GameDesign Pacman

106: Thomas Kuhlemeier & Sam Leurink

1. Introduction and Rules

For our project we are going to be attempting to recreate the famous game: “Pacman”.

The game consists of a maze in which the player, in the original depicted as a yellow circle with a mouth, has to eat all the food that is scattered around the maze. The player always goes forward(except when there is a wall in front of the player) unless turned left or right at an intersection. The game also has enemies who are called “Ghosts” whose target it is to eat the player. In the original there are 4: 1 who chases the player at all times, 2 who try to cut the player off at any given time and 1 who changes between chasing the player and fleeing from it.

There are two types of foods in the maze. The standard one which fills the entire maze and what in the original game is called “Power Pellets”. These “Power Pellets” are in every corner of the maze and allow the player to eat the enemies instead of them eating him. After being eaten the enemies turn into ghosts and have to go back to their starting positions in the middle of the maze.

The player wins if all the food in the maze is eaten and the player loses when the player has been eaten by the ghosts three times, since the player has only three lives.

2. Design

2.1 Data structures

2.1.1 The Game State

```
data Lives = One | Two | Three
type Score = Int -- is this type allowed? (if not, why not?)
type Level = Int -- and this?
data Paused = Yes | No -- and this?

data GameState = { lives :: Lives
                 , maze :: Maze
                 , score :: Score
                 , level :: Level
                 , paused :: Paused
                 }
```

Also, we would like to make a starting menu with a maze selection part, but how can we do this? And we might need a timer to count for how long the Ghosts have been in the dizzy state and for how long they have been regenerating etcetera etcetera. We hope that this design is not too OO-like, and if it is, we would gladly hear how we can improve on this.

2.1.2 The Player and Playing field

```
data Ghost = Blinky Position Direction Ai Status | Pinky Position Direction Ai Status
                | Inky Position Direction Ai Status   | Clyde Position Direction Ai Status
data Pacman = Pac Position Direction
data Maze = MkMaze Pacman [Ghost] [Circle] [Wall]
data Position = Pos Float Float
data Status = Normal | Dizzy | Eaten | Regenerating

Circle = Snack Position | Energizer Position -- (maybe also a Bool to indicate whether or not a Circle has been eaten, we're hesitating between this, or to update the list of circles everytime a circle is eaten by Pacman (but then we have to search the list which could make it slower?)?)

data Wall = MkWall Position Direction
data Ai = MkAi [Direction] -- A list of Directions which can be mapped to different states of the environment of a Ghost, so that the Ghost knows what direction to move in corresponding to which environment / or situation.
data Direction = North | East | South | West
```

2.1.3 Player Movement

Pacman can be controlled by using the updownleftright keys to change its direction. Pacman will automatically move forward, just like the Ghosts. Pacman automatically stops moving when he collides with a wall, unlike the Ghosts, which avoid the walls automatically by changing their direction with their artificial intelligence. Pacman automatically starts moving forward again if the player changes its direction to be away from all walls surrounding it directly.
The game will be momentarily stopped when a Ghost and Pacman collide and then start over and update the Lives data type. However, the Ghosts will be briefly stopped when it collides with Pacman if they are in a "dizzy" state. This is when they turn blue and can be eaten by Pacman. In this dizzy state the Ghosts are also slowed down in their forward movement. When a dizzy Ghost collides with Pacman, we say that the Ghost is eaten, and then it will disappear and respawn in the middle of the screen in a cage where it regenerates for a while until it gets released from the cage and starts its normal behaviour again.
Maybe we will have an eaten Ghost find its own way back to the cage whilst it has the appearance of floating eyes, however we're not sure yet whether and how we will implement this. If we don't implement it, the Eaten constructor of Status is useless and we will throw it away. All movements will be done smoothly with the Gloss library we presume.

2.2 Computer and AI

In the original game there are 4 enemies. Each of them having their own characteristics. As said in the introduction there are 3 types of enemies: The direct chasers, the predictors and the unsure. All of these will use a shortest route algorithm to find the fastest way. For this we are looking to use BFS or Breadth First Search. BFS would be of type:

```
Bfs :: Maze -> Ghost -> Pacman -> Direction
```

2.3 Interface

Graphical interface implemented with the Gloss library. It will show a window which at first contains a little menu with a Maze selection section and a play button. When the play button is pressed/ clicked the game will start and the player will see the all too familiar layout of the Pacman game. The player will also see the current score, lives, and level in the game.

2.4 Implementation of the Minimum Requirements

Player The player will be controlling pacman and can use the keys left- and right-arrow to move left and right.

Enemies The computer will control the four enemies using the things described in 2.2 Computer and AI

Randomness The randomness will be implemented in the use of Clyde (The unsure enemy) which will have a chance to change from fleeing to chasing and from chasing to fleeing.

Animation Animation can be added in multiple ways. We're mainly looking at the eating animation and the death animation. Also there needs to be a way where ghosts are distinctly in either the dizzy or the chasing state.

Pause We will use the 'p'-key as the pause button, which will stop both the players and the ghosts moving.

IO Our IO function will be writing highscores to a document.

2.5 Implementation of the Optional Requirements

#1: Maze Selection (different shapes and sizes of the Maze layout) and
#2: Four different kinds of enemies (which have Ai ranging from simple to more complex). We want to make it such that each Ghost has its own characteristic AI (artificial intelligence), see the wiki for this info. Also we could be interested in making an antagonist mode, in which you have a player, an AI controlled ghost(probably the chaser) and the antagonist, whose goal it is to eat the player.