RESCUER

Rescuer is a **top-down** arcade game that utilizes JavaFX to create a well-functioning game. The rescuer is on a mission to save the hostages all while avoiding dangerous, unpredictable enemies. Once all the hostages are collected and the trophy is collected, the rescuer escapes, emerging victorious.

HOW TO PLAY:

To play Rescuer, instructions are simple. All you have to do is move with the directional keys, and try to dodge the movement of the enemies. There are four hostages scattered across the map, and you must collect all four of them. If you miss a single hostage, there's no way for the end tile to spawn.

You start off with three hearts, and if you lose all three lives you lose the game. Hearts are easily viewable in the HUD menu right below the game screen. The game also features a bonus hostage that spawns somewhere on the board when all four primary hostages are saved, and grants extra points to generate a high score.

Installation of the game is easy. Once you create a JavaFX project with Maven integration, you can just unzip the files to the right target location to run and play the game. Keep in mind that JavaFX must be properly installed in order to play the game. You can find out more in the .README portion on Gitlab.

CHANGES:

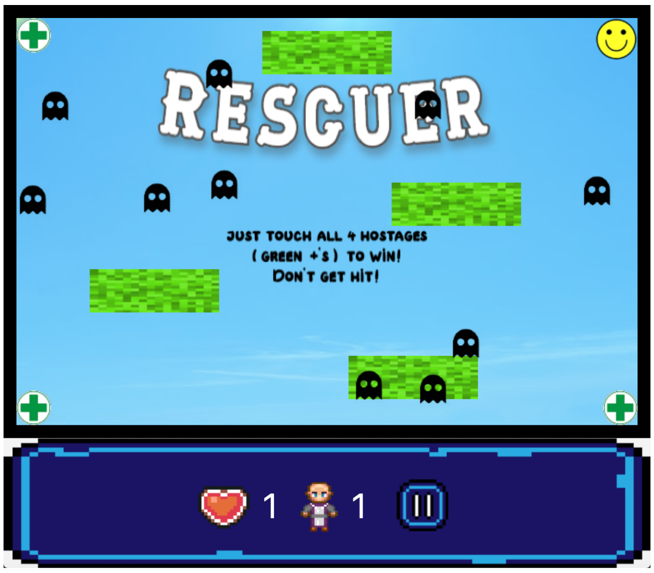
The main changes from the initial UML diagram from Phase 1 mainly consisted of how classes operated. We added classes such as Hostages, that relied on integration that was mainly done through a Person superclass, and later passed on methods to be used for the appropriate subclasses (ie: the Player, Enemies, Hostages, etc.). While the names/specifics of the UML diagram changed, the overall idea and design of the game did not. We still heavily rely on Superclassing for the game, just the methods and types of classes added were changed slightly.

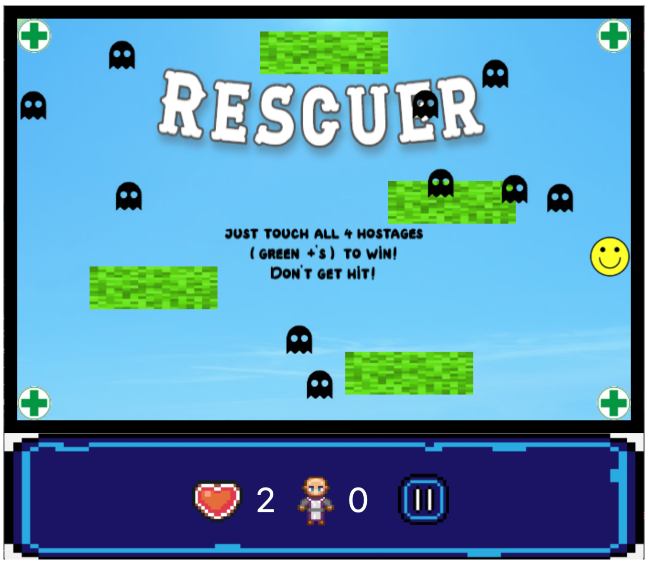
Furthermore, we made changes to the structure of the map. Initially, we were going to have unbreakable walls, breakable walls, doors, and an exit. We decided to keep the unbreakable wall, but remove the breakable walls and doors due to time constraints and so we could focus on the unbreakable wall implementation. In addition, we added another object to the map that acts as “sticky traps”. Whenever a player touches a trap, they are slowed down until they are no longer touching it. We made this change to further challenge the player and to make the game more engaging. We also had to forgo hostage attributes which would give the player special abilities because of a combination of time constraint and lack of experience with java and its other dependencies. Instead we just aggregated all hostages into a general primary hostage which helped reduce many nuisances and unnecessary complexity to our game.

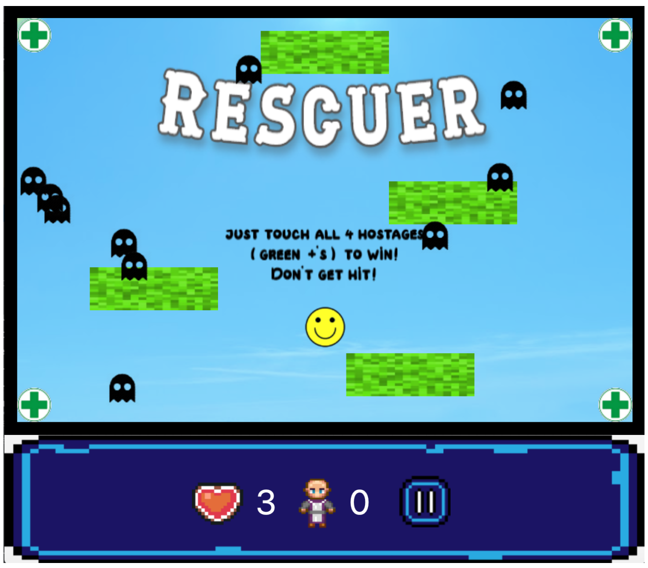
IMPORTANT LESSONS LEARNED:

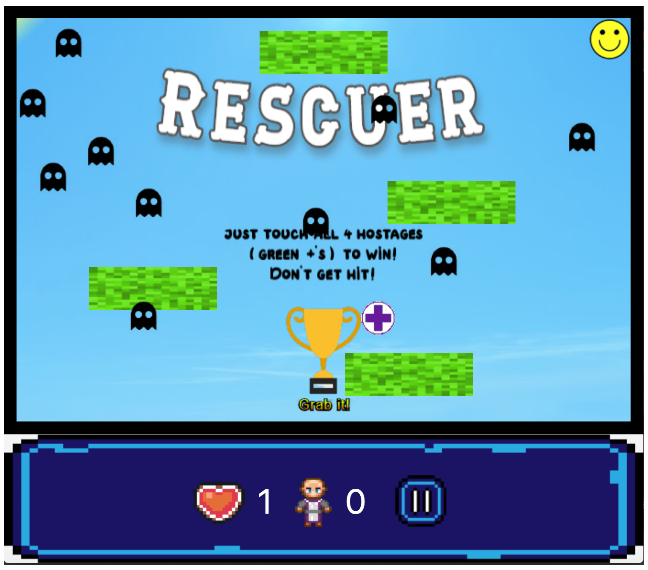
In general, the main lessons learned from the project were of how resiliency and communication are absolutely vital for a project. We used Discord to converse and chat about what needed to be done, and verbally expressed any issues or problems that had to be addressed for the game. Teamwork makes the dream work after all, and a successful project relies on great communication and amiability.

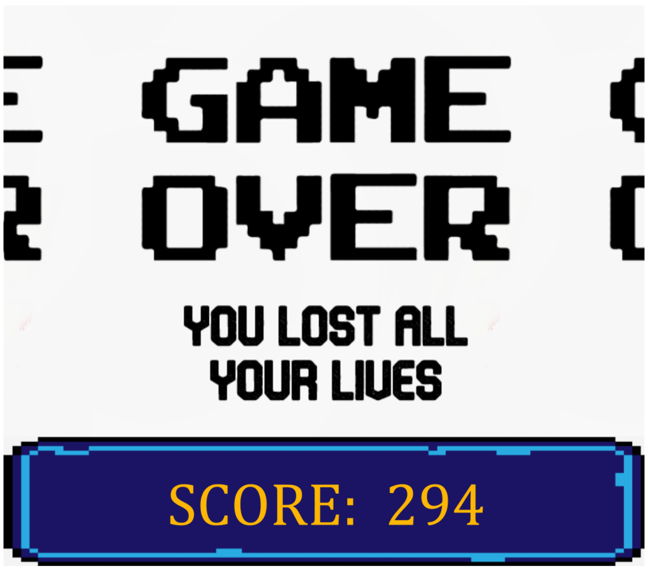
For overall improvements, we learned a lot more about the functionality and importance of using Git for creating group work. While we ran into some Git issues at the beginning, such as Git merge errors, we were able to solve them and learn the importance of good software development habits. Overall, Git seamlessly allows teams to work together fluidly throughout a project, and the beauty of the integration cannot be understated. Arguably the most important lesson of CMPT 276, and of Rescuer in general was the lessons learned in software development, with our group gaining crucial experience in these manners. It was an amazing experience to try and persevere through the intricacies and possible short-comings that occur during the software development cycle. The project lets us handle (on a fundamental level) possible problems that can naturally occur during software development. The freedom of choosing our own software design model, and subsequently dealing with the pros & cons of said model (past what was just taught as theory in class), was truly a valuable growth experience.

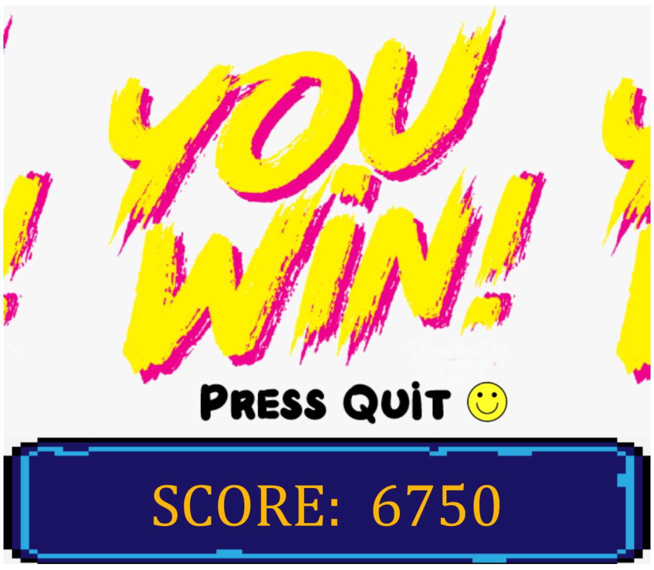

RESCUER
JUST TOUCH ALL 4 HOSTAGES
(GREEN +'s) TO WIN!
DON'T GET HIT!
1
1


RESCUER
JUST TOUCH ALL 4 HOSTAGES
(GREEN +'s) TO WIN!
DON'T GET HIT!
2
0


RESCUER
JUST TOUCH ALL 4 HOSTAGES
(GREEN +'s) TO WIN!
DON'T GET HIT!
3
0


RESCUER
JUST TOUCH ALL 4 HOSTAGES
(GREEN +'s) TO WIN!
DON'T GET HIT!
Grab it!
1
0


GAME
OVER
YOU LOST ALL
YOUR LIVES
SCORE: 294


YOU
WIN!
PRESS QUIT
SCORE: 6750