# Defuse the Bomb Manual

**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 five 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.

# Game Overview:

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.

# 

# Phase 0: Launch Menu

The game opens with a simple menu screen where players can start the game or quit. It may also include sound effects or background music.

# 

# Phase 1: Tic Tac Toe

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

# Phase 2: Red Light, Green Light

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

# Phase 3: Simon Says

# Phase 4: Hopscotch Module (Megan)

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 "panels" (represented as toggles or hopscotch blocks) to step on.
* Mechanics: One of the four panels is randomly selected to be "safe" each time. If the player selects the correct one, they advance. If they choose wrong, it results in a penalty (e.g., restart or game over).
* Probability: Currently 25% chance of success. May be increased to 50% or 75% (i.e., 2 or 3 panels safe) for accessibility.
* 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 correct panel. Logic is simple but allows room for enhancement (e.g., patterns, memory challenge).

# Credits and Team Roles:

Christa-Marie Seerattan

Khalil Smith

Megan Dowdell

Matthew Peplowski