Group Members:
Leila Farah Moussa < 76287 >
Khaoula Ait Soussi < 79155 >

# 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 that the ghost is on the clicked tile and returns that probability.

The returned probability is multiplied by the probability of the ghost because following Bayes' rule we know that the probability of the ghost given the color is proportional to the probability of the ghost multiplied by the probability of the color given the ghost.

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.

**Limitations of our solution:**

- Probabilities are inconsistent with the colors: although this might be expected given the noise of the sensors, but we still considered it a major problem. Sometimes, the probability displayed on a red tile is less than that displayed on a green or yellow tile which should not be the case.
- Unclicked cells are given a high probability after normalization: When a tile is clicked and ends up given a probability less than the uniform one. The unclicked tiles get a high probability after normalization. We thought that the unclicked tiles should not be given probability higher than the orange and red clicked tiles.
- The probabilities on unclicked cells are the same: The suggested formula and the normalization do not suggest any priority to be given to unclicked tiles based on their location with respect to clicked ones. For example, we may be tempted to say that the neighboring tiles to a yellow or green tile should have very low probability compared to those neighboring an orange or red tile; but this cannot be achieved given the formula only, some pre-processing of the tiles' probabilities should be done right after clicking a tile and right before normalizing.

**Future Enhancement:**

A good future enhancement to the way the probabilities are calculated is to take act upon the probabilities of the neighboring cells based on the color and probability of the clicked tile. For instance, if the clicked cell is green, all the cells located at a Manhattan distance less than 5 from the clicked cell should have a very low probability and the further apart cells (5+) should have a high probability, and then normalization should occur.