1 Game Manual - Welcome to Cells!

Welcome to Cells, and thank you for participating in this test. The game is in early development for a university project, which is why your feedback is extremely important. The following manual will describe the game's features and hopefully give you a better understanding of what it is you can achieve in Cells.

1.1 Your Objective

Your goal is to beat the other cell on the board. Your cell is green, and the enemy cell is red. It is a game of algorithmic thinking, so you will have to write some sort of algorithm, that solves your task. Consider an algorithm to be a set of instructions - such as a recipe, when you are cooking a meal. It will likely take more than one play through to win the game!

1.2 Strengthen Your Cell

By consuming stars, your energy level will rise, and when it reaches a point greater than your opponents, you are able to eat them up! But be careful, you spend energy on every action you make across the board, so it is important to stock up on energy (stars)!

1.3 To start press singleplayer..

You will find that some of the features are greyed out, this indicates what you cannot click on. To start up a new game simply chose **singleplayer** $->$ **skirmish** $->$ **new**.

1.4 Welcome to the Editor

The editor is where you create your cell's algorithm. Every hexagon dropped into the empty black fields, will be executed by the cell. When your cell has executed every action brought into the editor, it will start all over again, so your cell will never cease to do actions, unless directly specified not to.

1.5 General Editor Notes

The game asks you to specify when you see this:

Figure 1: Set Board Direction

Which direction on the board you wish to interact with. For example, pointing right as seen in the picture, would allow you to move right, with the move construct. If you use the 'look' construct, this would make your cell look one tile in the given direction. You will also see this:

Figure 2: Set Program Editor Execution Path

Which describes what direction you wish to continue on the board. If you point right, like in the picture, the program editor will continue right, and thus execute what is next to it. When you use an if construct, you will see that there are two of these, one is blue, another is red:

Figure 3: Set Program Editor Execution Path

Blue represents the **'true'** case and red represents the **'false'** case. If the case evaluates to true, the program will execute along the blue path, if it evaluates to false, it will continue along the red path in the program. This is not to be confused with actual movements on the board.

1.5.1 Move

The move action is the simplest action in the game. Simply specify which direction on the board you wish your cell to move either by clicking the

aforementioned wheel (figure: 1), or choosing a direction variable from the choose direction drop down box on the form.

1.5.2 Split

Split action functions exactly like move. The cell will split in the direction specified in figure: 1

1.5.3 Look

This will allow you to see what is in your near vicinity (1 tile). You can either use the direction you look in from a saved variable, or you can chose a direction. The result of look will be saved in two variables, a number variable defining the energy of what you saw in, and the entity, in a different variable.

1.5.4 Loop

Loop allows you to iterate over three different types, namely; numbers, direction and entity types. If you chose to iterate over entity types the loop will begin by iterating over **friends**. When the loop is done with this - and there is a path back to the loop - it will begin iterating over **enemies**, and later **food**, **empty tiles** and **out of bounds** - this is assuming that there is a path leading back to the loop.

If you chose to iterate over directions, it is possible for you to save a direction to a variable and use this later in combination with a 'look' construct, to determine what is on a given direction position. On its own it is not very useful! The final loop type is the numbers loop, which allows you to set up a bit of a counter, it will count an interval, and save this interval to a variable, allowing you to run a code segment a given number of times, before something new is done.

1.5.5 If

The if construct is fundamental to all programming languages. Like with the loop, you are able to make an if-check on **numbers**, **directions** or **entity types**. You may specify the input to read from a variable, or get new input on the fly, and make a comparison. If true, the construct will follow the **blue path**, if false, the construct will follow the **red path**.

1.6 When you are done

Press OK in the bottom right, this will start the game with the instructions you gave it in the editor. Remember that you can always return to the editor.

May the best cell win, and good luck!