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

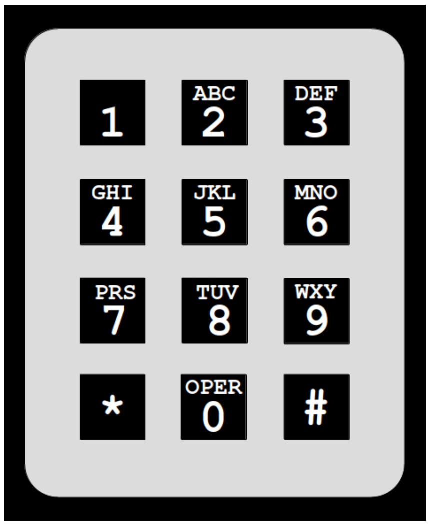
The launch menu is the first screen a player sees when opening the game. It sets the tone with cohesive Squid Game-inspired visuals and ominous background music. The menu is intuitive and designed using pygame surfaces, buttons, and text elements. Users can choose to start the game, learn about the game,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

- **Start Game:** Begins the game with a randomly generated sequence of four phases.
- **About Game:** Opens an informational screen about the Squidish game and mini games.
- **Meet Team:** Displays team member names, roles, information, and images.
- **Quit:** Exits the game window.

Phase A: 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.



In this phase, the player must face off against an AI opponent in a fast-paced game of Tic Tac Toe. The game simulates a bomb keypad input where each square on the 3x3 board corresponds to a number key (1–9). Quick decision-making and pattern recognition are key to survival.

Objective: Win or tie against the AI. A loss causes the bomb to explode. The game continues playing until either the user or AI reaches three wins.

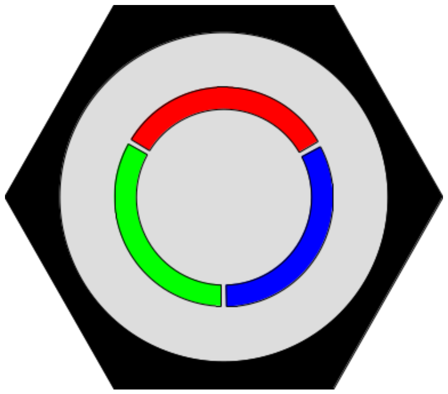
Mechanics:

- The game board is a 3x3 grid
- Player inputs are collected via a hardware numeric keypad or mapped keyboard keys (1–9).
- The player uses numbers to select grid positions. After each move, the AI selects an open tile using basic logic to block or win.

Phase B: Red Light, Green Light

<u>REGARDING THE BUTTON</u>

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



Inspired by the chilling Squid Game sequence, this phase requires precise timing and fast reflexes. The player must “move” only during green light phases by pressing the button. If the player presses the button during a red light, the game ends.

Objective: Reach the finish line by only pressing the button during green lights.

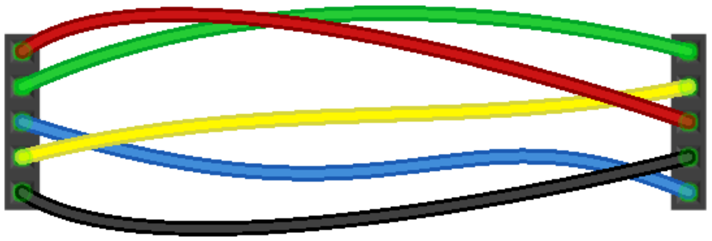
Mechanics:

- A timer randomly alternates between “RED” and “GREEN” states.
- The player presses a large GPIO button to simulate movement.
- During GREEN, each successful button press advances the player a small distance.
- During RED, any button press results in instant failure.

Phase C: 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.

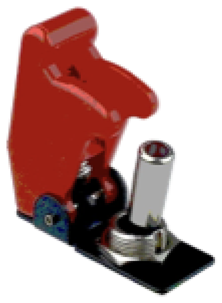


- **Objective:** Complete the command using the wires to avoid being eliminated.
- **Mechanics:**
  - The commands are randomly using the colors of the wires to urge the player to complete an interactive task using the wires.
  - The player has the option of reconnecting or disconnecting the green, red, brown, yellow, or orange wires.
  - After every command is announced, the player has 10 seconds to complete the task.
  - If the user fails to complete the task in time, GAME OVER!!
- **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. The player will be given ten commands and must listen carefully to beat this game.
- **Development Notes:** No matter how many times you play this game the order of the commands are randomized, sorry no memorization!

Phase D: Hopscotch

REGARDING THE TOGGLES

Hopscotch is a suspenseful, probability-based mini-game that draws inspiration from childhood games of chance. In this digital twist, the player faces a set of four “panels” or “tiles,” only one of which is safe to step on. The phase uses physical toggle switches to simulate choosing a panel, making it feel tactile and interactive—mirroring the Squid Game aesthetic where one wrong step can end it all.



- **Objective**: The player chooses one of four possible "tiles" (represented as toggles or hopscotch blocks) to step on.
- **Mechanics**:
    - The screen shows four hopscotch blocks labeled A, B, C, and D.
    - The game randomly selects two of the four as the “safe” tile per round using random.choice().
    - The player uses one of four toggle switches, each mapped to a tile, to “step” on a panel.
    - The correct toggle must be flipped to proceed to the next level, otherwise, the game resets to level 1.
- **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).