Imperial College London
Department of Computing

Pyrulan:
Automated lazy testing for Python

Final Year Individual Project

Interim Report

By: Lee Wei Yeong

Supervisor: Prof. Susan Eisenbach

2nd Marker:

November 2011

Abstract

"As games become technologically more advanced with the ages, there's never been a greater need for intelligent AI in games. Several methodologies exist, with the aim of making AI more human-like, making games more engaging and believable. From the days of using Finite State Machines to Hierarchical Logic, the different ways of approaching AI have substantially made AI more diverse. Behaviour Trees attempt to improve upon these techniques by allowing a modular, flexible and powerful architecture that is goal-orientated."

— Pyrulan

Acknowledgements

First of all, I would like to thank Professor Susan Eisenbach for supervising my project, and giving me invaluable advice and guidance throughout my work. I also thank Tristan Allwood for his support, ideas, encouragement, and the useful discussions we had about my project.

Secondly, I thank John Ayres for creating the original Stage, a language which has been incredibly well thought out and, during my use of it, has proved very robust. John has given me many useful suggestions and critique regarding Stage.

Also, I thank Namit Chadha for several discussions about our actor-based projects, and I thank John Field of IBM research for providing me with information about the Thorn language.

Many thanks to Chong-U Lim for his insight during our conversations.

Finally, I would like to thank my parents for their continued full support while I am at university.

Contents

1 Introduction

What is the problem? Why is it important? Why is it difficult to solve? What is the main idea? What issues are there to address?

Video games have become technologically more advanced over the years - exhibiting realistic graphic capabilities coupled with engaging gameplay. The area of Artificial Intelligence (AI) in video games has also arisen as an important factor in determining the quality of the game by measure of a player's experience. With increasing demand for more realistic computer controlled players, events and opponents capable of exhibiting human-like characteristics, AI has been a significant area of interest for the games industry and academics alike. Several methodologies exist to approach the modelling, scripting and the design of AI for video games, aiming to make game-controlled entities more realistic and intelligent. Behaviour Trees attempt to improve upon exisiting AI methodologies by being simple to implement, scalable to handle the most complex of game tasks and modular to improve reusability - ultimately improving the efficiency for developing and designing game AI.

In this project, we plan to use Behaviour Trees to design and develop an AI-controlled player for the commercial real-time strategy game DEFCON. We approach the design of Behaviour Trees for the AI using a behaviour-oriented methodology which we introduce as Behaviour oriented Design (BOD) and also intend for the AI to adapt and learn to play the game of DEFCON by allowing the AI to evolve into a better player automatically using evolutionary machine learning techniques. The project aims to showcase the feasibility of combining such machine learning techniques together with behaviour trees as a practical approach to developing an effective and intelligent AI player.

1.1 Motivation

1.1.1 Improvements of Behaviour Trees over traditional game AI

Behaviour Trees have been proposed as a new approach to the designing of game AI, with advantages over other AI approaches being its simple design, scalability and modularity. Behaviour Trees have been adopted for use in commercial games for various uses including controlling character animation and coordinating non-playable character(NPC) behaviours. An example of the use of Behaviour Trees in commercial game AI includes the AI developed for the commercial First-Person-Shooter(FPS) Halo2 [**?**]. Thus, the project wishes to investigate the feasibility of Behaviour Trees for the purpose of designing an automated AI player for a real-time strategy(RTS) game such as DEFCON.

1.1.2 Complexity of DEFCON as a Real-Time Strategy Game

Real-time strategy (RTS) games offer a level of complexity which differ from other games such as platform games, puzzle games or even first-person-shooters. There exists a need to perform micro-managing over the numerous units in the game, as well as adopting strategies to be able to outwit opponents. DEFCON differs from other RTS games as well. In most other RTS games, winning heavily centres around accumulating resource and maximising the size of one's army or number of units. In DEFCON, however, players have an equal set of resources and number of units. The key factor to winning centres around a coming up with a good strategy to coordinate one's units and resources effectively. DEFCON thus exists as a challenging and interesting platform to develop an AI for.

2 Background

2.1 Research

2.2 Approach

2.3 Current state of the art

Cite existing related work

2.4 Motivation

2.5 Target audience

3 Body

3.1 Specification

3.2 Architecture

3.3 Algorithm outline

4 Evaluation

Which aspects (qualitative/quantitative)? Comparison with existing work? Results of tests/benchmarks?

5 Plan of work

5.1 Remaining tasks

5.2 Estimated timetable

5.3 Possible extensions

Relative priority? Rationale?

Bibliography

[ACE11] Tristan Allwood, Cristian Cadar, and Susan Eisenbach. High Coverage Testing of Haskell Programs. In *International Symposium on Software Testing and Analysis*, July 2011.

[BAR09] Yosi Ben Asher and Nadav Rotem. The effect of unrolling and inlining for python bytecode optimizations. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, SYSTOR '09, pages 14:1–14:14, New York, NY, USA, 2009. ACM.

[CKK+05] Yoonsik Cheon, Myoung Yee Kim, Myoung Yee Kim, Ashaveena Perum, and Ashaveena Perum. A complete automation of unit testing for java programs. In *In Proceedings of the 2005 International Conference on Software Engineering Research and Practice*, pages 290–295. CSREA Press, 2005.

[GR08] Nirmal Kumar Gupta and Mukesh Kumar Rohil. Using genetic algorithm for unit testing of object oriented software. In *Proceedings of the 1st International Conference on Emerging Trends in Engineering and Technology (ICETET '08)*, pages 308–313. IEEE, July 2008.

[MMSW97] C. C. Michael, G. E. McGraw, M. A. Schatz, and C. C. Walton. Genetic algorithms for dynamic test data generation. In *Proceedings of the 12th international conference on Automated software engineering (formerly: KBSE)*, ASE '97, pages 307–, Washington, DC, USA, 1997. IEEE Computer Society.

[Pym10] Pymodel: Model-based testing in python. `http://staff.washington.edu/jon/pymodel/www/`, March 2010.

[Reg11] Alex Groce & Chaoqiang Zhang & Eric Eide & Yang Chen & John Regehr. Swarm testing. In *Swarm Testing*. Oregon State University, Corvallis, OR; University of Utah, Sep 2011.

[SG06] Arjan Seesing and Hans-Gerhard Gross. A genetic programming approach to automated test generation for object-oriented software. *International Transactions on Systems Science and Applications*, 1(2):127–134, September 2006. Special Issue Section on Evaluation of Novel Approaches to Software Engineering Guest Editors: Pericles Loucopoulos and Kalle Lyytinen.

[Whi01] R. Whitehead. *Leading a software development team: a developer's guide to successfully leading people and projects*. The SEI Series in Software Engineering. Addison-Wesley, 2001.