Design Documentation

Eric Kligner
cmput 379
Winter 2014
1278636

**** User Documentation

** Build and run
The instructions to build and start my game are included in the README. However if you type ‘make all’ and once successfully compiled run ‘make run’ the game will start.

** Game play
You begin with 5 rockets. If you end up with 0 rockets and your last show has completed without hitting a saucer you will lose the game. However for each enemy saucer you kill you get 2 additional rockets up to a total of 10 rockets. Once the ten rocket max is hit no more will be added for a successful kill. Now, if you still have rockets remaining you can still lose the game. If 5 enemy saucers reach the right side of the screen in tact then our planet is doomed to the alien invasion. In this game unfortunately there is no sense of winning, or finishing it. It will grind on until you make enough mistakes to let terra succumb to the alien threat. But that shouldn’t stop you from trying to best your friends high score.

** Points
For each rocket that hits its target and knocks an enemy saucer out of the sky you will get 100 points added to your score. There is no way to lose points, and your score will be returned to you once the game is over.

**** Developer Documentation

** Code structure

Main function
The main function defines all the thread ID’s and strings for the rockets and the cannon. It then creates one thread to handle all of the spawning of threads for each ship. Then main creates a thread to keep track of the score and the end conditions. Finally it creates a struct of a cannon and proceeds to a while loop. In the while loop we wait for key input and process it.

Keys
Q will quit and exit the program.
The left and right arrow keys will move the cannon left and right respectively.

Space will fire a rocket from the current cannon position.

** Configuration

`Configuration (for all operations related to start-up and configuration)`

Setup_ncurses will start up all the functions needed to begin drawing in the terminal. The spawn ship thread will, on a random timeout spawn a new ship, in a thread with a random speed. The ships will fly in the top 5 rows of the screen from left to right. Next a keep_score thread will start, this handles all the game stats and handles the end game conditions.

** Shooting

`Shooting (with all the functionality for creating and moving rockets)`

A space ‘ ‘ will fire a rocket up towards the ships. Each rocket fired will be in a new thread and interacts with each ship that’s visible. If two ships occupy the same space it comes to what ship is first in the data structure I use to

** Rendering

`Rendering (with all functionality for display)`

Each thread for that is involved in drawing an object (Rockets and ships) takes care of rendering them selves in the terminal. Each thread will call a mutex lock around their portion of code that clears, write or updates the screen. As well each object manages its own timeout in the thread. Sleeps for a random time and then calculates the new position and draws it. Each thread that draws to the screen handles cleaning up its old state as well.

** Data Structures

** Threads

`Explain what your main thread does, what your saucer threads do, etc.`

Main Thread

Spawn_saucer Thread

Saucer Thread

Rocket Thread

** Critical Sections

`Explain what parts of the code are critical sections, what global variables and mutexes might be associated with those sections, etc.`