Pacman Guide 106

Thomas Kuhlemeier, Sam Leurink

Index:

1. Requirements

1.1 Player

In our game the player can take control of pacman, which is the Big Circle, and the goal is to eat as much food, which are the smaller circles, as you can. Pacman can be controlled by the arrow keys.

1.2 Enemies

The enemies in this game are four “ghosts” which are represented by the red, magenta, cyan and orange squares. These enemies wander around randomly until they see you, at which point they will come directly at you. When they don’t see you anymore they will resort to wandering around randomly.

1.3 Randomness

As mentioned in enemies, the AI wander around randomly when without any vision. This is done by giving the ghosts possible directions which consist of forward,left and right (only if these are possible of course) and picks one of them randomly.

1.4 Animation

The animation in our program is in the form of pacman eating. His mouth opens and closes every 3 frame steps. This is captured by the bite function and the Jaw constructor of the Pacman data type. Furthermore, pacman moves around the level in a fluently animated manner. Just as the ghosts do. Pacman waits until he can take the desired turn by the player. Pacman also stops automatically when he bumps into a wall. There is still one little glitch: in some areas pacman cannot move out of a wall which he bumped into by strafing, only by going back the way he came from and then keep going that way or come back and take the turn in time to prevent bumping into the wall again. So remember when you’re stuck, back up and you will be fine. It is probably because of an inaccuracy of the Floats we are working with.

1.5 Pause

The pause button ‘p’ pauses the entire game. At the start of the game the player is asked to press ‘s’ to start the game. Furthermore we have running, victory, gameover, and dead runstates. However, for these last three states we unfortunately didn’t have enough time to implement them.

1.6 IO

Our IO function takes and writes the highscores to a file. Also, the IO part is separated from the rest of the pure code. The IO parts can be found at the bottom of each module they occur in.

2.

2.1 AI

In the AI file you find the framework for the ghost AI’s while also defining some functions for the controller document. For each ghost you can see a specific function that decides their AI, but all of them are the same in AI. They use the function vision to decide if the Ghost can

see Pacman and if that is the case the ghost will head to Pacman with the function topac. Later in the file the function cellVal is defined which returns the cell either pacman or a ghost is vacant in. Midpt is also defined to find out if the Ghost or Pacman is standing in the middle of the cell.

At the end of the file you can see a bfs function which would take the shortest path to 0,0 which was intended for the ghosts if they happened to get in an eaten state, sadly we couldn't get to using it due to time issues.

2.2 Controller

Controller defines all the moves taken during a step and also the moves taken on an input. In the movePac function it is taken into account that you can't just at any point change direction and move into a wall which is why a function is added that in the case a wall is in the direction you want to go the key is remembered and used at the time it is possible again.

2.3 Main

The main file is the file in which you can find how many steps per second, the window size and the functions called every frame.

2.4 MazeMap

There are two mazemap files: A text file, which consists of the playing field in a string form, and a hs file which puts each line as a string into a list.

2.5 Model

The model contains all the datatypes we use across all files. It also defines the starting state.

2.6 View

The view file turns all the elements of the gstate into visual representations.