Functional programming design document (Updated)

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 implemented a twist.

3. These are the data types we have implemented:

data GameState = GameState {

status :: Status,

infoToShow :: InfoToShow,

keys :: String,

timer :: [TimerFreq],

player :: Entity,

enemies :: [Entity],

background :: [Entity],

score :: Score,

elapsedTime :: Float -- heb het eigenlijk niet nodig

}

data TimerFreq = T EntityTypes Time Freq deriving (Generic, Show, Eq)

type Time = Float

type Freq = Float

type Damage = Float

type Score = Int

type Health = Float

type Direction = (Float, Float)

type Size = Float

data Pos = Pt Float Float deriving (Generic, Show, Eq)

type HitBox = (Pos, Size)

type Player = Entity

type Enemy = Entity

type Bullet = Entity

data Entity = E {

entityType :: EntityTypes,

health :: Health,

hitbox :: HitBox,

weapon :: Weapon,

damage :: Damage, -- on collision with another entity

direction :: Direction,

rate :: (Time, Freq),

bullets :: [Bullet]

}

data EntityTypes =

Player | Worm | Swarm | Turret | Boss

| Pea | Rocket | Laserbeam | Explosion

| Cloud | Mountain | Planet

deriving (Generic, Show, Eq)

data Weapon = None | Peashooter | Launcher | Laser deriving (Generic, Show)

data Status = StartScreen | Game | Pause | GameOver deriving (Generic, Show, Eq)

4. Minimal Requirements

* Player: het ruimteschip, kan in elke x y richting bewegen, kan schieten.

Dit doen we door elk moment bij te houden welke toets er is ingedrukt, en elk frame aan elke ingedrukte toets een effect toe te schrijven.

* Enemies: Er zijn meerdere tegenstanders, ze schieten af en toe op de speler, sommigen kunnen bewegen op een andere manier dan anderen.  
  We implementeren dit door entities te generen die en schade doen op contact en kogels schieten op de speler, dit zijn allemaal eigenschappen van het entity type
* Randomness: Waar de vijanden spawnen is semi-willekeurig, welke vijand gespawnt wordt is ook random.   
  In controller roepen wij de random functie randomRIO aan die met behulp van van te voren bepaalde variabelen de positie en frequentie van de vijanden bepaald.
* Animation: Als een raket contact met een vijand, explodeert de raket, ook explodeert de Brute, dit hebben we gedaan door explosies een entity te maken die per tick een aantal levens verliest, en aan het aantal levens koppelen we een bijbehorend frame van de explosie.
* 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.   
  Als het spel op pauze staat wordt dezelfde onveranderde game state teruggegeven tot het spel doorgaat of er wordt gequit naar het startscherm. Ook kan de speler ervoor kiezen om zijn vooruitgang te bewaren door op de “S” toets te drukken. In dat geval wordt de huidige gamestate opgeslagen naar een JSON file in de saveFile folder.
* IO: Je kunt de gamestate opslaan naar een json bestand die je weer kunt ophalen als je bij het menu bent.

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

* 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.
* Different enemies: er zijn 3 verschillende vijanden, de Swarm, Brute en Turret, allemaal met verschillende vuurpatronen.  
  allemaal een ander type entity dus we typechecken overal waar het relevant is om de verschillen tot uiting te laten brengen.

6. We hebben het grootste gedeelte van de impure code in controller en view, dit heeft er mee te maken dat input uiteraard impure is, en om relatief te werken met de grootte van het scherm in view moet je ook wat IO gebruiken.

7. We hebben geen type classes gebruikt, wel hebben we goed gebruik kunnen maken van entity omdat veel van de objecten een groot gedeelte van de eigenschappen delen.