# DEFUSE THE BOMB:

# A CSC 102 PROJECT

***Squid-ish***

Bomb Defusal Manual

TEAM: Christa-Marie Seerattan, Khalil Smith, Megan Dowdell, Matthew Peplowski

# **Squid-ish Game Bomb Manual**

**About the Game:**

**Defuse the Bomb** is an interactive Python-based game created in Pygame for our CSC102 final project. The game simulates a high-stakes situation where the player must progress through four mini-games (called “phases”) to safely defuse a cartoon-style bomb. Each phase represents a different childhood game with a twist—designed to test logic, memory, timing, and luck. If the player fails a phase, the bomb “explodes” and the game ends with a sound effect and visual cue. If all four phases are cleared, the bomb is defused and the player wins.

This game emphasizes event handling, Pygame GUI design, randomized outcomes, and modular code architecture. We also practiced working in a team using GitHub, merging branches, and writing clean, readable code with comments.

The game begins with a simple launch menu and transitions into four randomized mini-games that test your reaction time, memory, logic, and intuition. Each mini-game is associated with one of the bomb’s physical components, simulated through a combination of keyboard, hardware toggles, and buttons. The suspense builds with visual and audio cues, drawing inspiration from the Squid Game series’ eerie atmosphere.

# **Game Components:**

The game consists of four sequential modules or "phases" the player must complete. Each module mimics a bomb component (e.g., wires, keypad) and offers interactive elements. These modules or “phases” will be presented in a randomized order to the user. For example, for one bomb user, the order may be Simon Says, Tic Tac Toe, Red Light Green Light, and Hopscotch. Yet for the next person, the game order may be Red Light Green Light, Simon Says, Hopscotch, Tic Tac Toe.

Each phase is designed to mimic the challenge of defusing a complex bomb. The game is modular, meaning new games or hardware inputs could be added easily in future updates. Python’s pygame library controls the GUI and gameplay logic, while GPIO integration enables external hardware inputs like physical buttons, toggles, and wires to interact with the software.

# **Phase 0: Launch Menu**

The launch menu is the first screen a player sees when opening the game. It sets the tone with cohesive Squid Game–inspired visuals and ominous background music. The menu is intuitive and designed using pygame surfaces, buttons, and text elements. Users can choose to start the game, learn about the game, meet the team, or quit from this launch menu.

* **Start Game**: Begins the game with a randomly generated sequence of four phases.
* **How to Play**: Opens an instruction screen explaining game rules and input types.
* **Meet the Team**: Displays team member names and roles with a group image or icons.
* **Quit**: Exits the game window.

# **Phase 1: Tic Tac Toe**

REGARDING THE KEYPAD

The player must win or tie a quick round of Tic Tac Toe against a basic AI. If the player loses, the bomb explodes.

Insert Picture

# **Phase 2: Red Light, Green Light**

REGARDING THE BUTTON

The player must “move” during green lights and stay still during red lights. Any movement detected during a red light results in failure.

Insert Picture

# **Phase 3: Simon Says**

REGARDING THE WIRES

Simon Says is a childhood classic memory game where players must mimic a sequence of actions. This game starts with the recruiter of squid games communicating a simple command urging the player to use the wires to repeat the command exactly.



* The Goal: Complete the command using the wires to avoid being eliminated.
* Mechanics: The commands are randomly using the colors of the wires to urge the player to complete an interactive task using the wires.
* Audio/Visual: The recruiter is vocalizing the commands to the player and the player must listen and enact the task when told to do so.
* Player Feedback: When the task is successfully completed the next command will be announced until the game is complete!
* Development Notes: No matter how many times you play this game the order of the commands are randomized!

# **Phase 4: Hopscotch**

REGARDING THE TOGGLES

This phase simulates a decision-based mini-game inspired by the physical bomb layout shown in class presentations.



* Objective: The player chooses one of four possible "tiles" (represented as toggles or hopscotch blocks) to step on.
* Mechanics: Two of the four panels are randomly selected to be "safe" each time. If the player selects a correct one, they advance. If they choose wrong, it results in a penalty of resetting to level 1 (same success/fail tiles remain).
* Probability: Currently 50% chance of success
* Audio/Visuals: Consider integrating a unique sound effect on success/failure and a graphic showing hopscotch tiles.
* Player Feedback: Visual cue when a selection is made (e.g., highlight or blink), and message indicating success or failure.
* Development Notes: Uses Python's random module to determine the correct panel. Logic is simple but allows room for enhancement (e.g., patterns, memory challenge).