

Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Bachelor's thesis

**VIZIA: 3D VIDEO GAME-BASED ENVIRONMENT FOR
RESEARCH ON LEARNING AGENTS FROM RAW VISUAL
INFORMATION**

Michał Kempka, 105256
Grzegorz Runc, 109759
Jakub Toczek, 109704
Marek Wydmuch, 109746

Supervisor
Wojciech Jaśkowski, Ph. D.

Poznań, 2016

Tutaj przychodzi karta pracy dyplomowej;
oryginał wstawiamy do wersji dla archiwum PP, w pozostałych kopiach wstawiamy ksero.

Streszczenie

Zawartość streszczenia.

Abstract

Abstract's content.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Aims and scope	1
1.3	Thesis organization	1
1.4	Contributions	2
2	Framework Outline	3
2.1	Used Technologies	3
2.2	Architecture	3
2.3	Problems and Solutions	3
2.4	Performance	4
2.5	Building process	4
2.5.1	Prerequisites	4
2.5.2	Compilation	4
3	Application Programming Interface	5
3.1	Methods	5
3.2	Structures and Enumerations	5
3.3	Python Wrapper	5
3.4	Extended Examples	5
4	Scenarios	7
4.1	Definition	7
4.2	Tools	7
4.3	Advices?	7
4.4	Scenarios	7
4.4.1	Basic	7
4.4.2	Deadly Corridor	7
4.4.3	Defend the Center	8
4.4.4	Defend the Line	8
4.4.5	Deathmatch	8
4.4.6	Health Gathering	8
4.4.7	My Way Home	8
4.4.8	Predict Position	8
4.4.9	Take Cover	8
5	Experiments	9
5.1	Setting	9

5.2	AI Agent Design	9
5.3	Results	9
6	Conclusions	11
6.1	Achieved Goals	11
6.2	Future Work	11
	Bibliography	13
A	GitHub	15

Chapter 1

Introduction

1.1 Motivation

Deep Learning and Convolutional Neural Networks have become very popular in the last couple of years. DeepMind is a huge inspiration. Only 2D games have been researched so far, that's why we want to create a framework using 3D environment. Games are great for simulating 3 dimensional world and are perfect setting for Reinforcement Learning.

Stuff to mention:

- Deep Neural Networks
- Visual Learning, Convolutional Nets, AI
- Reinforcement Learning
- DeepMind atari
- 2D and 3D games

1.2 Aims and scope

TODO

- opensource lightweight, 3d, fps game/engine,
- total control over game's processing,
- customizable resolution, rendering parameters, no-display mode etc.
- spectator mode (human is playing, agent is watching),
- custom scenarios support and creation,
- reinforcement learning friendly API (state, action, reward),
- support for Linux, Windows, OS X, main focus on Linux,
- C++ core, API in python, perhaps in lua, java etc.

1.3 Thesis organization

Thesis structure

1.4 Contributions

Who did what

Chapter 2

Framework Outline

2.1 Used Technologies

What we used and why.

- zdoom, mention alternatives
- linux focus, cpp core, python wrapper
- acs scripting in doombuilder 2 for scenarios
- python and lasagne for experiments

2.2 Architecture

Nice diagram (in DOOM style) with the architecture.

- Zdoom separate process.
- Boost interprocess: shared memory to communicate with zdoom.
- Flow control and PLAYER vs SPECTATOR mode.
- Warnings and exceptions.

2.3 Problems and Solutions

- Why shared memory and separate doom process and what it entails.
- Why make/set action are like they are. Why action is a vector not just number.
- Why state is copied in Python but not in cpp.
- Zbuffer struggles.
- Why Windows and Mac are not supported so well.
- Why scenario is effectively divided into config file nad doom iwad file.
- Why multiplayer is barely usable.

2.4 Performance

Table with some fps ratings and a graph. Conclusions: it's fast enough, any reasonably good AI will be much slower during learning process.

2.5 Building process

2.5.1 Prerequisites

- preferably linux
- cmake
- make
- gcc 4.??
- boost v ?
- python 2.6 with numpy (v?) for pyhon wrapper
- java ? for java wrapper
- ...

2.5.2 Compilation

cmake, make and they lived happily ever after

Chapter 3

Application Programming Interface

This chapter describes C++ api of the framework. "Methods" and "Structures and Enumerations" sections describe methods and structures exposed by api along with short examples if needed. "Python wrapper" outlines differences between C++ and Python Api. "Extended examples" sections shows fully functional examples in a proper context.

3.1 Methods

All that is written in README (the api part) but nicer, more thorough and with examples

3.2 Structures and Enumerations

Just like above

- struct state
- enumeration types . . .or maybe move it to the appendix?

3.3 Python Wrapper

- naming convention is underscore not camelcase for all methods except for the constructor
- State is changed structurally: bufer is a numpy array and game variables are a Python list.
- getState COPIES the buffer and gameVariables, it doesn't happen in cpp.

3.4 Extended Examples

Chapter 4

Scenarios

To apply reinforcement learning we need a reward-driven environment. Modern state-of-the art AI solutions are not mature enough to cope with fully-fledged FPS game so availability of scenarios with simpler tasks and more transparent task-reward mechanics is crucial.

Creation scenarios is nice and easy so we created a couple of sample scenarios to show how it all works.

4.1 Definition

What a scenario is, what it does and what it does not.

4.2 Tools

A few words about Doom Builder 2, acs scripts, reference to zdoom wiki, screen from doom builder 2.

4.3 Advices?

How to easily achieve some most common tasks in acs scripts which are not so obvious and were used here. e.g. shaping rewards, infinite ammo, respawning, friendly monsters,

4.4 Scenarios

4.4.1 Basic

- motivation
- description
- screen

4.4.2 Deadly Corridor

- motivation

- description
- screen

4.4.3 Defend the Center

- motivation
- description
- screen

4.4.4 Defend the Line

- motivation
- description
- screen

4.4.5 Deathmatch

- motivation
- description
- screen

4.4.6 Health Gathering

- motivation
- description
- screen

4.4.7 My Way Home

- motivation
- description
- screen

4.4.8 Predict Position

- motivation
- description
- screen

4.4.9 Take Cover

- motivation
- description
- screen

Chapter 5

Experiments

This chapter shows that using Vizia for AI training is feasible. It was possible to train an AI agent on basic scenario described in chapter ...section ... <link>

5.1 Setting

what (and how) will be tested. Overall performance, training speed (time and learning steps) for different frame skip rates was suggested. Hardware used for the experiment.

5.2 AI Agent Design

- python, theano, lasagne
- Based on Google's DeepMind Atari DQN.
- Q learning, convolutional neural network, eps-greedy policy with linear epsilon decay, action replay
- pseudo code here?
- network architecture used in the experiment, fancy diagram of this architecture here? (there are 2 conv and 2 mlp layers so it can be drawn and still make sense)

5.3 Results

Graphs and conclusions . . .

Chapter 6

Conclusions

6.1 Achieved Goals

- Full control over 3D engine processing.
- Performance is satisfactory.
- Scenarios.
- Spectator mode (I hope).

6.2 Future Work

- Lua wrapper
- windows/mac ?
- Some more foolproofing and stability.
- Some better code commenting.
- Testing on Linux distributions more heavily used as servers?

Bibliography

Appendix A

GitHub

The thesis and the VIZIA OR WHATEVER framework are not-so-publicly available on the github server:

<https://github.com/Marqt/Vizia/>



© 2016 Michał Kempka, Grzegorz Runc, Jakub Toczek, Marek Wydmuch

Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Typeset using L^AT_EX in Computer Modern.

Bib_TE_X:

```
@mastersthesis{ VIZAI,  
  author = "Michał Kempka \and Grzegorz Runc \and Jakub Toczek \and Marek Wydmuch",  
  title = "{VIZIA: 3D Video Game-based Environment for Research on Learning Agents from Raw  
Visual Information}",  
  school = "Poznan University of Technology",  
  address = "Pozna{\n}, Poland",  
  year = "2016",  
}
```