**USER MANUAL**

1. **PacGame**
   1. **Characters**
      1. **PacMan**
      2. **Ghost**
         1. **Sam**
         2. **Cam**
         3. **Danny**
   2. **PacWorld**
      1. **PacDots**
      2. **PowerPellets**
   3. **PacGame**
2. **MouseGame**
   1. **Characters**
      1. **MouseHero**
      2. **Mouse**
   2. **MouseLand**
      1. **MouseTrap**
      2. **Exit**
   3. **MouseGame**
3. **PipeGame**
   1. **“Characters”**
      1. **Plumber**
      2. **Water**
   2. **PipeMap**
      1. **Pipe**
   3. **PipeGame**

**4. Engine**

**4.1. Classes**

**4.2. Game Play**

**1. PacGame**

Based on the original ‘PacMan’, PacGame is a tile based and slow-motion type game which requires the character, PacMan, to collect all the PacDots on the grid, while avoiding or killing Ghosts.

**1.1. Characters**

The characters are basic. The hero is PacMan, the enemies and Non Player Characters (NPCs) are the Ghosts.

**1.1.1. PacMan**

PacMan is the hero of this game. He is controlled by the user and is followed by the ghosts. PacMan eats PacDots to gain points and to complete the game, and may also eat PowerPellets to become invulnerable and thus gain the ability to eat ghosts.

**1.1.2. Ghost**

PacGame has a population of Ghosts that are set out to catch and kill PacMan. In order to effectively do this, each Ghost has its own personality, characterized in the original PacMan. The names of these ghosts are Blinky, Pinky, Inky and Clyde (source: http://en.wikipedia.org/wiki/Pac-Man). In this version, there are only three ghosts, with slightly distinct personalities (**not implemented yet**).

**1.1.2.1 Sam**

Sam is the sceptical Ghost, characterized by its (:S) face. It is always contemplating possible moves in order to cut off PacMan. You could say he is the smartest ghost of all and he is most likely to trap PacMan, along with the help of his other Ghost friends.

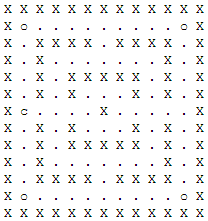
**1.1.2.2 Cam**

Cam is the sad ghost, characterized by its ☹ face. Its sole purpose is to trail PacMan in the hopes that PacMan will one day take a wrong step backwards into the stomach that is has been left empty for so long.

**1.1.2.3 Danny**

Danny is the happy ghost, characterized by its (:D) face. It has no sense of purpose and randomly travels the grid at its leisure. There is no way to predict its next move.

* 1. **PacWorld**

****

PacWorld is composed of Walls and PacDots. In this textual representation, ‘**X’ symbolizes a Wall**, **‘.’ represents a PacDot** and ‘**o’ represents a PowerPellet**. The **c** shown here is the starting position of pacman.

**1.2.1. PacDot**

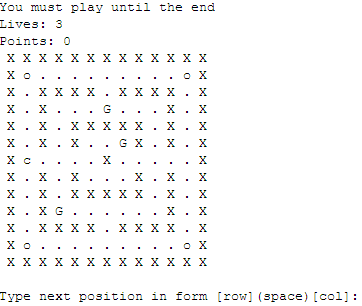
The PacDots feed PacMan points, and if all of them are collected, the game is won. **Each PacDot is worth 10 points**.

**1.2.2. PowerPellet**

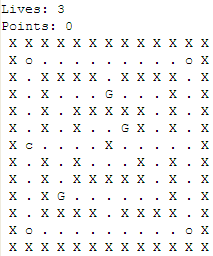
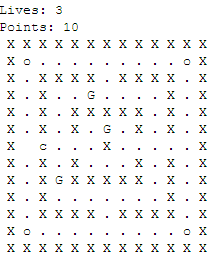
The PowerPellet makes PacMan **invulnerable for the following 10 turns**. While invulnerable, PacMan can eat Ghosts. The Ghosts that are eating will not return for the duration of the game. Ghosts should flee pacman while he is invincible (**not implemented yet**). After 10 turns, PacMan is vulnerable again and must avoid ghosts.

* 1. **PacGame**

The initial screen of the text based game is the following:



This screen states that the game does not end until either PacMan loses all of his lives, or until all the PacDots are collected. The input required is [row](space)[column], which will compute the best movement direction based on the given position. For example, if the first position given is ‘6 6’, the PacMan will move towards the middle of the board.



‘6 6’ ->

If no move can be computed from given position, the game will let you know. Refer to *Section 4.1.* to understand how the engine computes adjacent moves given a position.

**2. MouseGame**

**2.1. Characters**

**2.1.1. MouseHero**

**2.1.2. Mouse**

**2.2. MouseLand**

**2.2.1. MouseTrap**

**2.2.2. Exit**

**2.3. MouseGame**

**3. PipeGame**

**3.1. “Characters”**

**3.1.1. Plumber**

**3.1.2. Water**

**3.2. PipeMap**

**3.2.1. Pipe**

**3.3. PipeGame**

**4. Engine**

The engine describes the generic functionalities that all games may choose to use or overwrite. It contains basic classes and methods that match typical behaviour in a tile based game.

**4.1. Classes**

The classes provided by the engine are Board, Tile, Avatar, Hero, NPC, Item and Wall.

**4.1.1. Board**

The board is where the game is played. There is a method provided that carries out a typical turn, given a Position. This method moves the Avatar based on the given position, than moves the NPC’s based on the Avatar’s position. Each game board is responsible for setting up the layout of the [item] Map prior to initializing a game.

**4.1.2. Tile**

Everything that is placed on the board is a Tile. The Tile is the superclass for any all other classes that are present on the board. The Tile class also represents an empty space on the map.

**4.1.2.1. Avatar**

Avatar is everything that can move on the board. It is the superclass to Hero and NPC and contains methods are used to react to items on the board or how they move.

**4.1.2.1.1. Hero**

Hero is the user controlled Avatar that has a specific goal to accomplish.

**4.1.2.1.2. NPC**

NPC is the AI based Avatar that generally gets in the way of the Hero’s goal

**4.1.2.2. Item**

Items are initially placed on the map, and removed as they are picked up. They generally have consequences such as increasing points, killing Avatars, or other creative purposes.

**4.1.2.3. Wall**

A wall should not be accessed by Avatars unless specifications declare otherwise. They serve to create a maze like feel in a game.

**4.2. Game Play**

All tile based games using this engine use click-move functionality. For the most part, this involves clicking anywhere on the map, resulting in the movement of Avatars based on the position of the clicked tile.

** 4.2.1. Typical Movement**

In the majority of games, the movement of the avatar is one space to the North, South, East or West direction. If the user clicks on a tile that does not lie within this one space limit, the engine will compute what is assumed to be the desired direction. The grid on the right illustrates how the engine determines the direction. The Hero lies in the middle of the grid, and upon clicking inside the + region, it will move either left or right, if there is no Wall where it is to go. The – region will move the Hero either upwards or downwards. This allows the user to spend less time choosing which tile to click.