Section 11 Group 2 Executive Summary

Sung Hyun Shin, Nicholas Marcopoli, Patrick West, Thomas Eckburg, Santiago Calderon Novoa

---

Background Information and Purpose

The group's design statement is as follows: create a fun interactive game modeled after the game *Keep Talking and Nobody Explodes*[1] to allow the user to defuse a randomly generated bomb. Additionally, develop three algorithms aimed to simulate the game to find the fastest, most accurate solving method. In this two-player game, one player can see and manipulate the bomb via interaction with the GUI, while the other player is given only the rulebook - the printed-out instructions on how to defuse the bomb. Both users must work together to provide specific inputs to the GUI to "defuse" the bomb.

Selected Design

In *Figure 1*, the bomb GUI consists of four panels, Switches, Numpad, Buttons, and Wires, whose positions and contents are randomized, an attempt meter, timer, and an apply button. In *Figure 2*, the simulation tool consists of an editable number of bombs which serves as the number of trials, time delay which is needed in simulating real time for bomb solving, and three buttons. The top button starts a new simulation. The bottom "Load Last Data" loads the last data simulated. The "Start Simulation From Data" uses the information in "Number of Bombs" and "Time Delay" to add that amount of bombs onto loaded data. The graph shows the average simulated time to solve each section of the random bomb with three strategies. The table below it shows the average time, failed attempts, and computational time for each strategy.

The three methods utilized for bomb solving are Brute Force, Randomization, and Rule Abider. Brute Force tries every possible bomb answer in increasing order. Randomization attempts random solutions, without repeating attempted solutions. The Rule Abider solves the bomb correctly according to the rules of bomb solving.

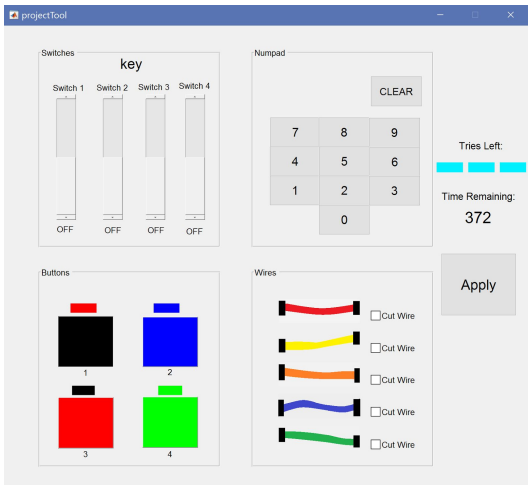

Figure 1. Game Bomb GUI Example

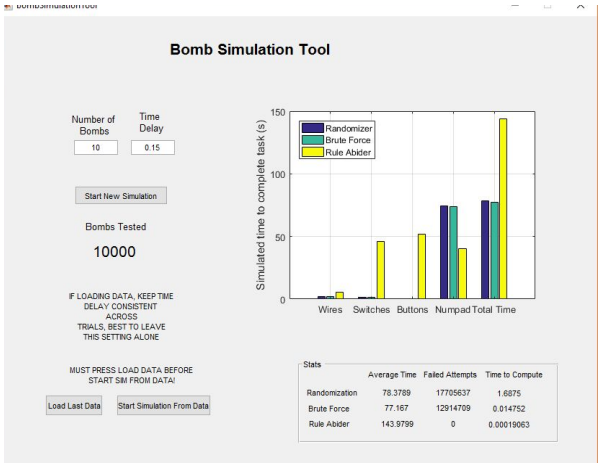

Figure 2. Bomb Simulation Tool GUI

[1] "Keep Talking and Nobody Explodes." Keep Talking Game. 2015. Accessed April 30, 2017. http://www.bombmanual.com/how-to-play-pc.html.

Results

After developing the game and creating three solving algorithms, ten-thousand bombs were tested to determine the advantages and disadvantages of using one solving method over another. The results of the test are displayed in *Table 1* below. The main conclusion that can be drawn from the data is that the Randomization algorithm takes significantly more computational time to compute a correct answer than the Brute Force or Rule Abider algorithms, while also failing more attempts than these algorithms.

One important point to consider is that the Rule Abider's accuracy hinges on knowledge of the rules. If the rules are not known, Rule Abider becomes useless and the usefulness of the Brute Force and Randomization algorithms increase, as they allow a computer to use raw computational power rather than knowledge of rules to solve the bomb. When Brute Force and Randomization are compared, it can be seen that Brute Force is the fastest and most accurate. It is the preferred method of bomb solving when rules are unknown, assuming that the computer can provide incorrect solutions to the bomb without losing the game.

Table 1. Data gathered for 10,000 bombs

	Rule Abider	Brute Force	Randomization
Average computational time (s)	$1.906 * 10^{-4}$	$1.475 * 10^{-2}$	1.688
Average failed attempts per bomb	0	1291.471	1770.564

Recommendations

After working on the project, the group thought of several improvements that could be made. Firstly, the introduction of another solving method was proposed in order to compare it to the ones currently developed. This new solving method would consist of a "smart" AI that would learn from its mistakes and, in the process, develop a database of rules that apply to solving the bombs. After many trials, this algorithm would be able to solve a bomb with no failed attempts, much like the Rule Abider algorithm. Secondly, the group proposed making the rules that dictate the solution to a bomb more complex by making more of the modules inter-dependent[2]. In this way, solving time would increase as finding the right solution would be more time-consuming. Additionally, the group thought of making the different game modes more difficult by decreasing the time to defuse the bomb, adding a penalty that would deduct time for every failed attempt, and/or only having one attempt per bomb in the hardest game mode.

After much discussion, the group came to the conclusion that the "simulated time" that was calculated for each algorithm, shown in the graph of *Figure 2,* did not truly represent the computational speed of each of the algorithms. The more accurate metric to compare the algorithms is the raw computational time that MATLAB took to solve the bombs; therefore, the group believes that by adding graphs that compare only the raw computational time and not simulated time, a more meaningful conclusion can be withdrawn from the results obtained.

[2] Wherein the solution to one of the modules depends on the solution of another module.