Software Engineering

Mohamad Sharkawi

I made my game using the **WPF application** template in **Visual Studio**.

I will be using the standard C# code convention.

How my Game works:

My game is played on a 15 x 15 grid, the grid is represented by a two-dimensional array.

Empty positions in this grid are represented by the **value 0, value 1** for positions occupied by the **snake,** and **value 2** for positions occupied by **food**.

How the snake is spawned:

First I made 2 properties of type int in the **GameState** class, called **Row{}** and **Cols{}**, in the same class I also added the **Grid{}** property to access the information from the **GridValue** class.

After that I made a constructer in the **GameState** class, this constructer takes the number of **rows and cols** in the grid as parameters, and then initialized my 2d array to the correct size using these parameters.

And now every position in the array will have an **empty grid value** because it is the first **enum value.**

And now for actually spawning the snake, I wanted the snake to appear in the **middle row on column 1 2 3** , so I made a method called **AddSnake(),** then I created a variable for the middle row, then I made a **for loop** that has an int that starts at 1 and stops after it reaches the number 3, then inside the loop I set **Grid value** with the **row** and **column** values to **the Grid value of the snake,** then I updated the actual position of the snake using these values.

How the food is spawned:

I made **2 for loops** to check all the **row and the columns** in the grid, then I added an **if statement** checking these positions if the are **empty**, if they are empty the I return these positions.

After that I added the **AddFood() method,** then I made a list of all the empty positions, then I choose a random **empty position** from the list to be occupied by the **Gridvalue.food.**

How the snake moves:

So instead of moving **the whole body** of the snake to a certain direction, what I did is just adding a **new head** for the snake to that direction using the **AddHead() method** and at the same time **deleting the tail** of the snake using the **RemoveTail() method**.

A bug that I encountered:

So for example if the snake was moving to a certain direction and the player decided to click the **opposite direction** the snake would **hit itself** and the **game is over**. So what I mean by that, let's say the snake was moving to the **left direction** and the player decided to input the **right direction** (which is the opposite direction) the **game is over.**

What I did to fix this is by making the game **ignore** the opposite direction inputs.

But It didn't end here, lets say the player decided to make **two inputs really fast which** will result in the snake hitting it self before it can actually change direction.

To fix this I just added a slight **input delay.**