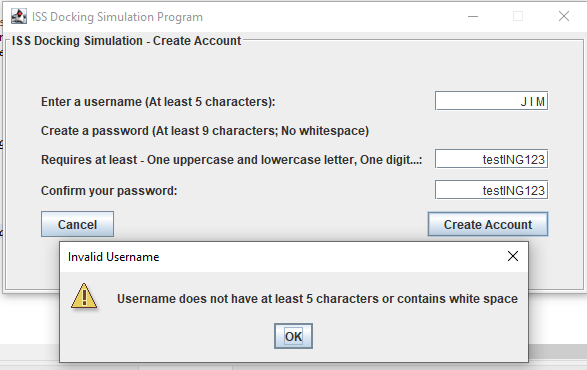
UsernameInput.txt and PasswordInput.txt

Invalid Input Catching

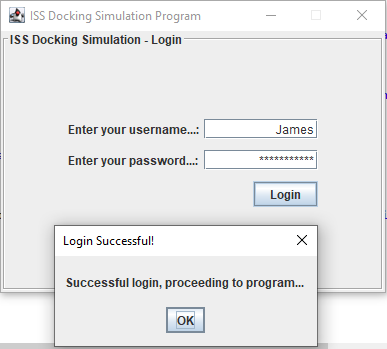




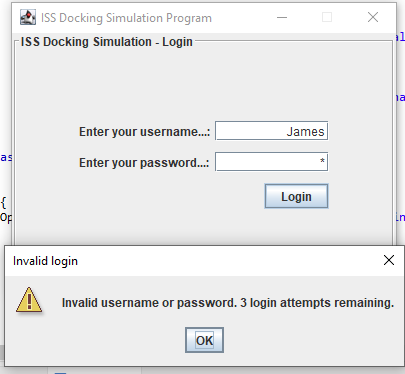




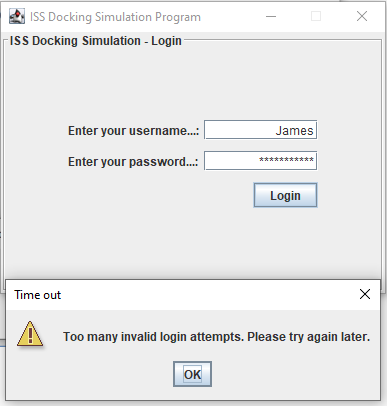
If invalid input is caught, Username and Password are not written to file



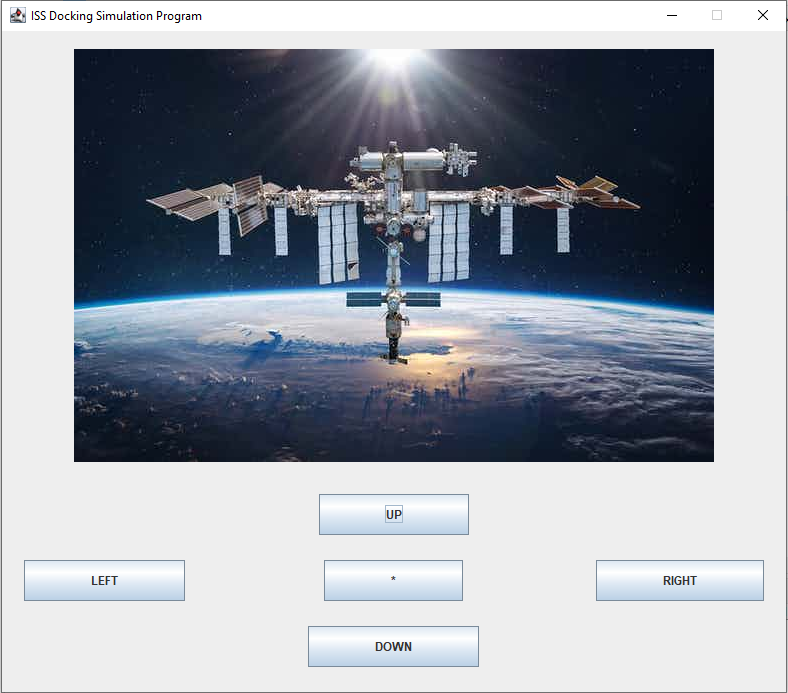
Login Operation



Login Attempts are counted



User is timed out if login attempts are exceeded



On Successful Login or Account Creation, User is taken to program. Currently non-functional.

Planning Layout

Create Account Pseudocode Checklist:

* User presses create account (**X**)
* Action listener is triggered (**X**)
* Program reads if the password is at least 9 characters and has at least an Uppercase letter, lowercase letter, and digit. (**X**)
  + If input is invalid, error message will be displayed (**X**)
  + If password i already in use, error message will be displayed (**X**)
* Username and password file is opened (**X**)
* Username file is written to username array (**X**)
* Password file is written to password array (**X**)
* User inputted Username is added to username array (**X**)
* User Inputted Password is written to password array (**X**)
* Username array is written to username file (**X**)
* Password array is written to password file (**X**)
* Success window is displayed to user (**X**)
* Program advances to ISS Simulation (**X**)

Login Pseudocode Checklist

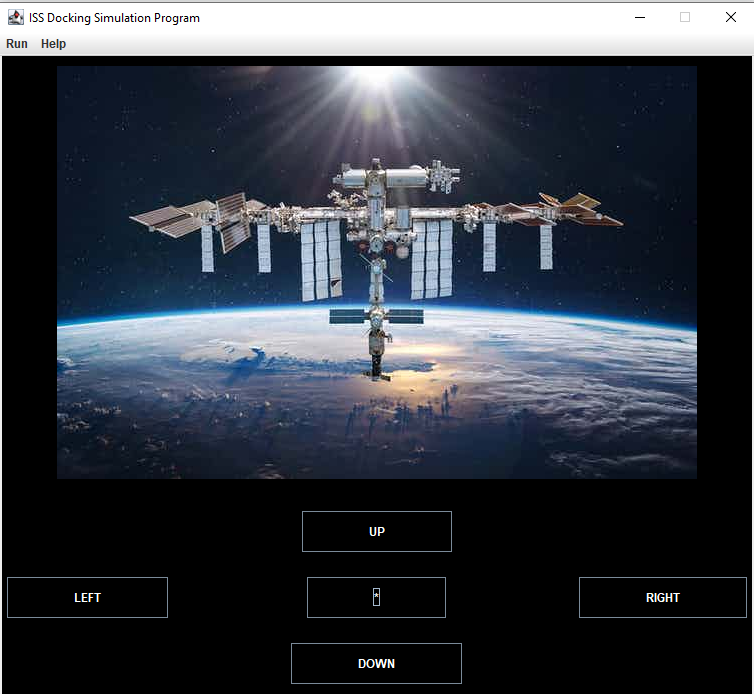
* User presses create account and action listener is triggered (**X**)
* Programs reads password file (**X**)
* When password is found in file, the line number is stored to a variable (**X**)
  + If password is not found, display invalid Login (**X**)
* Open username file (**X**)
* Go to line number and see if Username matches Login info (**X**)
  + If username in file does not match User’s login, display invalid login (**X**)
* Success window is displayed to user (**X**)
* Program advances to ISS simulation (**X**)

Issues to fix:

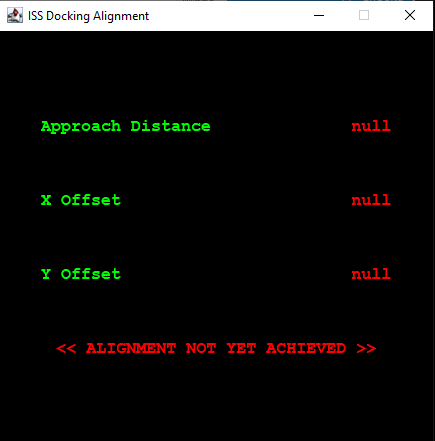
* Make Program GUI Look Better
* Change background color of program
* Possibly align the buttons better
* Make Login and Create Account windows close when program launches

## Milestone #3 - 3/28/22

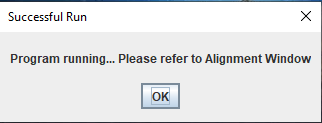
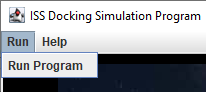
The goal of Milestone 3 is to develop a display window to show the Alignment values to the user, as well as develop functionality to the random number generators for the values.



Updated main panel GUI



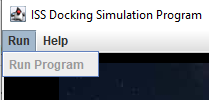
Alignment Window is now created with Main Panel



JMenu is used to run program

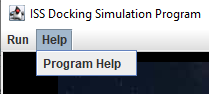


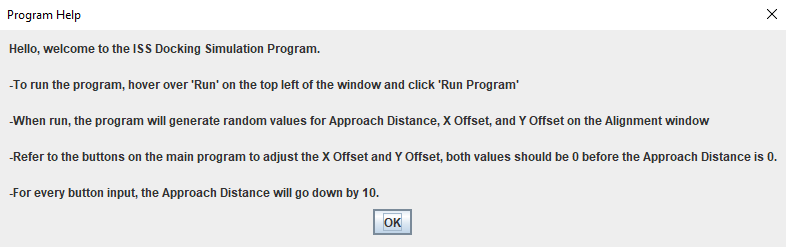
Random values are generated and updated on run



Run Button is disabled while program is running

A Help Button is integrated if needed





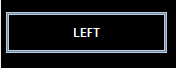
Functionality Pseudocode Checklist

* Create alignment information window (**X**)
* Make run program Jmenu item functional (**X**)
* Generate random approach distance between 100 and 250 meters (**X**)
* Display approach distance (**X**)
* Generate random x off center between -20 and 20 meters (**X**)
* Display random x off center (**X**)
* Generate random y off center between -15 and 15 meters (**X**)
* Display random y off center (**X**)
* Add functionality to Help Jmenu (**X**)

## Milestone #4 - 4/18/22

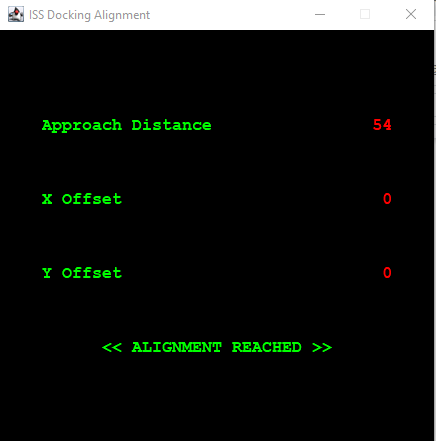
The goal of Milestone #4 is to give functionality to the buttons created in Milestone #2 as well as apply them to the Alignment Window created in Milestone #3.

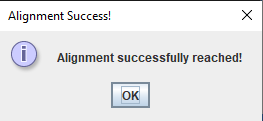






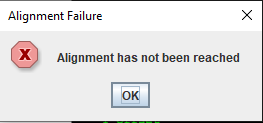
When a button is pressed, the alignment window updates.



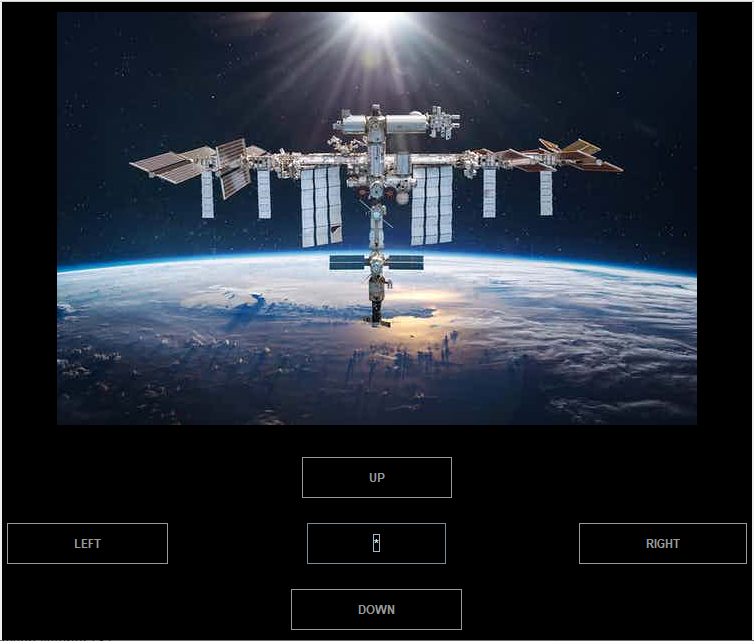


If the user is able to “align” the X Offset and Y Offset to 0 before approach distance reaches 0, the program prompts that alignment has been reached and stops.

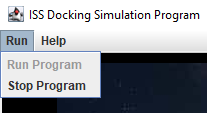




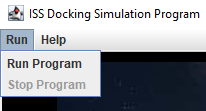
If the user isn’t able to align, the program prompts that the user has failed and stops.



Buttons are disabled when program is stopped



Users can manually stop the program before any conditions are met.



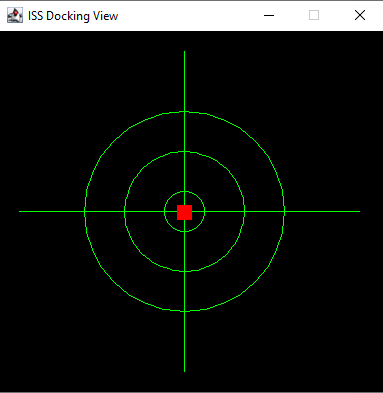
When the program is stopped, Run is enabled and Stop is disabled, and when program is running, Stop is enabled and Run is disabled.

Pseudocode Checklist

* Up button has action listener (**X**)
* Up button adds 1 to Y alignment and subtracts 10 from Approach distance (**X**)
* Down button has action listener (**X**)
* Down button subtracts 1 from Y alignment and subtracts 10 from Approach distance (**X**)
* Left button has action listener (**X**)
* Left button subtracts 1 from X alignment and subtracts 10 from Approach distance (**X**)
* Right button has action listener (**X**)
* Up button adds 1 to Y alignment and subtracts 10 from Approach distance (**X**)
* If alignment has been achieved, a success window will appear. (**X**)
* If approach distance has reached 0 and alignment hasn’t been achieved, a failure window will appear. (**X**)
* If success or failure has been achieved, all buttons will be disabled, and the Run Program JMenu will reenable. (**X**)
* Make buttons enabled on run. (**X**)

## Milestone #5 - 5/4/22

The goal of Milestone #5 is to create a functional display GUI that updates with every button input. This GUI will help give clarity to the user as to where they are in the simulation.



Display Window



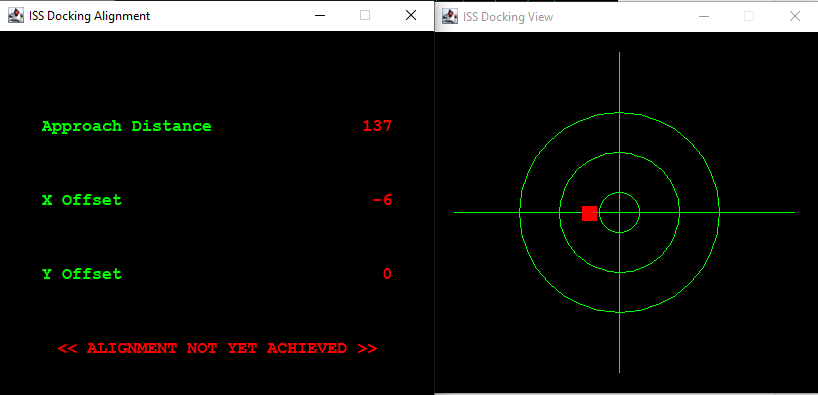




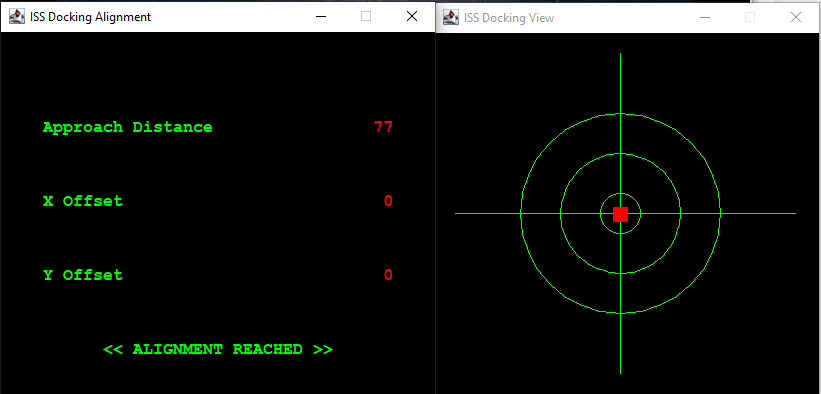
The Display Window matches the Alignment



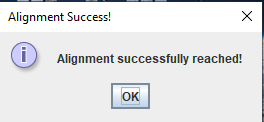






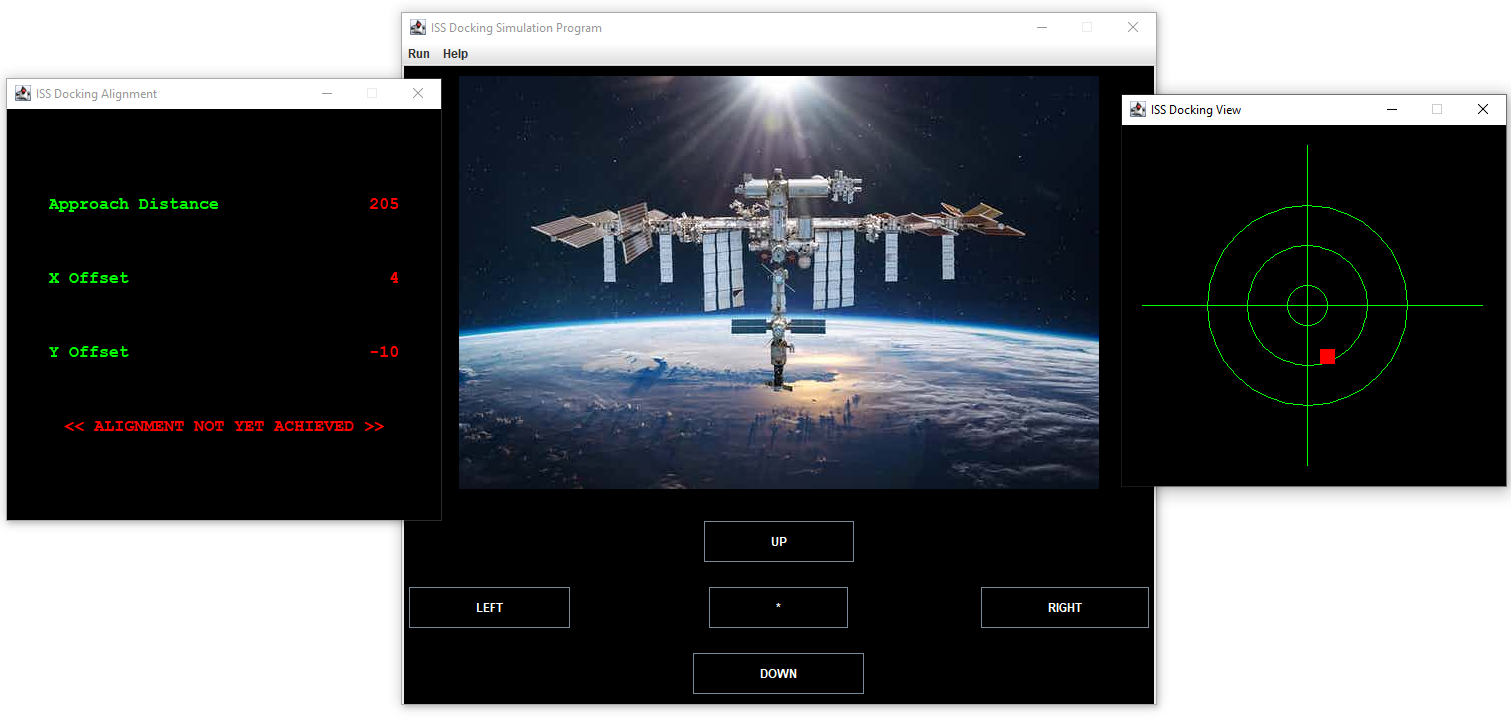


Display window updates with every button input





Alignment is reached when box is at (0,0)



The Final Program

## Program Files

**//Main.java**

// James Robbie ISS Docking Simulation Main

package issPackage;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

int selection = 0;

selection = BoxFunctions.guiInitialDialogBox();

if (selection == 1) {

new CreateAccount();

}

else if (selection == 2) {

new Login();

//new SimulationBox(); // FOR TESTING THE SIMULATION AND ALIGNMENT BOXES

//new AlignmentWindow();

}

else {

System.out.println("Cancel.");

}

// System.out.println(selection);

}

}

**// BoxFunctions.java**

package issPackage;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

public class BoxFunctions {

public static void guiAccess(){

System.out.println("GUI.java successfully accessed.");

}

public static void guiLoginFrame() {

JFrame loginFrame = new JFrame();

loginFrame.setTitle("ISS Docking Simulation Login");

loginFrame.setSize(600,500);

loginFrame. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

loginFrame.setVisible(true);

}

public static int guiInitialDialogBox() {

Object[] options = {"Cancel", "Create an account", "Login"};

int selection = JOptionPane.showOptionDialog(null,

"ISS Docking Simulation\nPlease select an option to continue.",

"ISS Docking Simulation Program",

JOptionPane.YES\_NO\_CANCEL\_OPTION,

JOptionPane.QUESTION\_MESSAGE,

new ImageIcon("E:\\Workshop\\ISS Docking Simulation\\iss-icon.png"),

options,

options[2]);

// System.out.println(selection);

return selection;

}

public static boolean usernameValid(String username) {

int index = 0, whiteSpace = 0;

while (index < username.length()) {

char ch = username.charAt(index);

// Checks if user inputted whitespace

if (Character.isWhitespace(ch)) {

whiteSpace++;

}

index++;

}

if (whiteSpace == 0 && username.length() >= 5) {

return true;

}

else {

return false;

}

}

public static boolean passwordValid(String password) {

int index = 0, digit = 0, upper = 0, lower = 0, whiteSpace = 0;

while (index < password.length()) {

char ch = password.charAt(index);

// If statements to see the number of Digits and Uppercase/Lowercase letters

if (Character.isDigit(ch)) {

digit++;

}

if (Character.isUpperCase(ch)) {

upper++;

}

if (Character.isLowerCase(ch)) {

lower++;

}

if (Character.isWhitespace(ch)) {

whiteSpace++;

}

index++;

}

// Test functions to see if password is being properly read

//System.out.println("Digits " + digit); // displays Digits 3

//System.out.println("Uppercase " + upper); // displays Uppercase 3

//System.out.println("Lowercase " + lower); // displays Lowercase 3

if (whiteSpace > 0) {

return false;

}

else if (password.length() >= 9 && digit >= 1 && upper >= 1 && lower >= 1) {

return true;

}

else {

return false;

}

}

}

**// CreateAccount.java**

package issPackage;

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.Insets;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class CreateAccount extends JFrame {

JFrame createAcctGUI = new JFrame("ISS Docking Simulation Program");

JPanel createAcctPanel = new JPanel(new GridBagLayout());

JLabel passwordInfoLabel = new JLabel("Requires at least - One uppercase and lowercase letter, One digit...: ");

JLabel userNameLabel = new JLabel("Enter a username (At least 5 characters): ");

JLabel passWordLabel1 = new JLabel("Create a password (At least 9 characters; No whitespace)");

JLabel passWordLabelConfirm = new JLabel("Confirm your password: ");

JTextField textFieldUserName = new JTextField(10);

JTextField textFieldPassword = new JTextField(10);

JTextField textFieldConfirmPassword = new JTextField(10);

private JButton createAcctButton = new JButton("Create Account");

JButton createCancelButton = new JButton("Cancel");

public CreateAccount(){

createAcctGUI.setSize(600,300);

GridBagConstraints con = new GridBagConstraints();

con.insets = new Insets(1,1,10,1);

// Label for Username

con.gridx = 0;

con.gridy = 0;

con.anchor = GridBagConstraints.WEST;

createAcctPanel.add(userNameLabel, con);

// Text field for Username

con.gridx = 1;

con.anchor = GridBagConstraints.EAST;

createAcctPanel.add(textFieldUserName, con);

textFieldUserName.setHorizontalAlignment(JTextField.RIGHT);

// Label for Password

con.gridx = 0;

con.gridy = 1;

con.anchor = GridBagConstraints.WEST;

createAcctPanel.add(passWordLabel1, con);

// Password Requirements Label

con.gridx = 0;

con.gridy = 2;

createAcctPanel.add(passwordInfoLabel, con);

// Text field for password

con.gridx = 1;

con.anchor = GridBagConstraints.EAST;

createAcctPanel.add(textFieldPassword, con);

textFieldPassword.setHorizontalAlignment(JTextField.RIGHT);

// Label for Confirm Password

con.gridx = 0;

con.gridy = 3;

con.anchor = GridBagConstraints.WEST;

createAcctPanel.add(passWordLabelConfirm, con);

// Text field for Confirm Password

con.gridx = 1;

con.anchor = GridBagConstraints.EAST;

createAcctPanel.add(textFieldConfirmPassword, con);

textFieldConfirmPassword.setHorizontalAlignment(JTextField.RIGHT);

// Create Account Button

con.gridx = 1;

con.gridy = 4;

con.anchor = GridBagConstraints.EAST;

createAcctPanel.add(createAcctButton, con);

// Titled Border

createAcctPanel.setBorder(BorderFactory.createTitledBorder(

BorderFactory.createEtchedBorder(), "ISS Docking Simulation - Create Account"));

// JPanel Settings

createAcctGUI.add(createAcctPanel);

createAcctGUI.setResizable(false);

createAcctGUI.setLocationRelativeTo(null);

createAcctGUI.setVisible(true);

createAcctGUI.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

createAcctButton.addActionListener(new CreateAcctButtonListener());

}

private class CreateAcctButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

// Declare user inputted strings

String username = textFieldUserName.getText();

String password = textFieldPassword.getText();

String confirmPassword = textFieldConfirmPassword.getText();

System.out.println(password);

// Checks if username is valid

boolean validUsername = BoxFunctions.usernameValid(username);

// If the username is 5 characters and has no whitespace, program proceeds

if(validUsername == true) {

// Checks if Confirm Password matches with Password.

if(confirmPassword.equals(password)) {

// Call boolean variables and functions to check if password is valid.

boolean validPassword = BoxFunctions.passwordValid(password);

boolean duplicatePassword = FileReading.validateAndWriteToPassword(password);

// If password is valid and is not a duplicate, program proceeds

if (validPassword == true && duplicatePassword == false) {

System.out.println("The password is valid");

FileReading.writeToUsername(username);

JOptionPane.showMessageDialog(null, "Username and password are valid, proceeding to program...", "Account Creation Successful!", JOptionPane.PLAIN\_MESSAGE);

createAcctGUI.dispose();

new SimulationBox();

new AlignmentWindow();

new GraphicsWindow();

}

// 4 Else statements to find what error the password has and display error message accordingly

else if (validPassword == false) {

System.out.println("The password is invalid.");

JOptionPane.showMessageDialog(null, "Password is invalid. Try again.", "Invalid Password", JOptionPane.WARNING\_MESSAGE);

}

else if (duplicatePassword == true) {

System.out.println("The password is a duplicate.");

JOptionPane.showMessageDialog(null, "Password is already in use, enter a new password", "Duplicate Password", JOptionPane.WARNING\_MESSAGE);

}

}

else {

JOptionPane.showMessageDialog(null, "Password does not match with confirm password, try again", "Passwords don't match", JOptionPane.WARNING\_MESSAGE);

}

}

else {

JOptionPane.showMessageDialog(null, "Username contains white space or is not a least 5 characters", "Invalid Username", JOptionPane.WARNING\_MESSAGE);

}

}

}

}

**// Login.java**

package issPackage;

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.Insets;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class Login extends JFrame {

JFrame LoginGUI = new JFrame("ISS Docking Simulation Program");

JPanel LoginPanel = new JPanel(new GridBagLayout());

JLabel userNameLabel = new JLabel("Enter your username...: ");

JLabel passWordLabel1 = new JLabel("Enter your password...: ");

JTextField textFieldUserName = new JTextField(10);

JPasswordField textFieldPassword = new JPasswordField(10);

JButton LoginButton = new JButton("Login");

//JButton createCancelButton = new JButton("Cancel");

int passwordAttempts = 0;

public Login() {

LoginGUI.setSize(400,300);

GridBagConstraints con = new GridBagConstraints();

con.insets = new Insets(1,1,10,1);

// Label for Username

con.gridx = 0;

con.gridy = 0;

con.anchor = GridBagConstraints.WEST;

LoginPanel.add(userNameLabel, con);

// Text field for Username

con.gridx = 1;

con.anchor = GridBagConstraints.EAST;

LoginPanel.add(textFieldUserName, con);

textFieldUserName.setHorizontalAlignment(JTextField.RIGHT);

// Label for Password

con.gridx = 0;

con.gridy = 1;

con.anchor = GridBagConstraints.WEST;

LoginPanel.add(passWordLabel1, con);

// Text field for password

con.gridx = 1;

con.anchor = GridBagConstraints.EAST;

LoginPanel.add(textFieldPassword, con);

textFieldPassword.setHorizontalAlignment(JTextField.RIGHT);

textFieldPassword.setEchoChar('\*');

// Cancel Button

//con.gridx = 0;

//con.gridy = 2;

//con.anchor = GridBagConstraints.WEST;

//LoginPanel.add(createCancelButton, con);

// Login Button

con.gridx = 1;

con.gridy = 2;

con.anchor = GridBagConstraints.EAST;

LoginPanel.add(LoginButton, con);

// Titled Border

LoginPanel.setBorder(BorderFactory.createTitledBorder(

BorderFactory.createEtchedBorder(), "ISS Docking Simulation - Login"));

// JPanel Settings

LoginGUI.add(LoginPanel);

LoginGUI.setResizable(false);

LoginGUI.setLocationRelativeTo(null);

LoginGUI.setVisible(true);

LoginGUI.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Declare Login Button Listener

LoginButton.addActionListener(new LoginButtonListener());

}

private class LoginButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

// User is not timed out.

if (passwordAttempts < 5) {

// Variable for printing remaining attempts.

int remainingAttempts = 4 - passwordAttempts;

// Declare Username and Password Variables

String username = textFieldUserName.getText();

char[] password = textFieldPassword.getPassword();

// Assigns the line number of the password in the file.

int lineNum = FileReading.checkLoginPassword(password);

// Checks Username at assigned line number to see if it matches.

boolean validLogin = FileReading.checkLoginUsername(lineNum, username);

// Valid Login

if (validLogin == true) {

JOptionPane.showMessageDialog(null, "Successful login, proceeding to program...", "Login Successful!", JOptionPane.INFORMATION\_MESSAGE);

LoginGUI.dispose();

new SimulationBox();

new AlignmentWindow();

new GraphicsWindow();

}

// Invalid Login

else {

if (passwordAttempts == 4) {

JOptionPane.showMessageDialog(null, "Too many invalid login attempts. Please try again later.", "Time out", JOptionPane.ERROR\_MESSAGE);

}

else {

JOptionPane.showMessageDialog(null, "Invalid username or password. " + remainingAttempts + " login attempt(s) remaining.", "Invalid login", JOptionPane.WARNING\_MESSAGE);

}

}

passwordAttempts++;

}

// User is timed out.

else {

JOptionPane.showMessageDialog(null, "Too many invalid login attempts. Please try again later.", "Time out", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

**// FileReading.java**

package issPackage;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Scanner;

public class FileReading {

final static String USERNAME\_FILE\_PATH = "UsernameInput.txt";

final static String PASSWORD\_FILE\_PATH = "PasswordInput.txt";

// Checks if Password is valid. If so, it will write the password to the file.

public static boolean validateAndWriteToPassword(String password) {

File inputFile = new File(PASSWORD\_FILE\_PATH);

boolean duplicate = false;

ArrayList<String> passwordList = new ArrayList<String>();

try { // Try block

Scanner in = new Scanner(inputFile);

//PrintWriter out = new PrintWriter("E:\\Workshop\\ISS Docking Simulation\\PasswordInput.txt");

// While statement to read each line of the file

while (in.hasNextLine()) {

String line = in.nextLine();

//System.out.println(line);

passwordList.add(line);

// Finds if the line is the same as the password

if (line.equals(password)) {

duplicate = true;

}

}

// Adds password to Array if it is not a duplicate

if (duplicate == false) {

passwordList.add(password);

}

// Declare printwriter

PrintWriter out = new PrintWriter(PASSWORD\_FILE\_PATH);

// Reads each line of the array and writes to the file

for (String name : passwordList) {

//System.out.println(name);

out.println(name);

}

// Close Scanner and Printwriter

in.close();

out.close();

}

catch (FileNotFoundException e) { // catch block

System.out.println("File cannot be found");

e.printStackTrace();

}

//System.out.println(duplicate);

if (duplicate == false) {

return false;

}

else {

return true;

}

}

// Writes username to file.

public static void writeToUsername(String username) {

File inputFile = new File(USERNAME\_FILE\_PATH);

ArrayList<String> usernameList = new ArrayList<String>();

try { // Try block

Scanner in = new Scanner(inputFile);

// While statement to read each line of the file

while (in.hasNextLine()) {

String line = in.nextLine();

usernameList.add(line);

}

// Adds username to Array

usernameList.add(username);

// Declare printwriter

PrintWriter out = new PrintWriter(USERNAME\_FILE\_PATH);

// Reads each line of the array and writes to the file

for (String name : usernameList) {

//System.out.println(name);

out.println(name);

}

// Close Scanner and Printwriter

in.close();

out.close();

}

catch (FileNotFoundException e) { // catch block

System.out.println("File cannot be found");

e.printStackTrace();

}

}

// Checks if password is valid

public static int checkLoginPassword(char[] password) {

File inputFile = new File(PASSWORD\_FILE\_PATH);

// Convert password char array to string.

String passwordString = new String(password);

int index = 0;

try {

Scanner in = new Scanner(inputFile);

// Read file.

while (in.hasNextLine()) {

String line = in.nextLine();

if (line.equals(passwordString)) { // If matching, returns the line number of the password in the file

return index;

}

index++;

}

in.close();

}

catch (FileNotFoundException e) { // catch block

System.out.println("File cannot be found");

e.printStackTrace();

}

// Return an impossible line value if no matching password is found.

return -1;

}

// Checks if Username matches with the Password's line number

public static boolean checkLoginUsername(int lineNum, String username) {

File inputFile = new File(USERNAME\_FILE\_PATH);

int index = 0;

try {

Scanner in = new Scanner(inputFile);

// Reads file and finds if username is matching with the login information at the correct Line Number.

while (in.hasNextLine()) {

String line = in.nextLine();

if (index == lineNum) {

if (line.equals(username)) {

return true;

}

}

index++;

}

in.close();

}

catch (FileNotFoundException e) { // catch block

System.out.println("File cannot be found");

e.printStackTrace();

}

return false;

}

}

**// SimulationBox.java**

package issPackage;

import java.awt.Color;

import java.awt.Component;

import java.awt.Dimension;

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.Insets;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import javax.swing.event.MenuEvent;

import javax.swing.event.MenuListener;

public class SimulationBox extends JFrame {

// Create Frame and Panel

JFrame SimulationBox = new JFrame("ISS Docking Simulation Program");

JPanel SimulationPanel = new JPanel(new GridBagLayout());

// Program Buttons

JButton RunProgramButton = new JButton("RUN PROGRAM");

static JButton UpButton = new JButton("UP");

static JButton LeftButton = new JButton("LEFT");

static JButton DownButton = new JButton("DOWN");

static JButton RightButton = new JButton("RIGHT");

JButton BlankButton = new JButton("\*");

JMenuBar mBar = new JMenuBar();

JMenu runMenu = new JMenu("Run");

JMenu helpMenu = new JMenu("Help");

static JMenuItem runItem = new JMenuItem("Run Program");

static JMenuItem stopItem = new JMenuItem("Stop Program");

JMenuItem programHelpItem = new JMenuItem("Program Help");

// Adding the Image

File issPicture = new File("ISSMain.jpg");

BufferedImage issPic; {

try {

issPic = ImageIO.read(issPicture);

}

catch (IOException ex) {

System.out.println("File cannot be found");

ex.printStackTrace();

}

}

// Label for Picture

JLabel issPicLabel = new JLabel(new ImageIcon(issPic));

// Public Class

public SimulationBox() {

SimulationBox.setSize(770,700);

// Simulation Panel initial settings

SimulationBox.add(SimulationPanel);

SimulationPanel.setPreferredSize(new Dimension(900,700));

SimulationPanel.setBackground(Color.black);

// Add JMenus

SimulationBox.setJMenuBar(mBar);

runMenu.add(runItem);

runMenu.add(stopItem);

helpMenu.add(programHelpItem);

mBar.add(runMenu);

mBar.add(helpMenu);

runItem.addActionListener(new ProgramRun());

stopItem.addActionListener(new UserProgramStop());

programHelpItem.addActionListener(new ProgramHelp());

// Initial GridBagLayout Setup

SimulationBox.setLayout(new GridBagLayout());

GridBagConstraints con = new GridBagConstraints();

con.insets = new Insets(5,5,20,5);

con.ipadx = 100;

con.ipady = 15;

con.gridwidth = 3;

// ISS Picture

con.gridx = 0;

con.gridy = 0;

//con.fill = GridBagConstraints.BOTH;

//con.anchor = GridBagConstraints.CENTER;

SimulationPanel.add(issPicLabel, con);

// Up Button

con.gridx = 0;

con.gridy = 1;

con.anchor = GridBagConstraints.NORTH;

SimulationPanel.add(UpButton, con);

UpButton.setBackground(Color.black);

UpButton.setForeground(Color.white);

// Left Button

con.gridx = 0;

con.gridy = 2;

con.anchor = GridBagConstraints.WEST;

SimulationPanel.add(LeftButton, con);

LeftButton.setBackground(Color.black);

LeftButton.setForeground(Color.white);

// Blank Button in Middle

con.gridx = 1;

con.gridy = 2;

con.anchor = GridBagConstraints.CENTER;

SimulationPanel.add(BlankButton, con);

BlankButton.setBackground(Color.black);

BlankButton.setForeground(Color.white);

// Right Button

con.gridx = 2;

con.gridy = 2;

con.anchor = GridBagConstraints.EAST;

SimulationPanel.add(RightButton, con);

RightButton.setBackground(Color.black);

RightButton.setForeground(Color.white);

// Down Button

con.gridx = 0;

con.gridy = 3;

con.anchor = GridBagConstraints.SOUTH;

SimulationPanel.add(DownButton, con);

DownButton.setBackground(Color.black);

DownButton.setForeground(Color.white);

// Run Program Button (Most likely unneeded, JMenu used instead)

/\*con.weighty = 50;

con.gridx = 0;

con.gridy = 4;

con.ipadx = 5;

con.ipady = 5;

con.insets = new Insets(5,5,50,5);

con.anchor = GridBagConstraints.SOUTH;

SimulationPanel.add(RunProgramButton, con); \*/

// JPanel Settings

SimulationBox.setResizable(false);

SimulationBox.setLocationRelativeTo(null);

SimulationBox.setVisible(true);

SimulationBox.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Add action listeners to buttons

UpButton.addActionListener(new UpButtonListener());

DownButton.addActionListener(new DownButtonListener());

LeftButton.addActionListener(new LeftButtonListener());

RightButton.addActionListener(new RightButtonListener());

// Input buttons start disabled

UpButton.setEnabled(false);

DownButton.setEnabled(false);

LeftButton.setEnabled(false);

RightButton.setEnabled(false);

// Stop Program Button starts disabled

stopItem.setEnabled(false);

}

class ProgramRun implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

AlignmentWindow.randomNum(); // Generates random values.

// Call reset alignment label method

AlignmentWindow.resetAlignmentLabel();

JOptionPane.showMessageDialog(null, "Program running... Please refer to Alignment Window", "Successful Run", JOptionPane.INFORMATION\_MESSAGE);

runItem.setEnabled(false); // Disables Run button

stopItem.setEnabled(true); // Enables Stop Button

// Enable input buttons

UpButton.setEnabled(true);

DownButton.setEnabled(true);

LeftButton.setEnabled(true);

RightButton.setEnabled(true);

}

}

class UserProgramStop implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

programStop(); // Call Program Stop Method

// Inform user that the program stopped

JOptionPane.showMessageDialog(null, "Program stopping...", "Stop Program", JOptionPane.WARNING\_MESSAGE);

}

}

class ProgramHelp implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog (

null,

"Hello, welcome to the ISS Docking Simulation Program."

+ "\n\n-To run the program, hover over 'Run' on the top left of the window and click 'Run Program'"

+ "\n\n-When run, the program will generate random values for Approach Distance, X Offset, and Y Offset on the Alignment window"

+ "\n\n-Refer to the buttons on the main program to adjust the X Offset and Y Offset, both values should be 0 before the Approach Distance is 0."

+ "\n\n-Also refer to the 'ISS Docking View' window to see where your alignment currently is."

+ "\n\n-For every button input, the Approach Distance will go down by 10.",

"Program Help",

JOptionPane.PLAIN\_MESSAGE

);

}

}

private class UpButtonListener implements ActionListener{

public void actionPerformed(ActionEvent e) {

int upButton = 1;

GraphicsWindow.updateGraphics(upButton); // Call method to update Graphic Window

AlignmentWindow.upButtonPress(); // Call method to update Alignment Window

}

}

private class DownButtonListener implements ActionListener{

public void actionPerformed(ActionEvent e) {

int downButton = 2;

GraphicsWindow.updateGraphics(downButton); // Call method to update Graphic Window

AlignmentWindow.downButtonPress(); // Call method to update Alignment Window

}

}

private class LeftButtonListener implements ActionListener{

public void actionPerformed(ActionEvent e) {

int leftButton = 3;

GraphicsWindow.updateGraphics(leftButton); // Call method to update Graphic Window

AlignmentWindow.leftButtonPress(); // Call method to update Alignment Window

}

}

private class RightButtonListener implements ActionListener{

public void actionPerformed(ActionEvent e) {

int rightButton = 4;

GraphicsWindow.updateGraphics(rightButton); // Call method to update Graphic Window

AlignmentWindow.rightButtonPress(); // Call method to update Alignment Window

}

}

public static void programStop() {

runItem.setEnabled(true); // Enables Run button

stopItem.setEnabled(false); // Disables Stop Button

// Enable input buttons

UpButton.setEnabled(false);

DownButton.setEnabled(false);

LeftButton.setEnabled(false);

RightButton.setEnabled(false);

}

}

**// AlignmentWindow.java**

package issPackage;

import java.awt.Color;

import java.awt.Component;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.Insets;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import java.awt.GraphicsEnvironment;

public class AlignmentWindow extends JFrame {

// Create Frame and Panel

JFrame AlignmentBox = new JFrame("ISS Docking Alignment");

JPanel AlignmentPanel = new JPanel(new GridBagLayout());

// JLabels

JLabel ApproachDistanceLabel = new JLabel("Approach Distance");

JLabel XOffsetLabel = new JLabel("X Offset");

JLabel YOffsetLabel = new JLabel ("Y Offset");

static JLabel ApproachDistanceValue = new JLabel("null");

static JLabel XOffsetValue = new JLabel("null");

static JLabel YOffsetValue = new JLabel ("null");

static JLabel AlignmentIndicator = new JLabel ("<< ALIGNMENT NOT YET ACHIEVED >>");

// Public Class

public AlignmentWindow() {

AlignmentBox.setSize(450,450);

AlignmentBox.getContentPane().setBackground(Color.black);

// Simulation Panel initial settings

AlignmentBox.add(AlignmentPanel);

AlignmentPanel.setPreferredSize(new Dimension(400,400));

AlignmentPanel.setBackground(Color.black);

// Initial GridBagLayout Setup

AlignmentBox.setLayout(new GridBagLayout());

GridBagConstraints con = new GridBagConstraints();

con.weightx = 0.75;

con.insets = new Insets(25, 25, 25, 25);

//con.ipadx = 100;

//con.ipady = 15;

con.gridwidth = 3;

// Approach Distance Label

con.gridx = 0;

con.gridy = 0;

con.anchor = GridBagConstraints.WEST;

AlignmentPanel.add(ApproachDistanceLabel, con);

ApproachDistanceLabel.setForeground(Color.green);

ApproachDistanceLabel.setFont(new Font("Courier", Font.BOLD, 17));

// Approach Distance Value

con.gridx = 1;

con.gridy = 0;

con.anchor = GridBagConstraints.EAST;

AlignmentPanel.add(ApproachDistanceValue, con);

ApproachDistanceValue.setForeground(Color.red);

ApproachDistanceValue.setFont(new Font("Courier", Font.BOLD, 17));

// XOffset Label

con.gridx = 0;

con.gridy = 1;

con.anchor = GridBagConstraints.WEST;

AlignmentPanel.add(XOffsetLabel, con);

XOffsetLabel.setForeground(Color.green);

XOffsetLabel.setFont(new Font("Courier", Font.BOLD, 17));

// XOffset Value

con.gridx = 1;

con.gridy = 1;

con.anchor = GridBagConstraints.EAST;

AlignmentPanel.add(XOffsetValue, con);

XOffsetValue.setForeground(Color.red);

XOffsetValue.setFont(new Font("Courier", Font.BOLD, 17));

// YOffset Label

con.gridx = 0;

con.gridy = 2;

con.anchor = GridBagConstraints.WEST;

AlignmentPanel.add(YOffsetLabel, con);

YOffsetLabel.setForeground(Color.green);

YOffsetLabel.setFont(new Font("Courier", Font.BOLD, 17));

// YOffset Value

con.gridx = 1;

con.gridy = 2;

con.anchor = GridBagConstraints.EAST;

AlignmentPanel.add(YOffsetValue, con);

YOffsetValue.setForeground(Color.red);

YOffsetValue.setFont(new Font("Courier", Font.BOLD, 17));

// Alignment Achieved/NotAchieved Label

con.gridx = 0;

con.gridy = 3;

con.anchor = GridBagConstraints.CENTER;

AlignmentPanel.add(AlignmentIndicator, con);

AlignmentIndicator.setForeground(Color.red);

AlignmentIndicator.setFont(new Font("Courier", Font.BOLD, 17));

// JPanel Settings

AlignmentBox.setResizable(false);

AlignmentBox.setLocationRelativeTo(null);

AlignmentBox.setVisible(true);

AlignmentBox.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void randomNum() {

// Generate random approach value number

int randomApproach = (int) (Math.random() \* ((250 - 100) + 1) + 100);

// Sets Approach Value to the converted string of Random Approach

ApproachDistanceValue.setText(String.valueOf(randomApproach));

// Generate random XOffset value

int randomXOffset = (int) (-20 + (Math.random() \* (41)));

// Sets XOffsetValue to the converted string of Random X Offset

XOffsetValue.setText(String.valueOf(randomXOffset));

// Generate random YOffset Value

int randomYOffset = (int) (-15 + (Math.random() \* (31)));

// Sets YValue to the converted string of Random Y Offset

YOffsetValue.setText(String.valueOf(randomYOffset));

// Initializes the Graphics Window

GraphicsWindow.graphicsInitialize(randomXOffset, randomYOffset);

}

public static void upButtonPress() {

// Create temporary ints to hold values.

int tempXOffset = Integer.parseInt(XOffsetValue.getText());

int tempYOffset = Integer.parseInt(YOffsetValue.getText());

int tempApproach = Integer.parseInt(ApproachDistanceValue.getText());

System.out.println(tempYOffset);

// Add 1 to Offset Value and Subtract 10 from Approach Distance

tempYOffset++;

tempApproach = tempApproach - 10;

// Assign these values back to JLabels

YOffsetValue.setText(String.valueOf(tempYOffset));

ApproachDistanceValue.setText(String.valueOf(tempApproach));

// Runs a check to see if alignment has been reached

alignmentCheck(tempApproach, tempXOffset, tempYOffset);

}

public static void downButtonPress() {

// Create temporary ints to hold values.

int tempXOffset = Integer.parseInt(XOffsetValue.getText());

int tempYOffset = Integer.parseInt(YOffsetValue.getText());

int tempApproach = Integer.parseInt(ApproachDistanceValue.getText());

System.out.println(tempYOffset);

// Subtract 1 from Offset Value and Subtract 10 from Approach Distance

tempYOffset--;

tempApproach = tempApproach - 10;

// Assign these values back to JLabels

YOffsetValue.setText(String.valueOf(tempYOffset));

ApproachDistanceValue.setText(String.valueOf(tempApproach));

// Runs a check to see if alignment has been reached

alignmentCheck(tempApproach, tempXOffset, tempYOffset);

}

public static void leftButtonPress() {

// Create temporary ints to hold values.

int tempXOffset = Integer.parseInt(XOffsetValue.getText());

int tempYOffset = Integer.parseInt(YOffsetValue.getText());

int tempApproach = Integer.parseInt(ApproachDistanceValue.getText());

System.out.println(tempXOffset);

// Add 1 to Offset Value and Subtract 10 from Approach Distance

tempXOffset--;

tempApproach = tempApproach - 10;

// Assign these values back to JLabels

XOffsetValue.setText(String.valueOf(tempXOffset));

ApproachDistanceValue.setText(String.valueOf(tempApproach));

// Runs a check to see if alignment has been reached

alignmentCheck(tempApproach, tempXOffset, tempYOffset);

}

public static void rightButtonPress() {

// Create temporary ints to hold values.

int tempXOffset = Integer.parseInt(XOffsetValue.getText());

int tempYOffset = Integer.parseInt(YOffsetValue.getText());

int tempApproach = Integer.parseInt(ApproachDistanceValue.getText());

System.out.println(tempXOffset);

// Add 1 to Offset Value and Subtract 10 from Approach Distance

tempXOffset++;

tempApproach = tempApproach - 10;

// Assign these values back to JLabels

XOffsetValue.setText(String.valueOf(tempXOffset));

ApproachDistanceValue.setText(String.valueOf(tempApproach));

// Runs a check to see if alignment has been reached

alignmentCheck(tempApproach, tempXOffset, tempYOffset);

}

public static void resetAlignmentLabel() {

// Resets label back to default

AlignmentIndicator.setForeground(Color.red);

AlignmentIndicator.setText("<< ALIGNMENT NOT YET ACHIEVED >>");

}

public static void alignmentCheck(int approach, int xOffset, int yOffset) {

if (approach < 0) { // If Approach Distance is less than 0

if (xOffset == 0 && yOffset == 0) { // If Alignment has been reached

// Update alignment indicator

AlignmentIndicator.setText("<< ALIGNMENT REACHED >>");

AlignmentIndicator.setForeground(Color.green);

// Display success message

JOptionPane.showMessageDialog(null, "Alignment successfully reached at the final moment!", "Alignment Success!", JOptionPane.INFORMATION\_MESSAGE);

// Stop Program

SimulationBox.programStop();

}

else { // If Alignment has not been reached

// Display failure message and stop program

JOptionPane.showMessageDialog(null, "Alignment has not been reached", "Alignment Failure", JOptionPane.ERROR\_MESSAGE);

SimulationBox.programStop();

}

}

else if (xOffset == 0 && yOffset == 0) { // If Alignment has been reached

// Update alignment indicator

AlignmentIndicator.setText("<< ALIGNMENT REACHED >>");

AlignmentIndicator.setForeground(Color.green);

// Display success message

JOptionPane.showMessageDialog(null, "Alignment successfully reached!", "Alignment Success!", JOptionPane.INFORMATION\_MESSAGE);

// Stop Program

SimulationBox.programStop();

}

}

}

**// GraphicsWindow.java**

package issPackage;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.GridBagLayout;

import javax.swing.JComponent;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class GraphicsWindow extends JFrame {

JFrame GraphicsFrame = new JFrame("ISS Docking View");

static int x = 178;

static int y = 174;

static JComponent component = new JComponent() {

public void paintComponent(Graphics g) {

// Draw X Y Lines

g.setColor(Color.GREEN);

g.drawLine(20, 180, 360, 180);

g.drawLine(185, 20, 185, 340);

// Draw proximity ovals

g.drawOval(165, 160, 40, 40);

g.drawOval(125, 120, 120, 120);

g.drawOval(85, 80, 200, 200);

// Draw ISS Indicator

g.setColor(Color.RED);

g.fillRect(x, y, 15, 15);

}

}; // end of JComponent

public GraphicsWindow() {

GraphicsFrame.setSize(400, 400);

GraphicsFrame.getContentPane().setBackground(Color.black);

GraphicsFrame.add(component);

// JFame Settings

GraphicsFrame.setResizable(false);

GraphicsFrame.setLocationRelativeTo(null);

GraphicsFrame.setVisible(true);

GraphicsFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void graphicsInitialize(int randX, int randY) {

// Resets X and Y to default

x = 178;

y = 174;

// Initializes the X and Y at a 5 pixel algorithm

x = x + (randX \* 5);

y = y - (randY \* 5);

// Repaint the graphics

component.repaint();

}

public static void updateGraphics(int button) {

// Up Button

if (button == 1) {

y = y - 5;

}

// Down Button

if (button == 2) {

y = y + 5;

}

// Left Button

if (button == 3) {

x = x - 5;

}

// Right Button

if (button == 4) {

x = x + 5;

}

// Repaint the graphics

component.repaint();

}

}

**// END OF PROGRAM**