Galactic Defense

Game Design Document

Designer/Programmer: Kevin Wong

ID : 3339323

Updated - October 2017

Game Overview

This game will take place in space. The user will control a spaceship on a 2D plane. The objective of the game is to obtain the highest score before losing all of your health The user will have a health bar of 3 and will have to survive incoming waves of asteroids. Each time the user is hit by an asteroid, the asteroid is destroyed and the user loses a health piece. The concept of the game stems from an old arcade style mouse pointer that shoots incoming pieces of rubble called Asteroids. In this game, for each asteroid you destroy you earn points - having a goal of obtaining the highest score before you lose all your health. The player will play on fix screen with asteroids hitting the edge show up on the opposite side of the screen (same with player). The player will be able to shoot laser/bullets, and have a rechargeable pulse that pushes all asteroids near the players in an opposite direction.

Game Goals

The goal of the game is to obtain the highest score before you lose all of your health. It is imperative that the user balances the use of their pulse in times that they need it as it takes a while for it to recharge. It is ideal that the player take note of how fast they are able to shoot bullets versus how much damage it takes to destroy an asteroid. The player can also challenge themselves with a Hard game mode.

Theme: Graphics and Sound

The theme of this game is classic arcade space. However, it is ideal to upgrade it to a more modern look, but keep the feel of the game similar. All of the sprites were designed and created by myself (Kevin Wong), and hopefully bring out this theme along with the modern gameplay and features. The theme of the song hopes to enhance simple actions such as bullet shots to give a more pleasant experience. As well as a soothing background music to not distract the player.

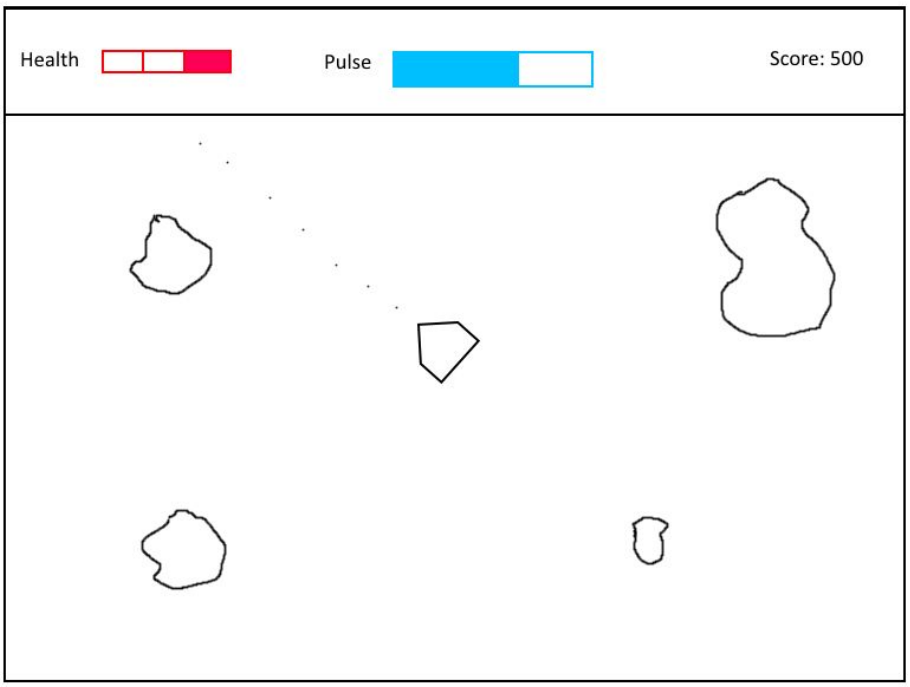
Interaction of player and game (Playing the game)

The interaction between the player and game will be on the keyboard. The player will begin with an initial game screen where they are able to start the game, look at instructions, or change game modes (Easy and Hard). They choose their option by using the arrow keys (up and down) and ENTER to selection. After the game has started, the player will use the arrow keys to turn (left and right), and to accelerate or decelerate (up and down). The player will use the spacebar to shoot and the ‘g’ key to activate the pulse which pushes all asteroids in an opposite direction if they are near the spaceship. The pulse is an ability that recharges after a specified amount of time. The player may also terminate the game at any time by pressing the ESC key, invoke a simple help module by pressing ctrl-h, or pausing/player the background music with ctrl-m. The small asteroids can be destroyed with 1 hit, while bigger asteroids take 3 hits, showing a sprite animation of their damage (cracks).

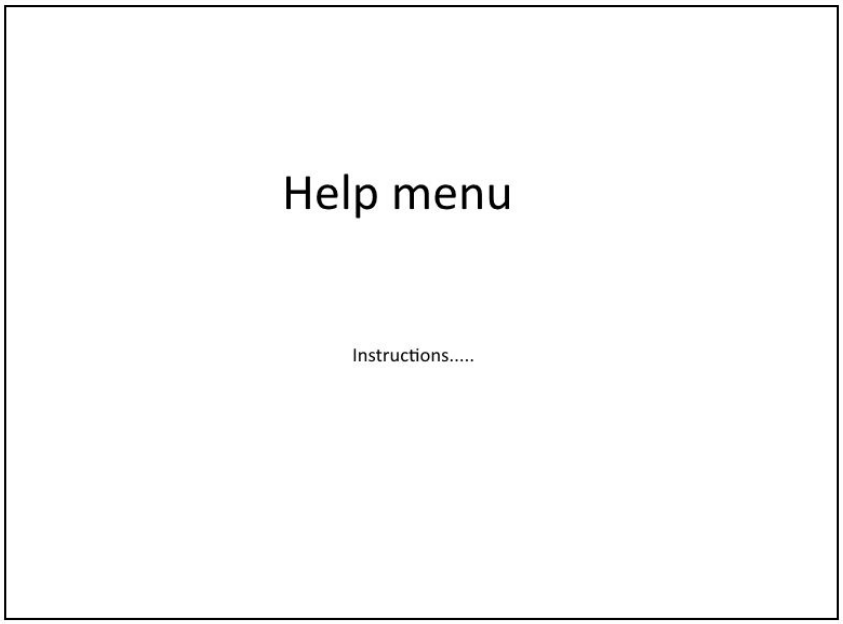
Basic menu layout and game options

The beginning screen will feature a welcome screen and instructions on how to select an option. The options will include ‘Start Game’, ‘Instructions’, and ‘Game Mode.’ Below is a general template of how it will look, but there will be improvement on graphics! Perhaps colors, background, and possibly animation.

Next if user selects ‘Start Game’ they can expect a game screen that shows the status of the player (health), score, and charge bar of the pulse ability on the top screen. While still retaining most of the screen for the actual gameplay of the spaceship vs asteroids.



This next screen is the “Instruction” screen which the user can get to by pressing ctrl-m or selecting it off the title menu. This will also contain Game info.



Story/Overview of enemies

The story is that you are tasked with asteroid defense duty. You must destroy as much asteroids as you can using your ship. Your ship is only strong enough to survive 3 hits from the asteroids, after that you are finished and must return to base to repair it. It is costly to repair a ship, that is why you must survive as long as possible before you have to repair it. Asteroids can be destroyed in 1 - 3 hits depending on its size. It will also vary in speed, so be on the alert for any fast moving ones. In the different game mode (Easy), you are in an asteroid belt where asteroids are unaffected by each other. However, in Hard mode, asteroids contain magnetic properties which interact with each other causing unusual and unpredictable effects. Your ship is able to fire at a constant rate and active a rechargeable pulse that can push away asteroids.

Feature: Pulse/Large Asteroids

One unique feature in the game is the rechargeable pulse on the ship. When you are in trouble with asteroids closing around you, you are able to press ‘g’ and it will activate a rechargeable pulse (progress of recharge show on top of screen). It will push all asteroids near you in an opposite direction. In addition, there are also large asteroids which take 3 hits to destroy. Each hit will trigger a change in sprite animation (cracks) to show the damage the current asteroid has taken. Although it takes longer to destroy, the defense patrol only counts of the number of asteroids destroyed, so it may be in your best interest to focus on the smaller ones.

Updated Feature: Datafiles

One updated feature in this game is the use of Datafiles to store the bitmaps and wav files. In the previous version, the source files were out in the open, which left it vulnerable to modification, copying, and stealing. By using datafiles in the form of data.dat, we are able to secure a majority of our files while still delivering our product in an efficient package.

Updated Feature: Multi-threading

Another updated feature of this game is the use of pthreads. In the previous version, all the work was done with the use of 1 thread, which meant it had to go through the entire process of updating, blitting, and checking different values, bitmaps, and sounds. Modern computers have multi-threading capabilities so it seems like a waste to not use it. With this updated version, we have used 4 different threads - One to handle asteroids, one to handle bullets, one to handle the update collisions, and one last one to handle the final cycle and inputs. By using these different threads the game now proceeds much faster and smoother without the symptom of choppy/super speedy gameplay. We also use mutexes to ensure no deadlock or race conditions happen between threads.

Conclusion

In conclusion, this game was definitely a great learning experience in using different routines that allegro has to offer. It has also pushed me to experiment, design, and create sprites from scratch and see what I am able to accomplish. Although there are still a bit of kinks

and bugs in this game, it is definitely interesting to observe and understand how the behaviour of the game acts because of it. In addition, the use of multi-threading really puts in the perspective of how useful threads can be. Though you have to be careful! However, I am still proud of what I have accomplished as it really seems like something I made myself (minus the sounds). Maybe in the future I'll record sound effects and see how that goes, but for now it was fun.

References

Sound Effects from Wavsource.com:

http://www.wavsource.com/snds_2017-09-17_1751672946049674/sfx/click_x.wav

http://www.wavsource.com/snds_2017-09-17_1751672946049674/sfx/phasesr2.wav

http://www.wavsource.com/snds_2017-09-17_1751672946049674/sfx/phasers1_x.wav

Background music from dl-sounds.com:

https://www.dl-sounds.com/royalty-free/star-commander1/