GraphBT Team

Agung Pratama
Ardi
Chairunnisa Atimas
Emerson Chan Simbolon
Ikhsanul Habibie

GraphBT
Release Notes

Version 1.0

Revision History

Date	Version	Description	Author
12/September/2012	1.0	Initial Release	GraphBT Team

Table of Contents

Release Notes

1. Introduction

1.1 Disclaimer of warranty

GraphBT Team makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

This plugin uses Eclipse Public License http://www.eclipse.org/org/documents/epl-v10.php. Anyone can distribute, use, and contribute to the project freely without limitation as long as mentioning GraphBT Team as the original developer.

1.2 Purpose

The purpose of the Release Notes is to communicate the major new features and changes in this release of the GraphBT. It also documents known problems and work-arounds.

1.3 Scope

This document describes the GraphBT plugin.

1.4 Definitions, Acronyms and Abbreviations

1.4.1 Plugin

This word refers to GraphBT.

1.4.2 Document

This word refers to this document.

1.5 References

N/A

1.6 Overview

This document consists of explanation about features, platform compatibility, etc.

2. About This Release

This is the initial release of GraphBT plugin.

3. Compatible Products

This plugin has been tested on:

- Operating Systems: Windows 7, Windows 8
- RAM: 2 GB
- Processor: 1.66 GHz
- Resolution: 1366 x 768, 1280 x 720
- Eclipse Indigo SR2 Modeling

4. Upgrading

For the upgrading process, we will use this following update site: http://mrzon.github.com/GraphBT

5. New Features

1. BT Code Generator
2. BT Model Checker
3. BT Debugger Tool and Execution Simulator

4. GraphBT as Diagram and Textual Editor for Behavior Tree
5. Integrated tool which consists of all the tools listed above

6. Known Bugs and Limitations

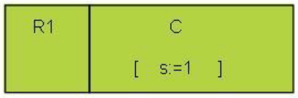
6.1 General Note

The project is maintained using Github (https://github.com/mrzon/GraphBT). Before start contributing, the list of prerequisite action can be accessed at https://github.com/mrzon/GraphBT/wiki/Installing-Guide. The documentation of the tool is still on progress and will be evolved as long as the code is improved. However, it can be accessed at http://mrzon.github.com/GraphBT/doc. The wiki page is containing important information to gaining more knowledge of the tool and can be accessed at https://github.com/mrzon/GraphBT/wiki. The application project page is available at http://mrzon.github.com/GraphBT. The information of the tool is available from there.

6.2 Defect or Bug

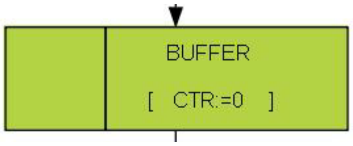
As a graphical editor, GraphBT supports all of the diagram syntax just like TextBE. Unfortunately, there are some limitations for several features such as Code Generator, Debugger and Simulation Tool which only support the following properties of BT:

1. *State Realization*



State Realization

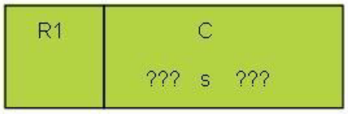
C[s] -> …
- A condition that one component could realize
- State c:
  - Enumeration {Cold, Hot, Warm}
  - Assignment (x:=2, t:=False)
  - Action (put what Food where (to) Oven)
  - Statement ([{}], in SetNotation, the component realize an empty Set condition), a statement needs formal representation



Example use of State Realization

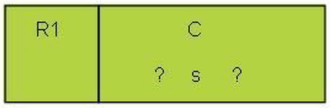
2. *Guard*

- C ??? s ??? ; …
- Reflect as “While” loop in programming language
- A branch will be executed if component C realize state „s‟, if not will block the child node
- In illustration below, producer will lock the buffer only if the buffer not in locked condition



Guard Node

3. *Selection*

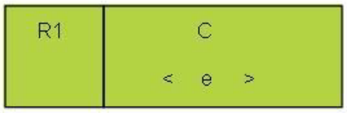


Selection Node

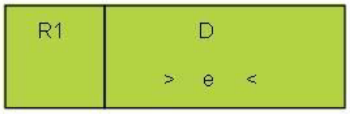
- C ?condition? -> …
- Reflect as "If" block in programming language
- A branch will be executed if satisfied the condition (if more than one satisfied, than it will choose one branch un-deterministically)
- If the condition not satisfied, than terminated

4. *Internal Input – Internal Output*

- C < e >; D > e <; …
- Reflects as "internal message passing" in programming language
- C and D shares same behavior with different type so they can communicate, in case above, it is equivalent with `D.setE(C.getE())`



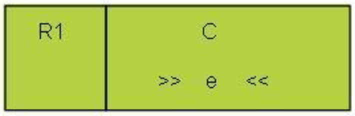
Internal Output



Internal Input

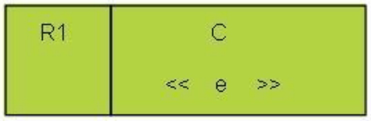
5. *External Input*

- C >>e<<; …
- Reflects as "input" request in programming language
- Will execute sub tree below, if C meets the event e, block if not



External Input

6. *External Output*



External Output

- C <<e>>;…
- Reflects as “output” to environment in programming language
- Sends event e to environment, e.g. print it out