Limited Space: A Prototype

Kellen Strydom  
2559739

Investigation Aim

To explore placement, movement, and pathfinding on a grid. This exploration will be done through a mouse finding food in a maze, where the maze can be updated, and the mouse’s movement and decisions adapt to the environment. This investigation is done with the intention of using the mechanics in TD games, city builder, and similar games.

Progress

Reflection

Placement:

I learnt that a grid is just the size of a cell that everything snaps to. I made a function that take the position of any object and returns the position of the cell that they should snap to. This function was then used to keep the all object on the grid.

I made the placement of the objects modular. This means that I can use the same code to make any object a placeable object. Having this modular design made it possible to quick add the ability for the player to place more mice and cheese in the maze. The placement code not being specific to this one project and allows it to be used in other projects that need placement (2D or 3D).

I found that players waned to place some objects, such as walls, multiply times in a row. The player would find it frustrating to constantly keep having to press the button to select a wall. I added functionality that allow the player to keep placing the same wall piece until the action was cancelled. I found that this greatly improved the user experience. From this I have learnt to think about actions that want to be repeated and make it convenient to do so.

Pathfinding:

The method was used for the mouse movement kept changing as I introduced more complexity. I would not think ahead to what I plan to add in the future and make a pathfinding algorithm that only solved that current state of the project. This was wildly inefficient as I would have to throw out all the progress, I had made a new variable was introduced. From this I learnt to plan out the functionality requirements before starting on a solution so that the whole thing does not need to be scrapped down the line.

The algorithm works by weighting the cell depending on how far it is away from food. The lower the number the closer the cell was to food, and wall would have an extremely large number so the mouse would not go to them. The mouse would check all adjacent cells and move to the cell with the smallest value.

The algorithm would update all cell weights every time the maze was updated. At first the update was recursive, starting for each food going outwards. Due to this, as the number of cells increased the number of computations increase exponentially. This meant that the algorithm worked for a small number of cells but was very slow for a large number of cells. To fix this, I remove the recursive nature of the method and only allowed the cell to be updated once. I did this by putting the cells that were needing to be updated in a queue. This made the it more efficient as the computations would scale linearly with the number of cells.

Lessons Learnt

* Make functionality modular – for both future addition in the project and to carry the code to other projects.
* Think about and identify actions that the player will want to repeat – make it convenient for the user to do so.
* Think of all requirements the solution needs to solve, so that work you do at the beginning does not have to be redone.
* Be aware of the complexity of solution and the inefficiency as the number increases.