Tuesday March 1st, 2001

Team project

Owners:

Jonathan Gravel

Boris Ionine

Bruno Colantonio

|  |  |
| --- | --- |
| 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. Game Play | 1. PipeGame    1. “Characters”       1. Plumber       2. Water    2. PipeMap       1. Pipe   3.2.1.1. Q pipe  3.2.1.2. I pipe  3.2.1.3. L pipe  3.2.1.4. T pipe  3.2.1.5. + pipe   * 1. PipeGame   4. Engine  4.1. Classes  4.1.1. Board  4.1.2. Tile  4.1.2.1. Avatar  4.1.2.1.1. Hero  4.1.2.1.2. NPC  4.1.2.2. Item  4.1.2.3. Wall  4.2. Game Play  4.2.1. Typical Movement |

**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 classic. The Hero is PacMan, the Non Player Characters (NPCs) are 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. Three ghosts exist in this version. Their names are Sam, Cam and Danny.

**1.1.2.1 Sam**

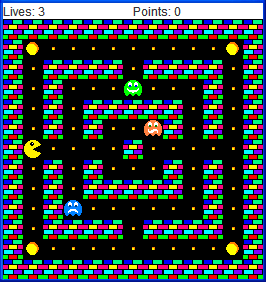
Sam is the sceptical Ghost, characterized by its orange color and (:S) face. It is always contemplating possible moves in order to cut off PacMan. This ghost tries to predict PacMan’s next position and moves towards this position. Sam will move away from PacMan when PacMan is inviulnerable.

**1.1.2.2 Cam**

Cam is the sad ghost, characterized by its blue color and ☹ face. It simply moves towards PacMan’s last position, regardless of of PacMan’s invulnerability. It is also pretty stupid.

**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.

**1.2. PacWorld**

PacWorld is composed of Walls and PacDots. The image to the right shows the starting positions of all the Ghosts and PacMan, as well as the location of the PacDots and PowerPellet.

**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 10 turns**. While invulnerable, PacMan can eat Ghosts. The Ghosts that are eaten will not return for the duration of the level. After 10 turns, PacMan is vulnerable again and must avoid ghosts.

* 1. **Game Play**

The user is allowed to click in the *general* direction he would like to go. For example, if the user wishes to go right, he may click 6 squares to the right and 1 square up. The game will assume this is intended to be a move to the right and will move PacMan 1 square to the right if there are no obstructions. **In order to reduce the number of pop-ups in the game, the game will not alert you when you have made an incorrect move; the move will simply be ignored.**

**2. MouseGame**

MouseGame is a Tile based game and requires you to take your Hero Mouse called Cheesy and get through the maze to a finish point, while all the other evil mice try to catch you. Cheesy has some help, because he can drop a mouse trap whenever he wants to at his current location, provided he has mouse traps remaining in his inventory.

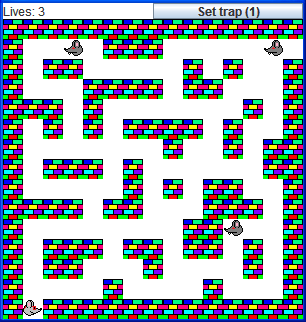
**2.1. Characters**

There are two different characters. The hero is a mouse called Cheesy, the enemies and Non Player Characters (NPCs) are mice called Red-Eyes.

**2.1.1. MouseHero**

Cheesy is the hero of this game and is trying to reach the finish line at the other end of the maze. But it is not simple because Cheesy is being chased by evil mice called Red-Eyes. In his favour he can drop a mouse trap at his position whenever he feels like it is useful; the mouse trap will eliminate the Red-Eyes.

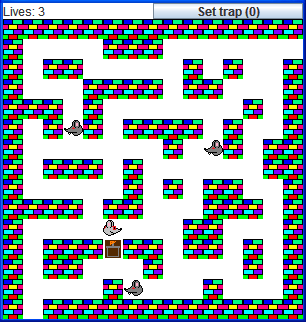
**2.1.2. Mouse**

The Red-Eye’s goal is to try to catch Cheesy, they do so by always going to the last position Cheesy has been. It seems pretty easy, but when you are cornered you have no way out.

**2.2. MouseLand**

MouseLand is composed of Walls and blank Tiles. The image to the right shows the starting positions of all the Mice and MouseHero. The exit is seen at the top right, where the hole in the walls is.

**2.2.1. Exit**

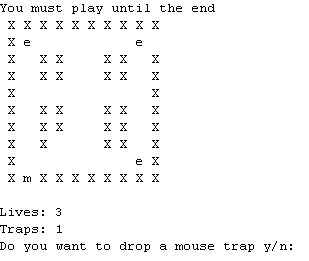
The Exit point as you can see is on the top right of the board where there is a space between the walls. Once you have entered your next position and it is row 1 and column 9 and you are only one spot away you win the game.

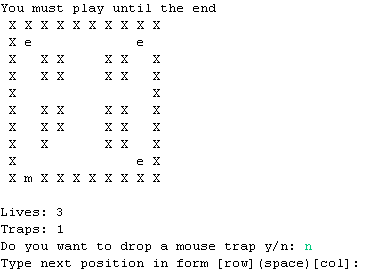
**2.2.2. Mouse Trap**

The mouse trap is shown by the character ‘**T**’ and does not move or do anything until a Red-Eye walks over it. When that happens **BOTH** the Red-Eye and the mouse trap will disappear.

**2.3. Game Play**

Here is the **initial screen of the MouseGame.** You are then asked if you want to drop a mouth trap or not, all you have to do it type in **‘y’** if you want to drop a mouse trap, if you don’t want to you can type in **anything** else.



Then you are asked to type in the **row and column** you want Cheesy to move. The very 1st position has to be of row 1, and example can be“1 1”... (It goes “number (space) number”) and will keep asking for your inputs until you are caught by a Red-Eye (which then the game restarts and you have one less life) or you reach the exit point. If Cheesy is caught, the number of Red-Eyes and mouse traps remain the same number as the previous stage. So if you used your mouse trap and eliminated a Red-Eye then you were caught by another Red-Eye. Everyone will go back to their original positions without a mouse trap and without the Red-Eye that was eliminated.

You do not have to type in the exact Tile that cheesy will be going on next, as long there is a clear path to that tile without having a wall in the way he will go. **(See 4.2 for more explanation).**

**3. PipeGame**

The game pipes is a tile based game, and the point of the game is to arrange a series of pipes such that the water does not overflow before reaching the end pipe. The type of pipe is chosen randomly for a new and exciting game every time!

**3.1. “Characters”**

There are two characters. The hero is the Plumber, the enemies and Non Player Characters (NPCs) is the water.

**3.1.1. Plumber**

Every turn the plumber gets a random type of pipe and he has to choose what tile to go on next for him “drop” the next pipe. If the plumber does not connect all the pipes together the water will overflow, causing the game to be lost.

**3.1.2. Water**

The water is the enemy in the game, it continues and splits to follow all open sides of the pipe (ex: the “+” pipe allows water to go up, down, left, right). Your first five turns the water does not move, then every action the water will leak to the tile(s) that the water can go.

**3.2. PipeMap**

PipeMap is initially composed of the initial start of where the water is and a plumber. In this textual representation, ‘**P’ symbolizes the plumber** and **the blank tiles are empty space** where you can place the next pipe.The ‘**Q’** shown here is -the starting position of the Water. The water is shown by a **‘w’**. The image shown here is the starting position of the PipeGame.

**3.2.1. Pipe**

Pipes are given to you randomly; right now you cannot see what the next pipe is going to be, but for the next milestone, it will be possible to see the next pipe coming (think of Tetris as an example). There are five different pipes.

**3.2.1.1. Q pipe (incoming = 1, outgoing = 0)**

The “Q” pipe can take water in from one side but then has a dead end.

**3.2.1.2. I pipe (incoming = 1, outgoing = 1)**

The “I” pipe can get incoming water in from one side and the water will come out on the opposite side it came from.

**3.2.1.3. L pipe (incoming = 1, outgoing = 1)**

The “L” pipe can get incoming water from one side and the water will leak out on either side but NOT both, depending where the water came in.

**3.2.1.4. T pipe (incoming = 1, outgoing = 2)**

The “T” pipe can get water coming in from one side and have two exit points, the exit points are on both sides of where the water came in.

**3.2.1.5. + pipe (incoming = 1, outgoing = 3):** -The “+” pipe can get water coming in from one side and the other three sides the water can exit.

**3.3. PipeGame**

Currently, only the “I” pipe is used. Since we don’t have any images, it is very difficult to show what the different pipes are. When you rotate the pipes you do not know how the pipe is rotated (what sides are open at start), for example the “I” pipe can let the water flow from up to down or left to right.

Initially the “I” pipe is set horizontally, but the plumber can rotate it after he has set it. To do so, the plumber just has to go on the tile the pipe is on and it will rotate by itself. The problem here is that we cannot rotate the character “I” so it seems that the pipe was not rotated.

In this game you control the plumber (“p”) by declaring the position you want to move to. “p” will have changed places after setting the position you desire, and you will notice that the tile “p” used to be on now holds an “I”. Keep in mind that “I” pipe is going from left to right so if you want it to be vertical, you have to move back to that tile to rotate it. You have 5 turns to play before the water starts to move, you will see pipes fill up as the water moves (“w” beside the pipe indicates water is inside).

To win the game you want the water to reach the open wall (tile 6, 1) and the water starts at tile (2, 1), So all you have to do is a straight line down, keep in mind you have to rotated every “I” pipe.

To test this, insurt:

(3,1)(4,1)(3,1)(5,1)(4,1)(6,1)(5,1)(6,1) then place the plumber at any blank area.

**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.