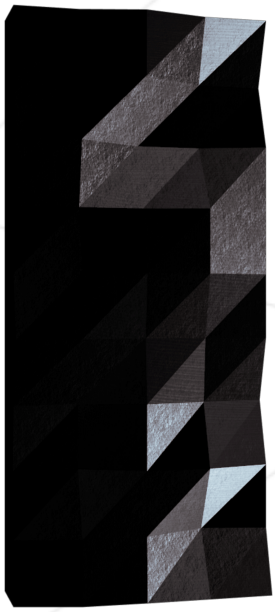

ft_shmup

Summary: You will create a small game using the ncurses library.

Version: 1

Contents

Chapter I

Preamble

Here's what Wikipedia has to say on Battlestar Galactica :

Battlestar Galactica is an American science fiction media franchise created by Glen A. Larson. The franchise began with the original television series in 1978, and was followed by a short-run sequel series (Galactica 1980), a line of book adaptations, original novels, comic books, a board game, and video games. A re-imagined version of Battlestar Galactica aired as a two-part, three-hour miniseries developed by Ronald D. Moore and David Eick in 2003. That miniseries led to a weekly television series, which aired until 2009. A prequel series, Caprica, aired in 2010.

All Battlestar Galactica productions share the premise that in a distant part of the universe, a human civilization has extended to a group of planets known as the Twelve Colonies, to which they have migrated from their ancestral homeworld of Kobol. The Twelve Colonies have been engaged in a lengthy war with a cybernetic race known as the Cylons, whose goal is the extermination of the human species. The Cylons offer peace to the humans, which proves to be a ruse. With the aid of a human named Baltar, the Cylons carry out a massive nuclear attack on the Twelve Colonies and on the Colonial Fleet of starships that protect them. These attacks devastate the Colonial Fleet, lay waste to the Colonies, and virtually destroy all but a small remaining population. Scattered survivors flee into outer space aboard a ragtag array of spaceworthy ships. Of the entire Colonial battle fleet, only the Battlestar Galactica, a gigantic battleship and spacecraft carrier, appears to have survived the Cylon attack. Under the leadership of Commander Adama, the Galactica and the pilots of "Viper fighters" lead a fugitive fleet of survivors in search of the fabled thirteenth colony known as Earth.

There is very little chance of you having a good grade on this project if you have not watched Battlestar Galactica in its entirety.

Chapter II

Introduction

The goal of this rush is to implement a simplistic shoot-em-up-style game in your terminal.

For those of you who don't know what that kind of game is (shame on you, by the way), have a look at Gradius, R-Type, etc...

You will use a 'screen' made up of a grid of 'squares', that you can equate to the characters on your terminal, so that the entities of your game are each represented by a character on screen.

Chapter III

General rules

- Your assignment has to be written in C or C++.

`It is strongly recommended to choose the C++ to do this rush. You can use any version of C/C++ you want.`

- No norm.
- `cc or c++` is used as compiler.
- You have to compile with the following flags `-Wall -Wextra -Werror`.
- You have to turn in a `Makefile` which will compile your source files.
- Your program should not quite unexpectedly (segmentation fault, bus error, double free, and so forth) except for undefined behaviors.
- Within the mandatory part, you're allowed to use the following functions:
  - Every functions in the standard library.
  - Any clock-based library.
  - The ncurses library.

Chapter IV

Instructions

Here are the basic requirements :

- The program name is `ft_shmup`.
- The game is single-player.
- The display must use the ncurses library.
- There must be a horizontal or vertical scrolling (The screen area moves through the world, very much like in R-Type for example).
- There must be a multiple random basic enemies.

`A basic enemy is an enemy that does not use any special abilities. This enemy doesn't even need to move or aim when he is going to shoot.`

- The player can shoot at enemies.
- Your game must have a basic collision mechanic (If an enemy touches you, you die).
- The game entities must occupy one 'square' of the map only.
- The game should have a frame-based timing.
- Displaying score, time, number of lives, etc... on screen.
- Clock-based timing (Use whichever system facility or library you like)
- Enemies can also shoot.
- Scenery (Collidable objects or simple background)

It may seem a little daunting, but the basic requirements can be fulfilled by a game even simpler and way uglier than the already pretty simple **Space Invaders**. So it's not actually that hard.

For the student who do this project in C++, there will be plenty of occasions to use the various facilities of C++. For example, representing the various entities of your game (Player ship, enemies, missiles, etc ...) as subclasses of a GameEntity class ? Maybe even an interface ?

For those of you who have never made a video game, here's how a very basic game should run:

- Acquire input (Player controls, network, etc ...).
- Update game entities.
- Render display.
- Repeat until game ends !

Of course, the "Update" phase includes a lot of things, including handling the spawning of enemies, making actions required by player input, moving entities according to their current vector and the time since the last update ... But those are the basics. When you get down to it, a basic video game is a really simple program. It should not be too hard to do, should it ?

When you're pretty sure you've done everything right, you are very welcome to improve on the basic requirements and make an even better game.

Chapter V

Bonus Part

Now it's time to improved your beautifull game, here are some ideas of cool bonuses, but keep in mind that we'll consider them useless if you haven't done the other stuff before:

- Large and hard-to-beat boss enemies.

Large simply means that the enemy will take up more space on your board. It will have more life like a real boss in some games.

- Enemies with a scripted behaviour.

An enemy with a script will have additional abilities such as a move in the direction of the player or a better way to shoot at the player.. The choice is yours!

- Multiplayer, either on the same keyboard or through the network if you're feeling cocky.
- Scripted game worlds, with pre-determined batches of enemies.

The bonus part will only be accessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

Chapter VI

Turn-in and peer-evualation

As usual, turn in your work on your repo GiT. Only the work included on your repo will be reviewed during the evaluation.

Good luck!