**Pacman**

1. Define global constants:
   1. height and width of the play area - H = 30 and W = 60 to use field in this doc
   2. the number of ghosts
2. Create a struct that can contain coordinates (x, y)
3. Create a PacMan struct with the following fields:
   1. Coordinates struct
   2. 2 integers (nx, ny) to define the direction the player is going
   3. An integer for the lives’ number
   4. An integer for the food collected
4. Create a Ghost struct for each of the ghosts with the following fields:
   1. Coordinates struct
   2. 2 integers to define the direction the ghost is going
5. Initiate an empty ghost struct for all of the ghosts
   1. struct Ghost allGhosts[NR\_GHOSTS];
6. Initiate the PacMan struct:
   1. Coordinates struct and lives number should be defined
   2. The other fields are 0
7. Initiate the playing area as a 2-d char array:
   1. Char area [][] = { {####}, {# #}, …., {####}};
   2. Obstacles and walls defining the outer are of the field should be defined with (#)

{

{ "############################################################" },

{ "# #" },

{ "# #" },

{ "# # # # # # # # # # # # # #" },

{ "# # # # # # #" },

{ "# # # # # # #" },

{ "# # # # # # # # # #" },

{ "# # # # # # # # #" },

{ "# # # # # #" },

{ "# # # # # #" },

{ "# # # # # #" },

{ "# # # # # # # # # # #" },

{ "# #" },

{ "# #" },

{ "# #" },

{ "# ## ## # ## # #" },

{ "# # # # # # # # # # #" },

{ "# # # # # # # # # # #" },

{ "# # # # # # # # # # #" },

{ "# # # # # # # # # # # # #" },

{ "# # # # # # # # #" },

{ "# # # # # # # # #" },

{ "# # # # # # # # #" },

{ "# # # # # # ## #" },

{ "# #" },

{ "# #" },

{ "# #" },

{ "# #" },

{ "# #" },

{ "############################################################" }

};

1. Create a void function to initialize:
   1. All ghosts of the array:
      1. The coordinate as a random place on the playfield where there is a space ( ) except the (1, 1) coordinate since this is where your PacMan will spawn
      2. At this coordinate the space will be replaced by a G to represent the ghost
   2. the play area. Put a dot (.) as food for the PacMan. The easiest is to replace all the spaces ( ) with this PacMan food. Do not of course replace the Ghosts Gs with food
2. Create a void function that gets the keyboard presses of the user:
   1. Do nothing if it is not an arrow
   2. If there is an up arrow your ny is 1, down it is -1 and nx = 0
   3. If there is a left/right arrow your ny is 0 and your nx is either -1/+1
   4. After each press move each ghost into a new random position, be careful that if there was food in his position to put it back again
3. Create a void function that moves the PacMan:
   1. delete PacMan (P) from old position
   2. compute new desired coordinate (nx,ny)
   3. test whether there is a wall at (nx,ny)
      1. If there is collision then remove 1 life then place PacMan in 1,1 position
      2. If not update PacMan coordinates; If there is food then add food collected count by 1; put PacMan back again to playfield
4. Create a void function that checks the lives left of PacMan:
   1. if it is negative then you print out user score as the number of food collected
   2. If it isn’t then it prints the playing field
5. Use the following functions for your cursor:

void set\_cursor\_position(int x, int y)

{

//Initialize the coordinates

COORD coord = { x, y };

//Set the position

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

void hidecursor()

{

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 100;

info.bVisible = FALSE;

SetConsoleCursorInfo(consoleHandle, &info);

}

* 1. write your main function as such:

int main()

{

system("cls");

hidecursor();

function\_in\_bulletpoint\_8();

while (1)

{

function\_in\_bulletpoint\_9();

function\_in\_bulletpoint\_10()();

function\_in\_bulletpoint\_11();

Sleep( 1000 / 30 );

set\_cursor\_position(0,0);

}

}