USER MANUAL

GAME

ComPuzzle is a game where you try to get your character from a starting location, to a finishing location. You interact with the game by using function blocks to control your character’s movements. You will string multiple commands together to create a Program that your character will follow.

The Game:

The game will start by bringing you into map 1, where it will teach you about the three core components of the game, but more on that later. When you load into the first map, you will see 3 main sections, the Map, the Function Selector, and the Command List.

Map:

The map is where you will plan out your path to the goal. You will notice that the maps all have tiles placed everywhere, those tiles are coordinates on the map, but you don’t need to know the locations, they are there to help visualize the path and number of commands you will need to get from point A to point B. Each map will have a defined Starting location (the tile you spawn on) and a Goal tile (the tile you want to reach). There may be obstacles throughout the map that forces you to take an alternate path to the goal, to spice the game up.

Function Selector:

The Function Selector is located on the bottom of the screen, and it holds the functions that you will use to create your Program. The functionality of these functions will be explained later on in this document. In the Function Selector you will see 8 buttons with the names of the function. When you click on a function, a copy of that function will be added to the Command List. The Function Selector also has the following buttons, ‘Menu’, ‘Run’, ‘Reset’, ‘Clear’, and ‘Create’. The Menu Button will bring you back to the Main Menu. The Run Button will start reading the Command List and will disable all of the Function Selector Buttons, and any functions in the Command List, with the exception of Main Menu and Reset. The Reset button will end the Program early and reset the Character to its starting location. The Clear Button will Remove all Functions from the Command List, leaving it empty. The Create Button will bring up a menu where you will have the ability to create your own function by using the base 8 that are provided to you. More on this later.

Command List:

The Command List is where you build your Program located on the right side of the screen. When you click a Function is the Function Selector, it will be created in the Command List. Each Function will have 4 things unique to the Command List, an Up button, a Down Button, a Delete button, and an Index number. The Up and Down button will move the Function up or down the Command List. The delete button will delete the Function. The Index number shows you where in the program it is, and its order in the list. The Command List will run the Program in a top down order, it starts at Function 1 and goes until the end (function 2 – infinity).

Functions:

The game has 8 built in functions, they are, ‘Take Step’, ‘Turn Left’, ‘Turn Right’, ‘Loop’, ‘End Loop’, ‘If’, ‘Else’, ‘End If’. While it may seem like it is complicated, these functions are relatively simple.

Take Step:

The ‘Take Step’ function is one of the simplest, but the most critical. This function will make the character take one single step in whichever direction it is facing.

Turn Left:

The ‘Turn Left’ function is one of the other core functions, it will make the character turn to **ITS** left, or counter-clockwise, whichever helps you visualize it better.

Turn Right:

The ‘Turn Right’ function is one of the other core functions, it will make the character turn to **ITS** right, or clockwise, whichever helps you visualize it better.

Loop:

The ‘Loop’ function is where things can start getting confusing. Essentially ‘Loop’ will repeat ALL functions within the Loop block and its ‘End Loop’ block. There will be a small counter (with an up and down arrow) in the Top Left of the Loop Function when in the Command List. That number controls how many times you will repeat the Functions inside the Loop before it exits and continues on to the rest of the Program. If there are multiple loops in your Program, the ‘End Loop’ block that your ‘Loop’ belongs to may get confusing. Each loop will have exactly ONE End Loop assigned to it. If you put a loop inside another loop and BEFORE the first’s End Loop, that is called Nesting. When you Nest a loop, you need to end the inner loop before you can end the outer loop. Think of it as a Russian Nesting Doll, you can put one inside another, but the ones inside need the Top and Bottom parts to be put together correctly. The same thing applies to loops, they need to be closed (inside loop) before you close the bigger one (outside loop).

End Loop:

The ‘End Loop’ function signals that it is the end of the section of Functions that you want to repeat. There is only ONE ‘End Loop’ per ‘Loop’. Each ‘End Loop’ is assigned to its closest ‘Loop’ and cannot be shared. If a Loop already has an ‘End Loop’, then it will attach to the next closest.

If:

The ‘If’ Function will run all functions between the ‘If’ and ‘End If’ OR ‘Else’ ONLY when the condition assigned to it is True. When the condition is True, The Functions underneath will be run until it the Program encounters either ‘End If’ or ‘Else’. Both of those will be explained after this. Once the Program finds one of those two, it will ignore all other Functions until the ‘End If’ where the Program will continue as normal. If the original condition happened to be false, then the Program will instead ignore all of the functions between the ‘If’ and either ‘End If’ or ‘Else’ where the Program will once again continue normally.

Else:

The ‘Else’ function is the only extra function here, and is used alongside the ‘If’ function. It must be placed within the ‘If’ and the ‘End If’ functions. As previously discussed, ‘If’ will test a condition and give a true or false, If the condition was false, it will ignore all of the functions until the ‘End If’ functions was found. There is a caveat to that though, and that is the ‘Else’ function. ‘Else’ is Similar to ‘If’ but will only do its Functions when the ‘If’ gives a false value to its condition. If the optional ‘Else’ is used and the condition was false, then all Functions placed between ‘Else’ and ‘End If’ will be run. If the original condition was true, and the ‘Else’ function was utilized, then the program will instead ignore all Functions between the ‘Else’ and ‘End If’.

End If:

Similarly to the ‘End Loop’, the ‘End If’ signals the end of the ‘IF’ Function and acts the same way. There can only be one ‘End If’ per ‘If’, and the ‘End If’s attach to the closest ‘If’ so long as that ‘If’ does not already have an ‘End If” assigned.