PacMan programmer manual

The game contains 3 C and 2 header files.

## Main.c:

int main(int argc, char \*argv[])

{

Inicialization(window,sound,timer, music, icon) Giving values to surfaces and checking if they could open. Place of GameState enum, which will tell us if we’re going to enter the menu, exit, or join the game (depending on the returning value). Also contains the liberations.

}

void nyertel(void) //win

{

If the user has won the game, this function get called. Zooming the „youwin” picture (origo centrally) in 50 steps. 20ms delay between each step. Also dimming the background.

}

void vesztettel(void) //lose

{

Same as Win, but get called if we lose.

}

int highscore(void)

{

After the win/lose function, the user can enter his/her name. The game accepts characters between a-z, A-Z, and 0-9. It uses dinamic memory allocation. At the event of pushing a button, the previous and the current characters are printed out. If the user hit a backspace the current name is being deleted.

}

void highscorefajlba(char\*nev) //Highscore\_to\_file

{

Loading the current highscore file into a linked list. Inserting the user current result into it’s space (depending on the point) and writes the new file.

}//This function is currently buggy.

## Menu.c:

int menu\_kezdo(void) //menu\_starter

{

Inicializing the menu, printing the menu texts. Waiting for events. Sorting the events depending on its type. If the user click, calls a subroutine. Also this function reads the map if the user choose new game or load game.

}

void menu\_hatter\_kirajzol(void) //menu\_background\_printer

{

Blackens the screen, then prints the background, the title, and the icon of the sound.

}

int menu\_leiras(void) //menu\_manual

{

Prints the block that contains the manual of the game, and draws the pictures of the score system. Just like in every submenu, there is an event waiter block. The aim of this block is to make the user be able to mute the song.

}

int menu\_eredmenyek(void) //menu\_highscore

{

Reading the highscore file, and prints that. Also event waiting. If the highscore file is empty it says: no data found.

}

## Jatek.c:

int jatek\_indit(void) //game\_starter

{

This function searches the position of Blinky and PacMan in the „palya” (means map) block, and computes their coordinates. This function handles the timers and the events. If there is a PeekEvent, the function will compute the next direction of Blinky, and gerenates extra scores (furthermore in the specification). In case of UserEvent, it deletes the position of pacman, an computes the new one examining if PacMan would bump into a wall or the end of the map.  
Checks if we win or lose. Handling the cheats and the controls.

}

int jatek\_menu(void) //game\_menu

{

Gets visible when the ESC button is pressed in the game. Contains 4 texts, the usual mute and exit events. When the mouse is clicked it decide where the user clicked depending on the coords of the mouse (just like all the other menus)

If the user Quits it calls the quit function which checks if the user want to save.

}

void kirajzol(void) //printer

{

Walks trough the map, and draws every picture to it’s place. ( #wall, 1PacDot, 2PowerDot, 3Cherry,4Orange, 5Melon, 6Bell, 7Key) The map can be found in the terkep.txt

}

int kilepes(void) //quit

{

Asks if the user really want to quit without saving.

}

int almenu\_start(void) //Submenu\_start

{

Modified GameState enum. The stacking of the submenu had been eliminated with this.

}

void palyaszele(void) //edge\_of\_map

{

Examines if Pacman or Blinky reached the edge of the map. If he reached, and the direction points forward, it places the character to the opposit site. (The edge of the map can be reached, if there is no wall, so in the original map it’s the 25th line)

}

void falvizsgalo(void) //wall\_examiner

{

If the next\_direction enum points toward a free place, changes the current\_direction.  
If the current\_direction points toward a wall (and PacMan reached the next integer place in the block) this function stops PacMan.

}

void pontrendszer(int y,int x) //score\_system

{

Checks how much score have to be added to the score based on the current position. Also printing the score to the screen.

}

void kovkoordinataszamolo(void) //next\_coord\_computer

{

Computes the next coordinates (current+speed).

}

void kiment(void) //save

{

Saves the map and the points into the save.sav file. In the file the content is stored as normal text.

}

void extrapontok(void) //extra\_scores

{

Generates the orange, melon, cherry, (etc.) and places it to an empty available place in the map. If the function still can’t find an empty place after 10 attempt, it gets discharged due to the continuity of the game.

}

void blinkykoord(void) //blinky\_coords

{

Same as wall examiner except it can be called to Binky’s vectors.

}

Notice: I’ve learned C and SDL during this game, and when I wrote the ghost’s functions, I had no idea about the generic functions. Instead of copy all the functions I’ve only implemented one ghost. I wouldn’t have the time, to implement the AI of Blinky, or to rewrite the code with generic functions, so instead of this Blinky moves faster but randomly, and when he would go toward a wall, his eyes turn into creepy mode.