# Design document – Final Project FP 2018/2019

Name: Oscar Müller & Christa Meelker

Student number: 6261906 & 5507332

Game: Pac-man

## Game description

Pac-man is one of the most iconic games ever made, which makes it easy to find documentation on the inner workings of the game itself. For the construction of this document we have chosen to use the [following website](http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior) which we think goes in to great detail on how the classis Pac-man works and looks.

### Player

The player in this game controls Pac-man. A yellow circle with a moving jaw/mouth whose goal it is to “eat” all of the yellow dots to continue to the next level. Besides yellow dots the maze also contains four energizers Pac-man can “eat” to make the ghosts go into “Frightened Mode”. More on that later. Movement will be handled by the keyboard using a combination of four buttons for up, down, left, and right. This combination can either be the WASD-keys or the arrow-keys. The maze also contains two tunnels the player can go through to teleport to the other side of the maze

### Enemies

Pac-man has a total of four enemies. The Red Ghost nicknamed “Blinky”, The Pink Ghost nicknamed “Pinky”, The Blue Ghost nicknamed ”Inky”, and The Orange Ghost nicknamed “Clyde”. Besides name and color, the four ghosts of Pac-man also differ in starting position in the maze and movement algorithm. Each ghost has a unique algorithm that comes in to play each time they come across a junction and decides which way they will go to catch Pac-man. We won’t go into all the details in this document, but the aforementioned website carefully details each ghosts behavior. The final thing that has to be coded to completer each Ghosts’ behavior is their Movement Modes. Ghosts have 3 movement modes, namely Scatter, Chase and Frightened. Each level has four waves of alternating Scatter and Chase modes and the Ghosts’ are entered in to Frightened mode if the player eats a energizer. In Scatter mode each Ghost has a predetermined tile outside the maze it is trying to move towards, but because it is outside of the maze, the Ghosts will never reach this tile and loop in each corner of the maze trying to reach it. Chase mode is the standard mode and is the mode the Ghosts stay in indefinitely after the fourth wave to chase down Pac-man. Frightened mode is the mode the Ghosts enter when Pac-man eats a energizer and causes the Ghosts to change appearance, slow down, and run away from Pac-man. While in this mode it is also possible for Pac-man to eat the Ghosts.

### Randomness

The original Pac-man game as it was designed by Toru Iwatani contains no randomness. The maze is the same for each level, including the placement of the energizers and the only thing that is different from each level is the duration of the Frightened mode, which becomes increasingly shorter as you progress through the levels. Elements of randomness can be added however. The starting position of each ghosts (inside or outside of the maze) and the spawn position of the four energizers can be randomized without straying too far of the original design of the game. A much bigger challenge would be to have randomized mazes, but that may fall out of the scope of this assignment.

### Animation

Besides the normal animation of the regular movement of Pac-man and the Ghosts, there is not a whole lot of animation to be found.

### Pause

Pausing will be done by means of a press of a button on the keyboard and the game will be continued after the same button is pressed again.

### Interaction with file system

The only sensible implementation of this constraint would be a high score table. After a player dies he/she can input their name which will be written to a file for everyone to see.

## Game structure

### Data types

One of the data types we need is the data type Direction, which consists of the four major directions, namely up, down, left, and right. The four ghosts and Pac-man himself will always have a direction they are facing, which will mainly be used for the algorithm of the Ghosts to decide whether they will turn left/right at a junction in the maze.

You play Pac-man in a maze, but we think it is a good idea to make the maze into a grid. This grid is made out of a list of rows and each row is a list of fields. For fields we are going to use records. We will be using records so we can update part of a field (per example when Pac-man eats a food-dot the content of the Field changes from having a FoodDot to being empty). Fields will also have a type which will be used to determine what happens when you interact with them (e.g. when you press up at a junction something else happens than when you press up at a Straightaway).

Ghosts and Player are both data types. These data types contain the position of the entity and its status. The player can either be energized or not and a Ghost can be in one of three modes, namely Scatter, Chase or Frightened. Position is also a data type of its own and consists of two integers for the row number and column number in the maze.

data Pos = Pt Int Int

data Player = Player { position :: Pos, status :: PlayerStatus }

Deriving (Drawable, Collision)

data Ghost = Ghost { position :: Pos, status :: GhostStatus }

Deriving (Drawable)

data PlayerStatus = Energized | Neutral

data GhostStatus = Chase | Scatter | Frightened

data Maze = [MazeRow]

data MazeRow = [MazeField]

data MazeField = MazeField { field :: FieldType, content :: ContentType }

data FieldType = Wall | Straightaway | Junction

data ContentType = FoodDot | Energizer | Empty

Deriving (Drawable, Collision)

data Direction = Up | Down | Left | Right

### Type classes

So far, we have decided on two distinct type classes. There are a couple of data types that should behave in the same way. Firstly, all the elements that should be drawn belong to the data type drawable. The instance for Player should in this case be a big yellow circle. The instance of a FoodDot should be a smaller circle.

Secondly, some elements can “collide” with each other. Whenever such a collision happens, depending on the types, something should happen. For instance, when a player collides with an Energizer, the state of player should change. Likewise, when a player collides with a FoodDot, the FoodDot should be removed from the MazeField.