Angel Chikumbirike, Abhiram Garre, Orion Golden, Nicholas Johnson, Senny Lu, Phyo Naing
Professor: James Daly, Ph.D.
Class: COMP 4110 -- Software Engineering I
10/27/2025

Project Requirements

Hardware Requirements:

1. The game requires a *computer to play.
    1.1. The *computer must have a screen or monitor to display images.
    1.2. The *computer must have a mouse or a touchpad.
        1.2.1. The mouse or touchpad must have the ability to move the computer pointer
        1.2.2. The mouse or touchpad must have a button to simulate a left-click

***Computer**: a device that is either a laptop or a desktop

UI Requirements:

1. The user must be able to control the runtime, options, and *main gamestates of the program through a UI.
    1.1. The game must start with a main menu.
    1.2. The main menu must display the game's title.
    1.3. The main menu must have all the *UI functionality of the pause menu
    1.4. Additionally, this menu must have an option to begin playing the game.
2. While playing (not on the main menu or the pause menu), the player must be able to access a pause menu.
    2.1. The pause menu must *functionally pause all ongoing activity in the game.
    2.2. The pause menu must contain a way to change the global volume of the game.
        2.2.1. This option should take the form of a slider.
    2.3. The pause menu must contain a way to return to the main menu.
        2.3.1. This option should take the form of a "Main menu" button.
    2.4. The pause menu must contain a way to quit to desktop.
        2.4.1. This option should take the form of a "Quit to desktop" button.
        2.4.2. Quitting to the desktop should be done *cleanly.
            2.4.2.1. Quitting the game should take at most 1 second.

***main gamestates:** The main gamestates of the game refer to the main menu, the pause menu, and the gameplay of the game.
***UI-functionality:** Shares the functionality of the pause menu, not pausing itself, as in the main menu, there is nothing to pause.
***functionally pause:** This means that no actions will happen against the player in the game when paused. Because of the nature of the game we are making, it is unnecessary to pause time, as the game waits for the player to make their turn before doing anything to them.
***cleanly:** In this context, quitting cleanly means that the game does not just crash itself to quit to the desktop (a surprisingly common and annoying thing some game devs just don't fix). It also means that quitting is functionally the same as clicking the X button on the window.

Game Layout Requirements

1. The game must consist of three main areas: *Hand, Equation, and Enemy
    1.1. The *Hand area must contain the current cards of the player.
    1.2. The *Equation area must allow the player to place cards into it from the *Hand area.
        1.2.1. The *Equation area must calculate the value formed by parsing the cards in it.
    1.3. The *Enemy area must contain enemies.
        1.3.1. Enemies must consist of a *mathematical criteria.
        1.3.2. Enemies must be defeated when their criteria are fulfilled by values calculated in the *Equation area.
2. The player must have *health points (hp)
    2.1. The player must lose *health points when they fail to defeat the enemies in a set number of turns
    2.2. When the player runs out of *health points, the run should end.

Card and Deck Requirements

1. At the start of each turn, the player must be dealt cards from the deck of cards to the *Hand area.
2. The deck must have three card types: Number, Operation, and Modifier.
    2.1. Number cards must display whole numbers on them.
        2.1.1. Number cards must not be placed *next to other number cards.
    2.2. Operation cards must display binary mathematical operators on them.
        2.2.1. Operation cards must not be placed *next to other operation cards.
    2.3. Modifier cards must display unary mathematical operators on them.
        2.3.1. Modifier cards must be placed *next to a number card.
3. When the deck runs out of cards, the cards in the *discard pile must be reshuffled into the deck.
4. At the end of each turn, any cards that were used should return to the *discard pile.

***Deck**: the draw pile of cards
***Discard pile**: the pile of used cards
***Next to / adjacent**: immediate left or right of a card
***Hand area**: location of cards dealt at the start of a turn
***Equation area**: location to compute the value of the cards in
***Enemy area**: location of enemies
***Health points**: the life of the player
***Mathematical criteria**: number-theoretic properties such as even, odd, prime, etc.

Game Difficulty Requirements:

1. The game must get harder the more *levels the player completes.

1.1. More specifically, the criteria needed to eliminate an enemy will increase. For example, an enemy previously killable with even numbers will require even numbers > 30, etc.
1.2. Additionally, the enemies may also increase in numbers.

2. The added difficulty must be manageable by a skilled player, I.E. the game doesn't just become impossible.
   2.1. After a certain point, it may be acceptable for the game to be impossible, but that point should be *hard, *time consuming, and *inconsistent to reach.
   2.2. The "point of impossibility" should be an unavoidable design constraint, not a conscious decision.

***hard:** In this context hard means that it is not easy to reach the point of impossibility; the knowledge requirement to complete the first 10 levels should not be above 8th grade math.

***time consuming:** In this context, time consuming means that reaching this point of impossibility is not able to be achieved with little time investment. Little time investment means less than 10 minutes.

***inconsistent to reach:** In this context, the player should not be able to reach the point of impossibility consistently.

Scoring Requirements

1. The game must track the player's score to measure their success during gameplay.
   1.1. The game must accurately track the player's score on the game's UI.
   1.2. The game must add score for every *level the player completes.
2. The game must display the player's score at the end of a *run.
   2.1. The game should display other scores of the player achieved as part of a leaderboard.

***level:** A level refers to a single scene with enemies in it. A new level occurs when all enemies in the current level are destroyed, and when a new card is added to the player's deck.
***run:** A run in this context refers to a playthrough of the game, from start to game over.

Performance Requirements

1. The game must be able to run on *most home/school computers.
   1.1. The game must be able to run on Windows (*most home/school computers are Windows).
2. The game must be able to run at AT LEAST 30 fps on *most home/school computers.
   2.1. The game should run at much higher fps (60-120).
      2.1.1. The game must not play differently at higher fps.
      2.1.2. The game must have a tick system or equivalent.
3. The game must be *relatively bug-free.
   3.1. The errors produced should be understandable by *users with a *decent understanding of computers.

***most home/school computers:** For this project's purposes, we will define this as being computers currently being produced by manufacturers. Older computers bought on some platforms like eBay don't count.

***relatively bug-free:** For this project's purposes, this means that out of every 10 hypothetical players who play this game, only 1 will run into a noticeable bug during a complete playthrough.
***decent understanding of computers:** For this project's purposes, a decent understanding of computers means knowing how to download and run executables or visit a website, knowing how to use a mouse, knowing what crashes look like, and potentially more.
***users:** users should include children at least 9 years old, parents, teachers, all of which have passable knowledge about computers.