Design Document F-Type

By Hassan Al-Awwadi and Stefan van der Sman

Inhoud

1. Introduction and Rules

This document describes the design of our game F-Type. F-type is a side scrolling shoot em up with a variety of enemies, and bullet types. The inspiration of the game is R-Type and touhou.

In this game you play as a spaceship that must destroy as many enemies as possible in order to protect your homeworld and get a highscore! If they hit you, you restart the level and lose a life. 0 lives = GAMEOVER.

You start the game at 3 lives and relatively low power, but every enemy you destroy has a small chance to spawn a powerup.

We made the game in Stack, and to build it you can simply do a stack install. To run it you can do stack exec FType-exe after you've installed it.

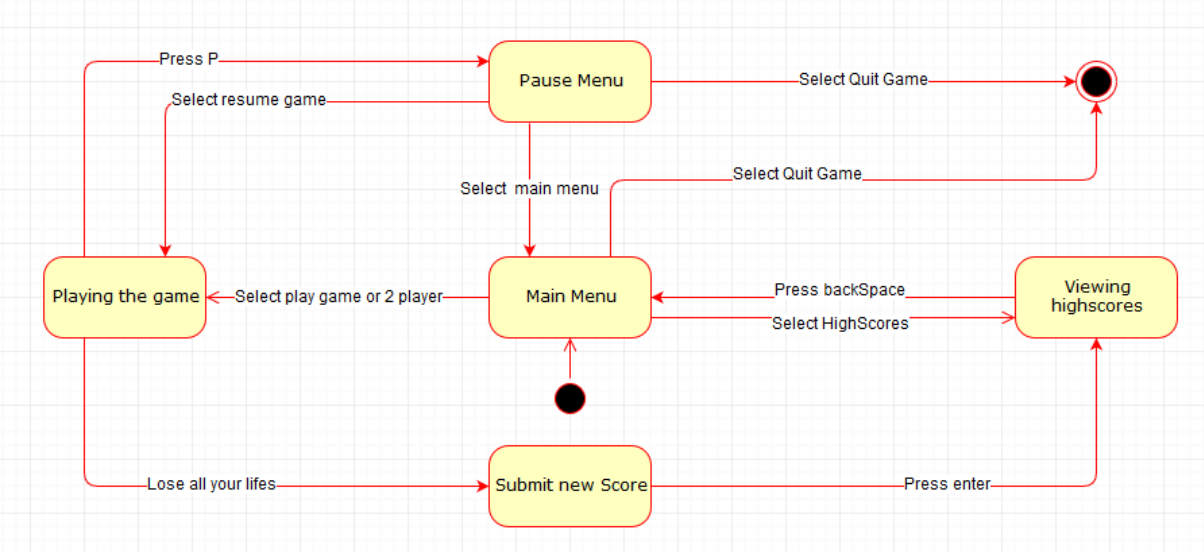
2. GameFlow

when you start the game, you enter the main menu. There you can select what you want to do. You can either Play the game on your own, with 2 people, view the current highscores or quit the game.

If you chose to play the game (on your own or with a friend!) you start the infinite wave game mode. Enemies will spawn periodically, and you will have to survive the onslaught. You have 3 lives (2 player mode, you have 3 lives shared) and once all your lives are gone you will be able to submit your high score and see how well you have done.

If you press P you will enter the pause menu, from which you can quit the game, go back to the main menu, or re-enter the game after taking a short break.

If you selected highscore in the main menu or have lost the game and submitted your new score, you will be sent to the high score screen. Press backspace to go back to the main menu or w and s to scroll up or down through the names.



3. Data Structures

The state of the game gets represented in our gamestate data type:

```
data Game = Playing { world :: World}
            | Menu { menu  :: Menu (IO Game) }
            | HighScores { mvps  :: [(String, Float)], scrollUp ::
Bool, scrollDown :: Bool, scrollDelta :: Float, stat ::
StaticResource, dyn :: DynamicResource }
            | NewHighScore { nameWIP :: String, score :: Int, stat
:: StaticResource, dyn :: DynamicResource }
```

the state of the world gets represented in our world data type:

```
data World = World{
    player :: S.Ship,
    enemies :: [E.Enemy],
    lives :: Int,
    score :: Int,
    level :: Int,
    timer :: Float,
    powerUps :: [PowerUp],
    state :: WorldState,
    stat :: StaticResource,
    dyn :: DynamicResource
}
```

There is one ship controlled by the player, multiple enemies, the amount of lives before game over, the level and the current gamestate.

The ship that the player controls is represented in this data type:

```
data Ship = Ship{
    pos :: Point,
    speed :: Float,
    direction :: Vector,
    gun :: Gun,
    bullets :: [Bullet],
    bombs :: Int,
    size :: (Float,Float),
    borders :: Border,
    anim :: [Picture]
}
```

The details of the shooting method get represented by a Gun type

```
data Gun = Simple{
     cal :: Float,
     power :: Float,
     cooldown :: Float,
     countdown :: Float,
     speed :: Float,
     color::Color }
         | SpreadShot{
     cal :: Float,
```

```
      power :: Float,
      cooldown :: Float,
      countdown :: Float,
      speed :: Float,
      amount :: Int,
      angle :: Int,
      color::Color }
```

The details of the bullets get represented by the Bullet type:
data Bullet = Bullet { size :: Float , pos :: Point, speed :: Float, direction :: Vector, dmg :: Float, color::Color } deriving(Show, Read)

The enemies get represented in the following data type, gravemarker is a placeholder for dead enemies to keep bullets on screen:

```
data Enemy = Enemy{
      size :: Float,
      pos :: Point,
      speed :: Float,
      direction :: Vector,
      health :: Float,
      gun :: Gun,
       bullets :: [Bullet],
      deathAnim:: [Picture],
      cullRange::Border }
            | GraveMarker {
       pos :: Point,
      bullets :: [Bullet],
      deathAnim::[Picture],
      cullRange::Border }
```

Menus gets represented in the following data type:

```
data Menu a = Menu { menuItems :: [(String, a)], maxSelected :: Int,
selected :: Int }
```

Resources get represented in the following data types:

```
data StaticResource = StaticResource   { explosion :: [Picture],
playerShip :: [Picture], border::Border }
data DynamicResource = DynamicResource { rng :: StdGen}
```

4. Interface

The interface of the game is a side scrolling game in which enemies appear. The player appears as a ship. There is also a main menu and a high score screen. You can select an option with the W and D keys and confirm it with ENTER.

Once you are playing the game, player 1 controls his ship with wasd, and player 2 with ijkl. Shooting happens automatically. P will enter the pause menu that works in the same manner as the main menu.

Once you die 3 times you will be required to enter your name for score keeping. Type in using your keyboard in order to input the name. You can use the backspace key when you make a mistake and enter to finalise your name.

In the highscore screen you can use ‘w’ and ‘s’ to scroll up and down and search for your position.

5. Requirements

5.1 Minimal requirements

Player

The player can control the direction of the spaceship with the WASD keys and shoots bullets automatically

Enemies

The enemies move in the direction of the player. There are two different kinds of enemy, one with a normal shot, and one with a spread shot. A death animation is displayed when the life total is 0. Enemies can drop power ups on death.

Randomness

The enemies randomly drop random power ups. The enemies that get spawned are random as well.

Animation

Enemies display an explosion as death animation. Ship has a flying animation. There are menu animations.

Pause

Everything in the game freezes and saves the state of everything when you press P and the menu appears, when the game continues everything continues moving according to the saved states.

Interaction with the file system

Highscores are saved in a file, you can enter your name once you die to add your highscore to the list

5.1 optional Requirements

The player and enemy explosions contain complex graphics
There are 2 kinds of enemy, a red one with standard bullets, and a green one with spread shot and 1.3 times as much health
There is a multiplayer mode, enemies target the closest player, the second player is controlled with the IJKL keys