Chapter 12

Implementation

This chapter discusses the implementation details of the BW4T environment and details the interface which agents use to interact with the environment. Section 12.1 describes the modeling of the environment. Section 12.2 details the EIS interface developed for the agents.

12.1 Modeling the environment

The environment is implemented in two dimensional space since we are not interested in height as a third dimension.

BoundedMoveableObject The objects in the Java implementation of the BW4T environment are all based on the *BoundableMoveableObject* class. This class gives each object a unique integer for identification. It also offers basic functionality such as adding/removing the object from the world, putting objects in another position, changing the size of objects and doing intersection checks.

Robot The robot is modeled as an entity that is shaped like a square. It has a current location and a location it wants to move to. The robot is capable of picking up blocks that are a certain distance from it, much like a robot would have an arm. The robot also has a human-readable name for visualization purposes. The movement of the robot is done by moving at a set speed towards the location it wants to move to, slow-down or speed-up has not been added but would require little effort if found needed.

Room The rooms are modeled as rectangular areas. A robot is considered inside a room if it is entirely inside. Rooms are capable of having blocks placed inside of them however the room has no knowledge of this. In the implementation each room also has a color for visualization purposes.

The *DropZone* is a special kind of room which keeps track of the sequence of colors the blocks dropped in the zone need to adhere to to complete the mission.

Code Snippet 12.1 Text file defining a map in the BW4T environment.

```
100 100
R 45 45 10 10 B B R W
D 5 5 10 10 B R
E Bender 20 25
J WallE 20 20
```

A drop zone is capable of removing blocks from the world once a block has been dropped in it. Every block will be removed even if it doesn't have the right color, however the color sequence will not advance.

Blocks Blocks are modeled as rectangular objects, they have a color not only for visualization purposes but also important for the sequence in which they need to be delivered to a dropzone. Blocks know if they are being held by a Robot.

MapLoader The MapLoader is an important class that exposes a static method to load an environment from a text file. The text file that defines a map is similar to Snippet 12.1.

The first line in the map file specifies the width and the height of the map. Every next line indicates an object in the world, the first character of that line indicates which type of object we are dealing with.

The R indicates a room with at a certain (x, y) coordinate followed by the width and height and finally $[0, \infty >$ colors of blocks blocks that are put in this room upon initialization. The colors are coded as single characters and only a limited amount is supported at this time.

The D indicates a dropzone, only one of these may be present in a map file. It has the same specification as a room with one change, the block color indicators indicate the sequence of blocks of the dropzone.

The E stands for EIS entity and will spawn a Robot that can be controlled through EIS. The E is followed by the human readable name (without spaces) and the starting location of the robot in (x, y) format.

The J stands for a Robot that is controlled by Java itself, a programmer can built an implementation for it. The syntax is similar to that of a robot that can be controlled through EIS.

12.2 EIS

This section details the percepts and actions that are made available through EIS to control a Robot entity in the BW4T environment.

12.2.1 Percepts

This section details the percepts a robot can receive during the simulation.

Percept: at(*id*, *x*, *y*)

id The identifier of the object that is at the location.

x The x coordinate of the location.

y The y coordinate of the location.

Description Percept that tells the location of the object with the given id. This can be used to find out where to navigate to but also to find out where the robot itself is.

Percept: block(*id*)

id The identifier of the block.

Description Percept identifying that a certain id belongs to a block.

Percept: room(*id*)

id The identifier of the room.

Description Percept identifying that a certain id belongs to a room.

Percept: dropZone(*id*)

id The identifier of the drop zone.

Description Percept that couples an identifier to a drop zone.

Percept: robot(*id*)

id The identifier of a robot.

Description Percept identifying that a certain id belongs to a robot.

Percept: sequence(*seq*)

seq List of color identifiers.

Description This percept contains the color order in which blocks need to be delivered to the dropzone. Once a block, with the right color, has been delivered the sequence is updated to reflect that fact.

Percept: color(*id*, *c*)

id The identifier of the block.

c The character that represents the color of the block.

Description This percept gives information about which color a certain block has.

Percept: holding(id)

id The identifier of the block that is being held by the robot.

Description The percept gives information about which block the robot is currently holding.

12.2.2 Actions

This section introduces the actions a robot can perform. Although they are very basic it gives the robot the exact power it needs to perform the task at hand.

Action: goTo(*x*, *y*)

x The x coordinate of the position to move to.

y The y coordinate of the position to move to.

Description Instructs the robot to move to the given coordinates.

Action pickUp(*id*)

id The identifier of the block to pick up.

Description The robot will try and pickup the block. The action will fail if the block is not in range of the robots arm.

Action putDown

Description The robot will drop the block it is holding at its current position. If the current position is in the dropzone the block will be removed from the environment and the sequence may progress, if and only if the block is of the right color.