BhTSL, Behavior Trees Specification and Processing

Miguel Oliveira
Centro ALGORITMI, DI, Universidade do Minho, Portugal

Pedro Mimoso Silva
Centro ALGORITMI, DI, Universidade do Minho, Portugal

Pedro Moura
Centro ALGORITMI, DI, Universidade do Minho, Portugal

José João Almeida
Centro ALGORITMI, DI, Universidade do Minho, Portugal

Pedro Rangel Henriques
Centro ALGORITMI, DI, Universidade do Minho, Portugal

Abstract

In the context of game development, there is always the need for describing behaviors for various entities, whether NPCs or even the world itself. That need requires a formalism to describe properly such behaviors.

As gaming industry has been growing, many approaches were proposed. First, finite state machines were used and evolved to hierarchical state machines. As this wasn't enough, a more powerful concept appeared. Instead of using states for describing behaviors, people started to use tasks. This concept was incorporated in behavior trees.

This paper focuses in the specification and processing of these behavior trees. A DSL designed for that purpose will be introduced. It will also be discussed a generator that produces LaTeX diagrams to document the trees, and a Python module to implement the behavior described. Aditionally, a simulator will be presented. These achievements will be illustrated using a concrete game as a case study.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Game development, Behavior trees, DSL, NPC, Code generation

Digital Object Identifier 10.4230/OASIcs.CVIT.2016.23

Acknowledgements I want to thank ...

1 Introduction

In some point of the video-game history, NPCs were introduced. With them came the need to describe behaviors. And with this behaviors came the need of the existence of a formalismm so that they can be properly specified.

As time passed by, various approaches were proposed and used. Finite state machines, hierarchical state machines and decision trees are some examples. However some companies decided that it was easy to humans to understand behaviors instead of states, so originated behavior trees used to model NPCS. It was first used in games like Halo, Bioshock, and Spore, and then other gaming companies saw the potencial, and now they are being used in game environments such as PyGame, Unreal Engine and Unity that are a majority of the platforms used to create games. In recent days it expanded to be used in Artificial Inteligence and in Robotics

Behavior trees became popular for their development paradigm: being able to create a complex behavior by only programming the NPC's actions and then designing a tree structure (usually through drag and drop) whose leaf nodes are actions and whose inner nodes determine the NPC's decision making. Behavior trees are visually intuitive and easy to design, test, and debug, and provide more modularity, scalability, and reusability than other behavior creation methods.

Over the years, the diverse implementations of behavior trees kept improving both in efficiency and capabilities to satisfy the demands of the industry, until they evolved into event-driven behavior trees[15]. Event-driven behavior trees solved some scalability issues of classical behavior trees by changing how the tree internally handles its execution, and by introducing a new type of node that can react to events and abort running nodes. Nowadays, the concept of event-driven behavior tree is a standard and used in most of the implementations, even though they are still called "behavior trees" for simplicity.

- Motivation - Design goals, ir direto ao assunto

2 State of the Art

- conceitos básicos

3 Architecture and Specification

- Gramática - Especificação

4 Tools

- processador

5 Example

6 Conclusion