Pacman Game using Qemu Emulator – Gearoid Sheehan (R00151523)

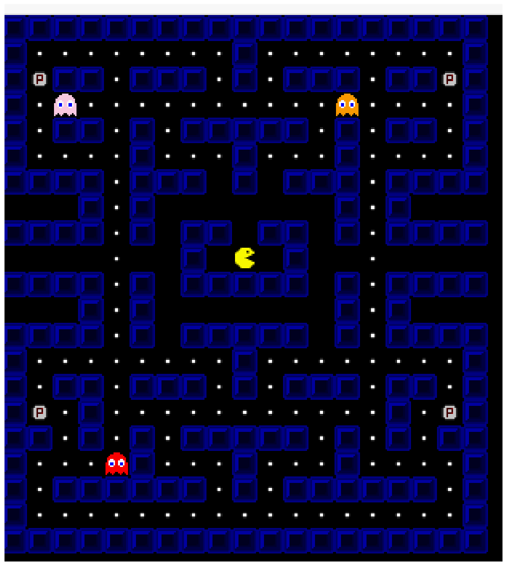
Briefly explain how the firmware in the pacman-qemu.zip works

The firmware required to run the Pacman game is made up of several different files. All images used are of type BMP. We use BMP images as we can retrieve individual pixels from them by using a set of coordinates that allows the data to be conceptualized as a grid. The uart.c file contains all the driver code for the UART functionality in the system, which allows for the key entries for Pacman's movement to enter the UART peripheral in QEMU and be used in the program. In the t.c file which contains the main driver code, an infinite while loop runs the background code, moving the sprites around the map. Inside this while loop exists an if statement with the function upeek() as the condition. This checks for user input without blocking the other code for the sprites movement from running in the background. If a key is entered, the ugetc() function retrieves the value. A graph called table is created, which is a 2d array. This graph is populated with a variance of 'M', 'P', 'W' and 'V' values. These values correspond to elements in the final Pacman game environment, which are walls, points, power-ups and empty spaces respectively. The BMP files are used then to provide the visual aspect for these values, as well as BMP files for the Pacman itself and the Sprites which chase it. All functionality for printing the BMPs to the QEMU emulator is in the vid.c file. When a key is entered to change the position of the Pacman, an if statement checks if the desired direction will push the Pacman outside of the game's outer walls. If the move keeps the Pacman in the game, the Pacman attempts to move a further 16 bits in that direction. Another if statement uses the function check_boundary() as a condition, which takes the x and y values and compares with the graph if making this move will place the Pacman over one of the inner walls, which is an invalid move. If this is the case, the move is reverted. If the Pacman successfully moves into the desired new position, the value residing in that new position is checked in the graph. If a 'P' value is moved into, the score is incremented by one. If a 'W' value is moved into, the Pacman receives a power-up, changes colour and its score is multiplied by three. If a 'V' value is moved into, nothing happens. After each of these values are moved over, they are replaced by 'V' value, signifying that the payload at that value has been used up. This is done by calling the black_point() function in vid.c. Another function in vid.c, called putback(), removes the duplication trail from the Pacman and the Sprites as they traverse the board. The movement of the Sprites are controlled by Dijkstra's algorithm. The implementation of this algorithm is in dijkstra.c. Edges and vectors are found in the graph for possible paths. For each Sprite, the algorithm is called and maps out their path towards the Pacman. The path is returned as an array of absolute vertex's, which are calculated back into x and y values for each Sprite to step through. If one of the Sprites catches the Pacman and enters the same position on the graph the game is finished, and a display message is shown on the terminal.

Extra feature

When the Pacman BMP enters the graph index where a power-up P value is present and 'eats' the power-up, the Pacman BMP changes colour from yellow to white and the score is tripled.

Pacman and the Sprites at the beginning of the game:



Pacman and the sprites after eating a power up: