# 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. [This website](http://www.gamasutra.com/view/feature/132330/the_pacman_dossier.php) was also consulted multiple times and the document found in this website is one of the sources of the first mentioned website.

### Player

The user in this game controls Pac-man. Pac-man is drawn as a solid yellow circle and can be moved using the arrow keys. Pac-man is confined to the maze and cannot go through walls. When Pac-man walks over a foodDot (represented by a pinkish dot in the maze) he will eat said foodDot and earn ten points. If Pac-man collides with a Ghost, the position of Pac-man and all the Ghosts will be reset and the game will continue as normal, but the player will lose a life.

When the player collides with a Ghost three times he will have lost all his lives and he will have lost the game. A message will be displayed notifying the user that he has lost and the application can be restarted to play again.

When the player successfully eats all the foodDots without losing all his lives, a congratulatory message will be displayed and the application can be restarted to play again. For score writing to the file when the game ends in one of these two ways see “Interaction with file system”.

The maze also contains four Energizers (represented by a pinkish circle in the maze). These Energizers should make Pac-man go into Energized mode and the Ghosts into Frightened mode, but this functionality wasn’t implemented.

### 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 ghost’s 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 an 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 an 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

The user can press the “F1” key on the keyboard to pause the game. When the game is paused, the screen will display a message saying that the game is paused and to press the “F1” key again if the user would like to continue to play. When the game is paused, the enemies and player will not move.

### Interaction with file system

The folder where the game is located in contains a text document called “highscore.txt”. When the game hasn’t been played the only thing this file will contain is the number 0. The game can end in one of two different ways: the player eats all the foodDots on the screen or the player loses all his lives. In both cases the score in the “highscore.txt” file will be read, but will only be overwritten if the score gotten from the GameState when the game ends is higher than the score in the file.

## 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 GameState = GameState { pacman :: Player, blinky :: Ghost, maze :: Maze, score :: Int, status :: Gamestatus }

deriving (Show)

data GameStatus = GameOn | GameLost | GameWon | GamePaused

deriving (Eq, Show)

data Player = Player { playerPosition :: Point, playerDirection :: Direction, playerStatus :: PlayerStatus, playerSpeed :: Float, playerLives :: Int }

deriving (Show)

data Ghost = Ghost { ghostPosition :: Point, ghostDirection :: Direction, ghostMode :: GhostMode, ghostSpeed :: Float }

deriving (Show)

data playerStatus = Neutral | Energized

deriving (Show)

data GhostMode = Chase | Scatter | Frightened

deriving (Show)

type Maze = [MazeRow]

type MazeRow = [MazeField]

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

deriving (Eq, Show)

data FieldType = Straightaway | Intersection | Wall | GhostWall

deriving (Eq, Show)

data ContentType = FoodDot | Energizer | Empty

deriving (Eq, Show)

data Direction = FaceUp | FaceLeft | FaceDown | FaceRight

deriving (Eq, Show, Ord)

instance Num Point where

(x0, y0) - (x1, y1) = (x0 - x1, y0 - y1)

(x0, y0) + (x1, y1) = (x0 + x1, y0 + y1)

### 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.