# Snake

Alessandra Sasanelli, Simone Maccario

November 16, 2023

**Abstract**

In this document we would like to discuss our proposal for the PF1 project. We decided to reproduce the "Snake" game; here we will explain our intentions and ideas for implementing this game in the racket ASL language.

## 1 The game

Our aim for this project is to recreate the Snake game. This game consists of 3 basic elements, which are a snake, an apple and the background: on the background, the snake moves at specified pace whilst the apple is positioned at available random positions at the starting of the game or if the apple has been eaten. The objective of the game is to get your snake eat as many apples as it can to become bigger. The user looses the game if the snake hits the border or it hits itself. To interact with the application, the user can use the keyboards' buttons arrow "up", "down", "left", "right", the "r" button and the "escape" button. The firsts four buttons are needed to change the direction of the snake; the "r" button is, instead, needed to start a new game if you lost the previous game; the escape button is needed to quit the application.

## 2 Libraries

To implement our project we will mainly use 2htdp/image and 2htdp/universe.

## 3 Data types

To construct our appstate we thought to include these elements:

- a data type that represents the snake
- a data type that represents the apple
- a data type that represents whether the game has been lost or not
- a data type that represents whether the application has been quit or not

In particular, the snake data type will be a structure and will preserve key informations about the snake itself, meaning that it will represent the length of the snake at a particular moment, the position of the head of the snake and the direction of the snake itself. We will create a basic unit for the snake that will be needed to render the snake on the background; in fact, the length of the snake will simply represent how many of these basic units will be needed to draw the snake.

We thought of the apple as simply being the position of the apple; this way, the draw function will draw the apple at he specified position and, if the head of the snake and the apple are at the same position, the position of the apple will be changed.

To decide wether the game has been lost or not, we will use a simple boolean; when you are playing and the game hasn't been lost, this will be true, whilst loosing the game will change it to false. Clicking on the "r" button will make this again true if it was false, making the whole appstate back to its original state, starting a new game.

The "quit" will be just as the other type we described before, but in this case, this element will stop the loop, making the user stop the application.

# 4 About functions

To implement this game, we will use the big-bang function. In particular, we will need these core functionalities of the big-bang:

- the on-draw
- the on-tick
- the on-key
- stop-when

For the on-draw, we will implement a draw function that will draw the whole appstate. This function will be divided in different auxiliary functions to first draw separately the different component of the game: we will implement a function to draw the snake based on the snake structure and a function to draw the apple. These two will be combined together to obtain the final result.

The on-tick function will need instead two functions: one to change the position of the snake on-tick and the other to change the rate of the clock as the game procedes