Interactive Blocksworld Application Manual

This manual describes all steps necessary to obtain the source code the Interactive Blocksworld application. Additionally it is shown how the application can be used in the Unreal Engine editor. Furthermore, configuration settings in the game are elaborated.

Obtaining the Source Code

The source code for the application can be downloaded in this archive. You can click on the "Source Code" entry to download the source code of the application. The source code also contains the files that have to be added to Fast Downward.

Usage of the code and the application

The code was generated using the Unreal Engine version 5.1.1. It is not recommended to import the code with another Engine version. To use the application in the Unreal editor, a new project must be created with the engine (recommended version: 5.1.1). From there config, content and source files can be copied and pasted in the newly created project. In some cases, for the editor to work the UProject file containing the project ID is necessary. This file is also contained in the source code.

The application can be extended by adding user interface functionality. This is mostly done in Blueprints in the content folder. Other implementations are written in C++ in the content folder. The application can be packaged in the editor. For this, the shipping option has to be chosen.

The application uses a third-party library: the graph visualization tool graphviz. This tool has to be linked. The graphviz library can be obtained by the following command on Ubuntu/Debian distributions:

sudo apt install graphviz

For other distributions the command is a bit different. The path where the graphviz library must be located is specified in the Blocks.build.cs file in the source folder. This location can also be changed or made customizable.

Fast Downward can be downloaded on the dedicated GitHub page (https://github.com/aibasel/downward). Additionally, some adjustments have to be done in order to use Fast Downward in the application. The application uses a self defined Blocksworld heuristic. This heuristic has to be added to the Fast Downward code. The adjustments for the planner can also be downloaded with the source code of the application. They are contained in the DownwardAdditions folder. The source and header file of the blocks heuristic have to be added to the heuristic folder in the search directory of Fast Downward. Additionally, the cmake DownwardFiles file in the search directory has to be appended with the definition of the Blocksworld plugin. Therefore, the following lines have to be added in the DownwardFiles file:

fast_downward_plugin(
NAME BLOCKS_HEURISTIC
HELP “The ‘blocks’ heuristic”
SOURCES
heuristics/blocks_heuristic
DEPENDS TASK_PROPERTIES
)

After these changes, the blocks heuristic can be used in the planner.
To use the Fast Downward planner in the application, the configuration file must be adjusted in the application. The configuration file consists of a path to the Python interpreter and a path to the fastdownward.py file.

The application provides a system where states can be saved. The purpose is that states with different difficulties can be generated and used for different target groups. Either these states are generated in the application by applying actions. Otherwise, the SavedStates file can be altered to generate new initial states. The format used in the file is as follows:

NAME_OF_STATE: ID_LANDSTACK1 ID_LANDSTACK1;ID_SHIPSTACK1 ID_SHIPSTACK1,ID_SHIPSTACK2 ID_SHIPSTACK2,ID_SHIPSTACK3 ID_SHIPSTACK3;ID_ATTACHEDCONTAINER

An example instance of a saved state would be:

SavedState: 3;0 1 2,4 5,7 8;6