The BeeBot AI Environment

You have been tasked with developing a search algorithm for automatically controlling BeeBot, a Bee which operates in a hexagonal environment, and has the capability to push, pull and rotate honey 'Widgets' in order to reposition them to target honeycomb locations. To aid you in this task, we have provided support code for the BeeBot environment which you will interface with to develop your solution at `https://github.com/comp3702/2024-Assignment-1-Support-Code`. To optimally solve a level, your AI agent must efficiently find a sequence of actions so that every Target honeycomb cell is occupied by part of a Widget, while incurring the minimum possible action cost.

Levels in BeeBot are composed of a Hexagonal grid of cells, where each cell contains a character representing the cell type. An example game level is shown in Figure 1.

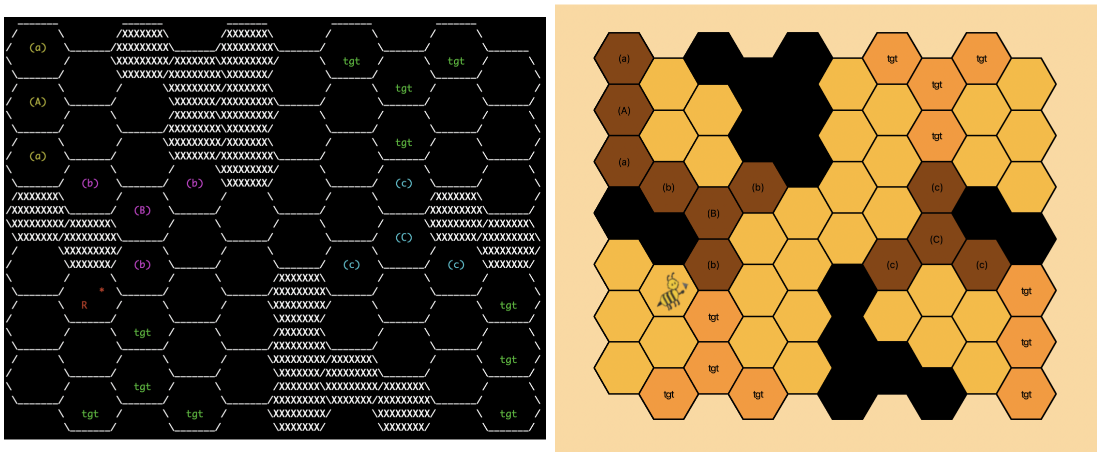
Figure 1: Example game level of BeeBot, showing character-based and GUI visualiser representations

Environment representation

Hexagonal Grid

The environment is represented by a hexagonal grid. Each cell of the hex grid is indexed by (row, column) coordinates. The hex grid is indexed top to bottom, left to right. That is, the top left corner has coordinates $(0, 0)$ and the bottom right corner has coordinates $(n_{rows} - 1, n_{cols} - 1)$. Even numbered columns (starting from zero) are in the top half of the row, and odd numbered columns are in the bottom half of the row. An example is shown in Figure 2.

```
   ____        ____
  /    \      /    \
 /row 0 \____/row 0 \____  ...
 \col 0 /    \col 2 /    \
  \____/row 0 \____/row 0 \   ...
  /    \col 1 /    \col 3 /
 /row 1 \____/row 1 \____/  ...
 \col 0 /    \col 2 /    \
  \____/row 1 \____/row 1 \   ...
       \col 1 /    \col 3 /
   ...  \____/  ... \____/
           ...         ...
```

Figure 2: Example hexagonal grid showing the order that rows and columns are indexed

Two cells in the hex grid are considered adjacent if they share an edge. For each non-border cell, there are 6 adjacent cells.

The BeeBot Agent and its Actions

The BeeBot Bee occupies a single cell in the hex grid. In the visualisation, the Bee is represented by the cell marked with the character 'R' (or the Bee in the graphical visualiser). The side of the cell marked with '*' (or the arrow in the graphical visualiser) represents the front of the Bee. The state of the Bee is defined by its (row, column) coordinates and its orientation (i.e. the direction its front side is pointing towards).

At each time step, the agent is prompted to select an action. The Bee has 4 available actions:

- Forward $\rightarrow$ move to the adjacent cell in the direction of the front of the Bee (keeping the same orientation)
- Reverse $\rightarrow$ move to the adjacent cell in the opposite direction to the front of the Bee (keeping the same orientation)
- Spin Left $\rightarrow$ rotate left (relative to the Bee's front, i.e. counterclockwise) by 60 degrees (staying in the same cell)
- Spin Right $\rightarrow$ rotate right (i.e. clockwise) by 60 degrees (staying in the same cell)

The Bee is equipped with a gripper on its front side which allows it to manipulate honey Widgets. When the Bee is positioned with its front side adjacent to a honey Widget, performing the 'Forward' action will result in the Widget being pushed, while performing the 'Reverse' action will result in the honey Widget being pulled.

Action Costs

Each action has an associated cost, representing the amount of energy used by performing that action.

If the Bee moves without pushing or pulling a honey widget, the cost of the action is given by a base action cost, `ACTION_BASE_COST[a]` where 'a' is the action that was performed.

If the Bee pushes or pulls a honey widget, an additional cost of `ACTION_PUSH_COST[a]` is added on top, so the total cost is `ACTION_BASE_COST[a] + ACTION_PUSH_COST[a]`.

The costs are detailed in the `constants.py` file of the support code:

```
ACTION_BASE_COST = {FORWARD: 1.0, REVERSE: 1.0, SPIN_LEFT: 0.1, SPIN_RIGHT: 0.1}
ACTION_PUSH_COST = {FORWARD: 0.8, REVERSE: 0.5, SPIN_LEFT: 0.0, SPIN_RIGHT: 0.0}
```

Obstacles

Some cells in the hex grid are obstacles. In the visualisation, these cells are filled with the character ‘X’ (represented as black cells in the graphical visualiser). Any action which causes the Bee or any part of a honey Widget to enter an obstacle cell is invalid (i.e. results in collision). The outside boundary of the hex grid behaves in the same way as an obstacle.

Widgets

Honey Widgets are objects which occupy multiple cells of the hexagonal grid, and can be rotated and translated by the BeeBot Bee. The state of each honey widget is defined by its centre position (row, column) coordinates and its orientation. Honey Widgets have rotational symmetries - orientations which are rotationally symmetric are considered to be the same.

In the visualisation, each honey Widget in the environment is assigned a unique letter ‘a’, ‘b’, ‘c’, etc. Cells which are occupied by a honey Widget are marked with the letter assigned to that Widget (surrounded by round brackets). The centre position of the honey Widget is marked by the uppercase version of the letter, while all other cells occupied by the widget are marked with the lowercase.

Three honey widget types are possible, called Widget3, Widget4 and Widget5, where the trailing number denotes the number of cells occupied by the honey Widget. The shapes of these three honey Widget types and each of their possible orientations are shown in Figures 3 to 5 below.

```
      SLANT_LEFT
   _____
  /     \
 /  (a)  \_____
 \       /     \
  \_____/  (A)  \_____
        \       /     \
         \_____/  (a)  \
               \       /
                \_____/
```

```
  VERTICAL
   _____
  /     \
 /  (a)  \
 \       /
  \_____/
  /     \
 /  (A)  \
 \       /
  \_____/
  /     \
 /  (a)  \
 \       /
  \_____/
```

```
         SLANT_RIGHT
                _____
               /     \
         _____/  (a)  \
        /     \       /
  _____/  (A)  \_____/
 /     \       /
/  (a)  \_____/
\       /
 \_____/
```

Figure 3: Widget3

```
          UP
         _____
        /     \
       /  (a)  \
       \       /
        \_____/
        /     \
  _____/  (A)  \_____
 /     \       /     \
/  (a)  \_____/  (a)  \
\       /     \       /
 \_____/       \_____/
```

```
         DOWN
  _____         _____
 /     \       /     \
/  (a)  \_____/  (a)  \
\       /     \       /
 \_____/  (A)  \_____/
       \       /
        \_____/
        /     \
       /  (a)  \
       \       /
        \_____/
```

Figure 4: Widget4

```
      HORIZONTAL
  _____         _____
 /     \       /     \
/  (a)  \_____/  (a)  \
\       /     \       /
 \_____/  (A)  \_____/
 /     \       /     \
/  (a)  \_____/  (a)  \
\       /     \       /
 \_____/       \_____/
```

```
      SLANT_RIGHT
         _____
        /     \
       /  (a)  \_____
       \       /     \
        \_____/  (a)  \
        /     \       /
  _____/  (A)  \_____/
 /     \       /
/  (a)  \_____/
\       /     \
 \_____/  (a)  \
       \       /
        \_____/
```

```
      SLANT_LEFT
         _____
        /     \
  _____/  (a)  \
 /     \       /
/  (a)  \_____/
\       /     \
 \_____/  (A)  \_____
       \       /     \
        \_____/  (a)  \
        /     \       /
       /  (a)  \_____/
       \       /
        \_____/
```

Figure 5: Widget5

Two types of widget movement are possible – translation (change in centre position) and rotation (change in orientation).

Translation occurs when the Bee is positioned with its front side adjacent to one of the honey widgets' cells such that the Bee's orientation is in line with the honey widget's centre position. Translation results in the centre position of the widget moving in the same direction as the Bee. The orientation of the honey Widget does not change when translation occurs. Translation can occur when either 'Forward' or 'Reverse' actions are performed. For an action which results in translation to be valid, the new position of all cells of the moved widget must not intersect with the environment boundary, obstacles, the cells of any other honey Widgets or the Bee's new position.

Rotation occurs when the Bee's current position is adjacent to the centre of the widget but the Bee's orientation does not point towards the centre of the widget. Rotation results in the honey widget spinning around its centre point, causing the widget to change orientation. The position of the centre point does not change when rotation occurs. Rotation can only occur for the 'Forward' action - performing 'Reverse' in a situation where 'Forward' would result in a widget rotation is considered invalid.

The following diagrams show which moves result in translation or rotation for each honey Widget type, with the arrows indicating directions from which the Bee can push or pull a widget in order to cause a translation or rotation of the widget. Pushing in a direction which is not marked with an arrow is considered invalid.

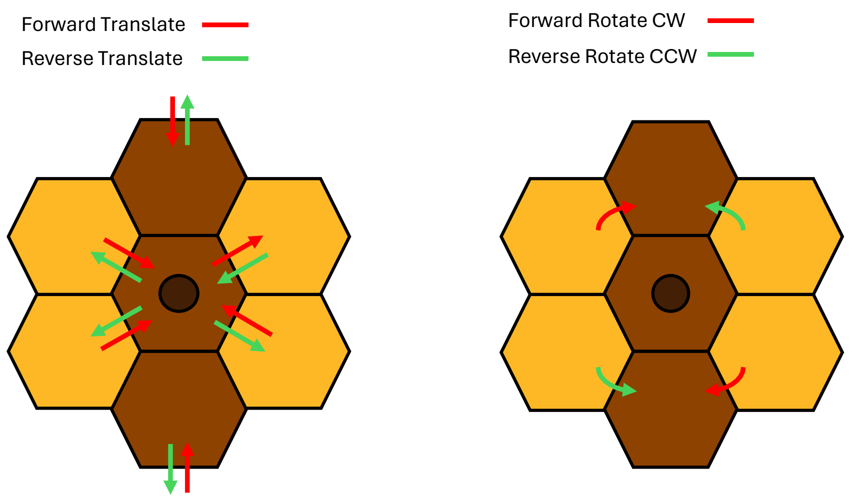

Figure 6: Widget3 translations and rotations

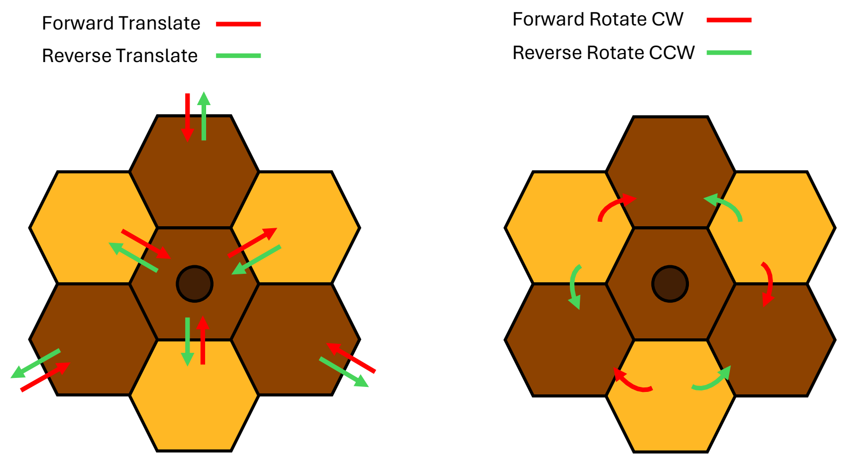

Figure 7: Widget4 translations and rotations

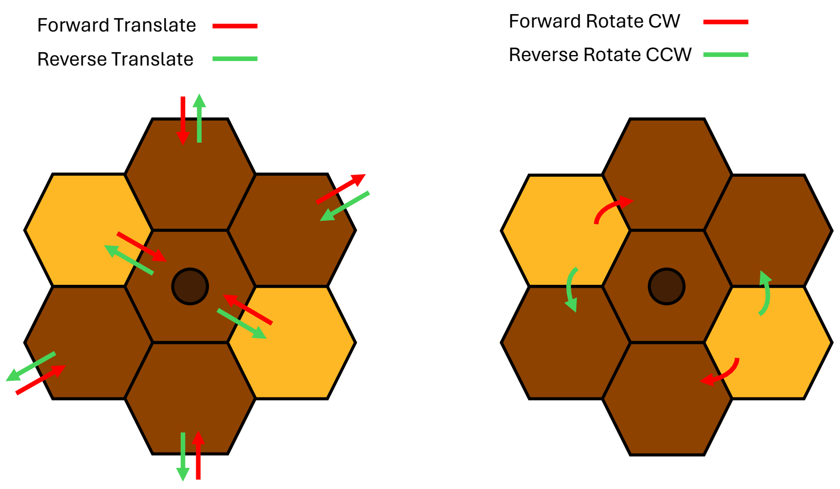

Figure 8: Widget5 translations and rotations

Targets
The hex grid contains a number of 'target' honeycomb cells which must be filled with honey. In the visualisation, these cells are marked with 'tgt' (cells coloured in orange in the graphical visualiser). For a BeeBot environment to be considered solved, each target cell must be occupied by part of a honey Widget. The number of targets in an environment is always less than or equal to the total number of cells occupied by all honey Widgets.

Interactive mode

A good way to gain an understanding of the game is to play it. You can play the game to get a feel for how it works by launching an interactive game session from the terminal with the following command for the graphical visualiser:

```
$ python play_game.py <input_file>.txt
```

or the following command for the command-line ASCII-character based visualiser:

```
$ python play.py <input_file>.txt
```

where `<input_file>.txt` is a valid testcase file (from the support code, with path relative to the current directory), e.g. `testcases/ex1.txt`.

Depending on your python installation, you should run the code using `python`, `python3` or `py`.

In interactive mode, type the symbol for your chosen action (and press enter in the command line version) to perform the action: press 'W' to move the Bee forward, 'S' to move the Bee in reverse, 'A' to turn the Bee left (counterclockwise) and 'D' to turn the Bee right (clockwise). Use 'Q' to quit the simulation, and 'R' to reset the environment to the initial configuration.