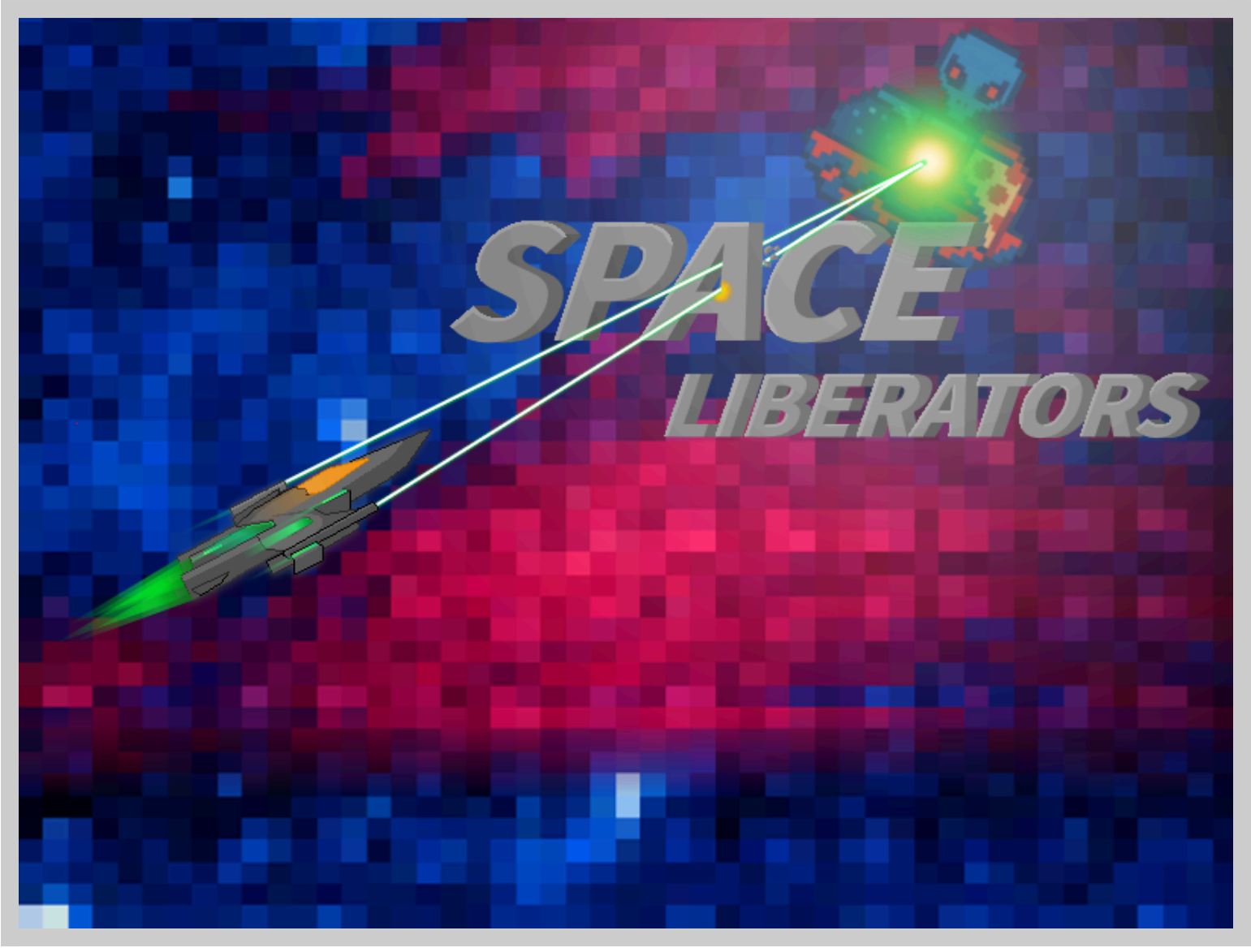
CSCI 185
Antoun Elabd & Mohamed Youssef
Dr. Wenjia Li
Final Programming Project
5/8/2025

Collaboration Report By Antoun Elabd & Mohamed Youssef



Our Individual work

Antoun Elabd:
My work consisted of creating the frontend. I worked on the fast-to-slow background transition that plays after the logo screen, as well as the scrolling effect that follows. I worked on several components such as JButton, JLabel, and JTextField. I created barriers to prevent the ship from passing the midpoint of the screen or the outlines. I implemented the arrow key controls and developed Wave 1 of the game. I also worked on the sprites, including the ship, the background,the game logo, and a few others that were unused. I implemented file I/O and the exception handling behind it so that scores can be saved with some errors caught by Mohamed. Additionally, I worked on the backend functionality of the waves, including organizing the enemies into rows and giving them that slight random movement. I also added a barrier to prevent enemies from moving past a certain point.

Mohamed Youssef:
My work primarily focused on the backend. I learned how to efficiently use and implement the "update" and "render" methods in my game and player classes to help implement every change and player movement. For example, I created the canvas used to render the game elements. I implemented collision detection so that lasers destroy enemies by drawing a rectangle around them and seeing where and when they intersect so both items can be removed. I also contributed to sprite development, such as the "Wave 2" sprite and some unused sprites that were left out due to time constraints. I created Wave 2 of the game and implemented the spacebar as the shoot function. I helped organize the code into classes, as seen in the number of classes we have. I also worked on the event listener functionality to help read all the users controls.

Note: Given the project's size, we collaborated on several of each other's parts which may not have been listed and often helped fix each other's errors.

How we met the requirements

We started by building two layers using a JFrame for the main window and a custom JPanel to draw the moving background. We used Java Swing and AWT for the interface and graphics, and added a second panel at the top for buttons and a text field. We used more than four GUI components including JButtons for Start, Pause, and Save Score, a JLabel to prompt name entry, a JTextField, and an AWT Canvas for the game area. For file input and output, the Save Score button writes to a scores.txt file using a try-catch block to handle errors. We used conditionals and loops throughout, like if-else for splash and pause logic and for-loops to spawn enemy waves. The code is broken into custom classes like Player, Enemy, Waves, and Background to

keep it readable and organized. We also set up event handling with ActionListener for buttons and KeyListener on the canvas for game controls.

Two Layers (JFrame + JPanel) + Four (GUI Components)

```
public Display(int width, int height) {  1 usage
    super( title: "Space Liberators");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setResizable(false);

    setLayout(new BorderLayout());

    controlPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
    startButton = new JButton( text: "Start");
    pauseButton = new JButton( text: "Pause");
    nameLabel = new JLabel( text: "Name:");
    nameField = new JTextField( columns: 10);
    saveButton = new JButton( text: "Save Score");

    controlPanel.add(startButton);
    controlPanel.add(pauseButton);
    controlPanel.add(nameLabel);
    controlPanel.add(nameField);
    controlPanel.add(saveButton);

    add(controlPanel, BorderLayout.NORTH);

    canvas = new Canvas();
    canvas.setPreferredSize(new Dimension(width, height));
    add(canvas, BorderLayout.CENTER);
```

File I/O with Exception Handling

```
saveButton.addActionListener( ActionEvent e -> {
    if (currentGame == null) {
        JOptionPane.showMessageDialog( parentComponent: this,
                message: "Game isn't running yet!", title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        String name = nameField.getText().trim();
        if (name.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "Please enter your name above!", title: "Warning", JOptionPane.WARNING_MESSAGE);
        } else {
            try (PrintWriter out = new PrintWriter(
                    new BufferedWriter(new FileWriter( fileName: "scores.txt",  append: true)))) {
                out.println(name + " - " + currentGame.getScore());
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "Score saved!", title: "Done", JOptionPane.INFORMATION_MESSAGE);
                saveButton.setEnabled(false);
                nameField.setEnabled(false);
```

Event Handling

```java
@Override
public void keyPressed(KeyEvent e) { keyspressed[e.getKeyCode()] = true;

@Override
public void keyReleased(KeyEvent e) {
    keyspressed[e.getKeyCode()] = false;
```