Functional programming design document

Lex Klaassen en Viggo de Breij

1. For the final project, we will implement a shoot’em up using a model view controller structure for our game.

2. We have not planned any twist at this moment

3. These are the data types we have planned for now:

data GameState = GameState { player :: Player ,

enemies :: [Enemy],

score:: Score,

probably more for example background details and such}

data Player = Weapon Position Velocity Lives

data Weapon = Peashooter | Launcher | Laser

data Power = Upgrade Point

| Health Point

data Enemy = Swarm Lives Point

| Turret Point

| \*\*Worm\*\* \*\*Lives\*\* \*\*Point\*\*

| Boss Lives Point

data Point = Pt Float Float

data Score = Score

4. Minimal Requirements

* Player: het ruimteschip, deze zal in eerste instantie omhoog en omlaag kunnen bewegen (we gaan kijken naar alle richtingen), kan schieten.
* Enemies: Er zijn meerdere tegenstanders, ze schieten af en toe op de speler, sommigen kunnen bewegen op een andere manier dan anderen. De hoeveelheid enemies hangt af van de score van de speler
* Randomness: Waar de vijanden spawnen is semi-willekeurig, welke vijand gespawnt wordt is ook random
* Animation: de dood leidt tot explosies, het hele spel is een animatie als het maar snel genoeg update naar de volgende staat.
* Pause: De speler kan op de toets “P” drukken om het spel op pauze te zetten en als deze dat nog een keer doet gaat het spel verder.
* IO: Je kunt de gamestate opslaan naar een json bestand die je weer kunt ophalen als je bij het menu bent. Tevens is er ook een scorebord waar we de hoogste scores in een tekstbestand opslaan.

5. Optional Requirements: (Vooral parallax en different enemies, rest is voor als we snel bezig zijn)

* Multiplayer: twee onder elkaar en degene die het het langste volhoud die wint. Player 1 kan de muis of pijltjes gebruiken en Player 2 kan de wasd gebruiken
* Parallax: gaan we implementeren om de achtergrond objecten met verschillende snelheden te laten bewegen door praktisch ze te implementeren alsof het vijanden zijn die geen interactie hebben met de speler of de omgeving
* Levels: Verschillende enemies en bosses maken voor verschillende levels.
* Different enemies: We gaan hoe dan ook verschillende vijanden implementeren, denk aan een horde van kleine vijanden en grotere langzamere monsters.
* Mouse Input: We gaan het zo doen dat men de muis kan gebruiken om het ruimteschip rond te manoeuvreren. Ook kun je schieten met de muis. (is misschien broken/op)

6. We willen het pure en het impure van elkaar gescheiden houden door het impure zo veel mogelijk in een aparte module te zetten.

7. Alle bewegende object kunnen waarschijnlijk wel wat eigenschappen delen bv. speler, vijanden of achtergrondobjecten hebben allemaal wat overeenkomende eigenschappen die we mogelijk tegelijkertijd kunnen definiëren.

Inspiratie:

https://stackoverflow.com/questions/52871673/haskell-gloss-do-something-every-frame-key-is-pressed