# FP 2017 Design Document

Bug HHunt

**L. Werkman and N.Hendrikx**

Computer Science

Utrecht University

Year 2017/2018

# Contents

## Introduction

Dear Reader,
This document covers our concepts of our game of functional programming. The team members are Lars Werkman and Niels Hendrikx.

# Chapter 1

# Logical entities in the game

This chapter will cover all the subjects of the mandatory part of this project.

## 1.1 Bug HHunt

Our game is called "Bug HHunt". The extra 'h' stands for Haskell. This is a shooter based on the famous game Duck Hunt.

## 1.2 General

You see a screen with or without flying object. These objects move randomly across the screen and change directions randomly. Their movement speed is increased every level. The player is allowed to miss 2 bugs, at the third the game is over. In this story the bugs are killing your computer. Every time you do not kill a bug your computer will get an malfunction. At 3 bugs your computer is completely ruined. The goal is to reach the highest level possible and kill as much bugs as possible. But there are infinite amount of levels.

## 1.3 Player

The player has to aim at bugs using arrows and shoot using space-bar. This is done by the keyboard.

## 1.4 Enemies

As usual the bugs are the enemies. These are trying to break your computer. They move random.

## 1.5 Randomness

The game has to include random components.

1. The enemies move in random directions.

2. The bugs have randomly generated frequency and coordinates.

## 1.6 Animation

We will make an background that is animated. The bugs animated. Also the bug collector is animated. The bug collector is a Haskell expert who collects the bugs the player kills. Also the effects when you not kill a bug is animated.

## 1.7 Pause

You can pause the game by pressing the key 'p'.

## 1.8 Interaction with the file system

We will have to safe high-scores on the file system.

# Chapter 2

# Important data types and type classes

This chapter will cover some of the most important data types and type classes. On the end of this chapter we give an example in code.

## 2.1 Data types

We have the folowing data types:

- Monster type which can be a Orc, Zombie, Vampire or Dracula.
- Player
- Graph

## 2.2 Type classes

we have the following type classes:

- RoomId
- Health
- Damage
- Score

Listing 2.1: data types and type classes

```haskell
type Size = (Int, Int)
type Resource = (Size, Bitmap)
type Coords = (Int, Int)

class Drawable d where
  draw :: d -> Picture

data Bug = Bug {
  resource :: Resource,
  velocity :: Int,
  angle :: Float,
  coordinates :: Coords
} deriving (Drawable)

data Player = Player {
  resource :: Resource,
  coordinates :: Coords
} deriving (Drawable)

data Collector = Collector {
 resources :: [Resource],
 state :: Int,
 coordinates :: Coords
} deriving (Drawable)

data Level = Level {
  missed :: Int,
  bugs :: [Bug]
}

data Game = Game {
  score :: Int,
  level :: Level,
  player :: Player
}
```