**Building a Probabilistic Agent : Bust the Ghost Game Using Unity**

The submitted code contains three scripts:

* Tile.cs: This class represents tile objects which refer to the cells on the grid. The main behavior of the Tile class is the OnMouseDown() behavior which listens to the mouse click on a tile. If the click is the first click, it is handled by calling the UpdatePosteriorProbabilities() which resides in the probabilities script (more details on this later). On the other hand, if the click is the second click, it is interpreted as an attempt to bust the clicked tile. This click is handled by calling the Bust() function which checks if the busted cell is the same as the ghost’s position in which case the player wins, otherwise the player loses.   
  This script contains two other important functions SetProba() and SetColor(). SetProba() receives a probability to be displayed on the tile, converts it to a percentage, casts it to a string, then writes the probability on the tile. SetColor() receives a color and displays it on the tile.
* GameGrid.cs: This is the class that starts the game. It creates a grid based on the width and height set in Unity (8 by 20), and it instantiates the tiles. It also invokes the InitGamePlay() in the Probabilities script which places the ghost, initializes some global attributes, and calls the initializeProbabilities() function.
* Probabilities.cs: This is the script where the probabilities are being set and updated as the game proceeds. First, when the initializeProbabilities() function is invoked, it iterates over all the tiles and sets their initial probability(i.e., 1/(20\*8)) by calling the tile’s SetProba() function.  
  Next, when the UpdatePosteriorProbabilities() function is called (upon first mouse clicks), it gets the distance from the ghost through the GetDistanceFromGhost() function and passes it to GetDisplayedColor() which returns the color to be displayed on the clicked tile based on the distance and taking into consideration the sensors noise. Then, in order to know which probability is to be displayed on the tile, the ConditionalProbabilityDistribution() function is called, this latter finds the probability of the color given the distance and returns that probability. The returned probability is multiplied by the previous probability of the clicked tile, and then the probabilities on the tiles are normalized using the Normalize() function. The Normalize() function sums up the probabilities on the tiles, then it iterates over the 2D array of probabilities corresponding to the tiles and divides each probability by the sum.