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

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
- DeppMind atari
- 2D and 3D games

Mr Jaśkowski's note:
"Ten rozdzial musi byc dobry. Luźne rozwazania do wykorzystania:

There were: research in reinforcement learning in games (even 3d games), research in (reinforcement) learning from visual information. Little research (?) on reinforcement learning from visual information. No research on RL from 3d visual information (CHECK). No research on RL from 3d visual information in games.

Why games? 1) Formula 1 for AI. 2) Mature engines with good performance and scripting abilities (scenarios) 3) Popular, well-known worlds, 4) Realistic worlds (graphics) Why 3D games? Learning from visual information. DeepLearning advanced AI's *perception* abilities.

No software for such research => Need."

1.2 Aims and scope

Mr Jaśkowski's note:
"The main aim of this thesis is to...
Requirements:
..."

- 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 abd creation,
- reinforcement learning firendly 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. Who wrote what.

Chapter 2

Framework Architecture

2.1 Used Technologies

What we used and why.

- zdoom, whole table with alternatives alternatives (full page)
- 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 arcitecture.

- Zdoom separate process.
- Boost interprocess: shared memory to comunicate 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.

Chapter 3

Application Programming Interface

Mr Jaśkowski's note:
"Przykłady też będą w C++? Wydaje mi się, że to może być dość niewygodne, bo wiadomo, że C++ jest mniej czytelny niż taki Python.

Zastanawiam się czy z p. widzenia czytelności tego rozdziału nie lepiej byłoby jednak pokazać API Pythonowe, a następnie wskazać różnice z C++ i wskazać na to, że jest wrapper.
"

This chapter desribes C++ api of the framework. M̈ethods¨ and S̈tructures and Enumerations¨ sections describe methods and structures exposed by api along with short examples if needed. P̈ython wrapperöutlines differences between C++ and Python Api. Ëxtended examplesösections 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 apendinx?

3.3 Python Wrapper

- naming convention is underscore not camelcase for all methods except for the constructor
- State is changed structuraly: 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

”Zacząłbym jakoś tak:
To allow a researcher to evaluate learning agents in conditions of different characteristics, we provide a mechanism of scenarios.
In this chapter we describe...”

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 of 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

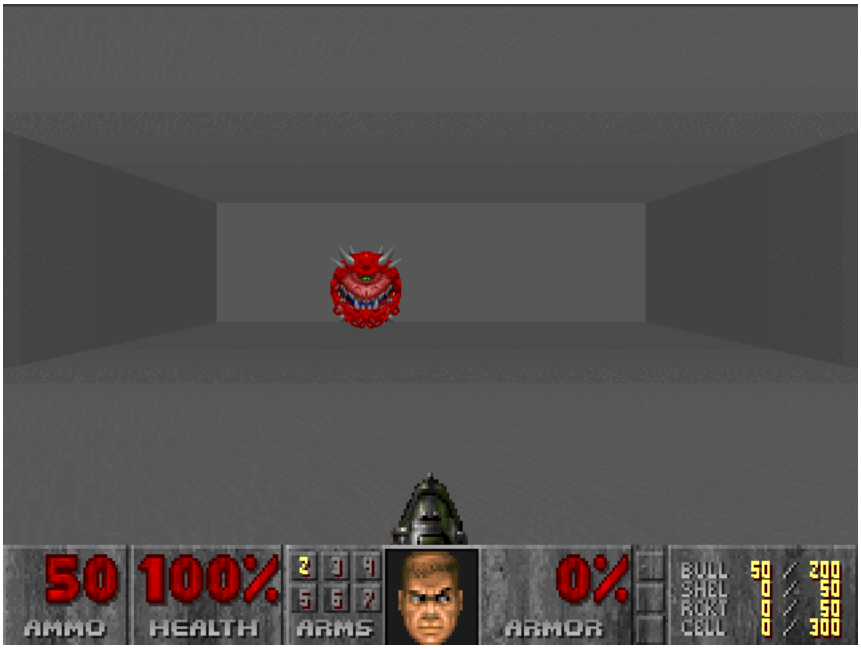

FIGURE 4.1: Doom gameplay frame from "basic" scenario

- motivation
- description

4.4.2 Deadly Corridor

Figure 4.2: Doom gameplay frame from "deadly corridor" scenario

- motivation
- description

4.4.3 Defend the Center

Figure 4.3: Doom gameplay frame from ”defend the center” scenario

- motivation
- description

4.4.4 Defend the Line

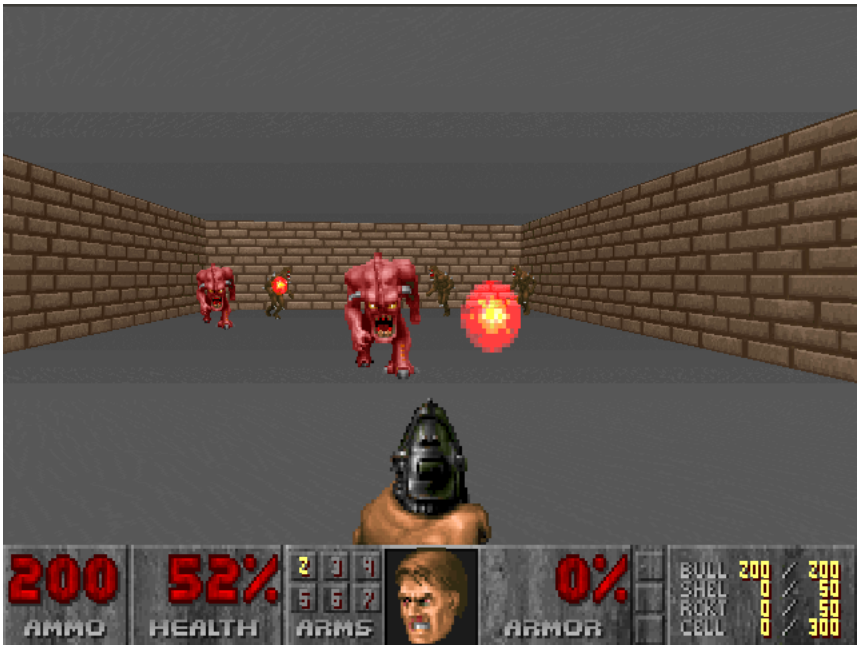

Figure 4.4: Doom gameplay frame from ”defend the line” scenario

- motivation
- description

4.4.5 Deathmatch

GRAPHICS SOON

- motivation
- description

4.4.6 Health Gathering

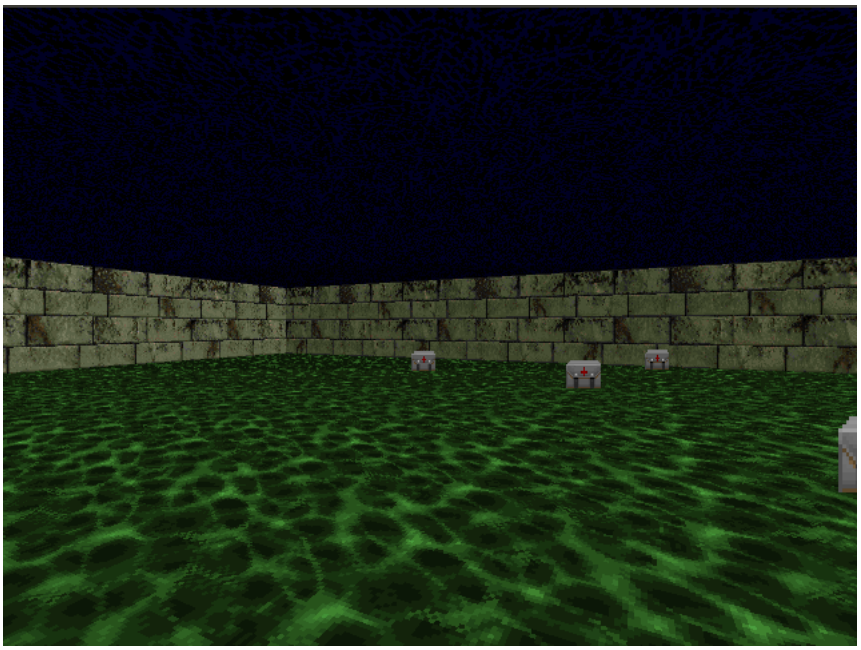

Figure 4.5: Doom gameplay frame from ”health gathering” scenario

- motivation
- description

4.4.7 My Way Home

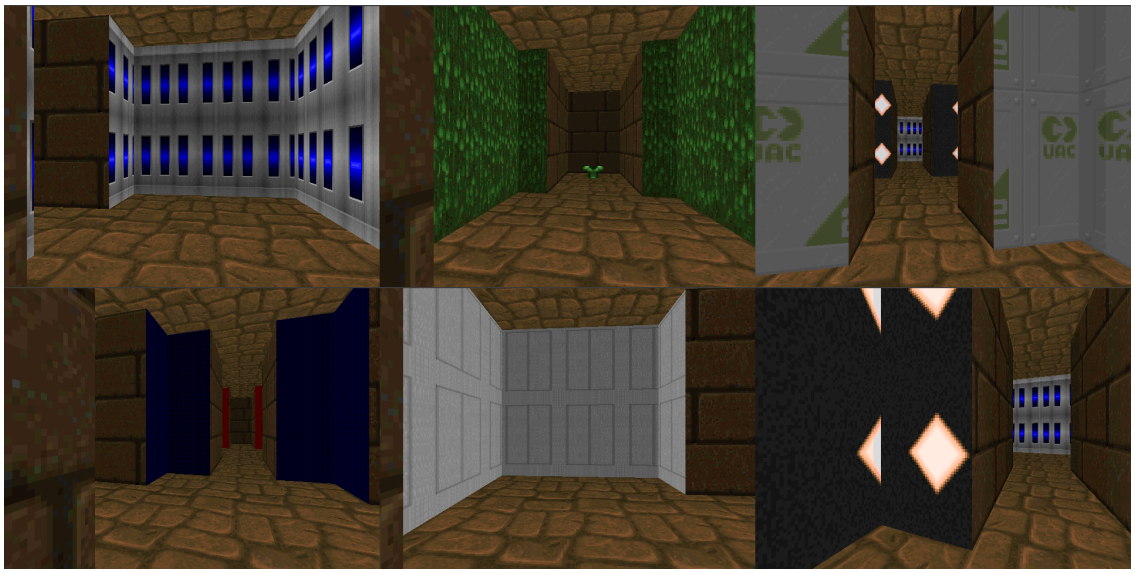

Figure 4.6: 6 random Doom gameplay frames from "my way home" scenario

- motivation
- description

4.4.8 Predict Position

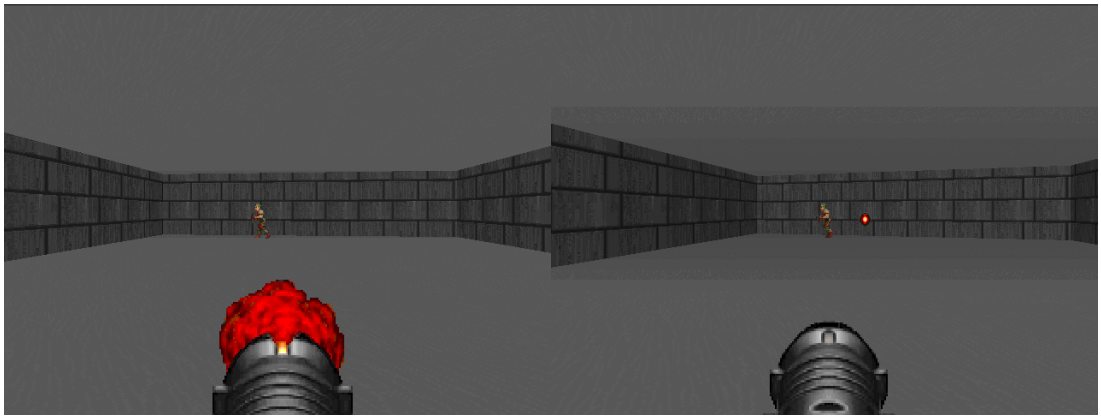

Figure 4.7: 2 random Doom gameplay frames from ”predict position” scenario

- motivation
- description

4.4.9 Take Cover

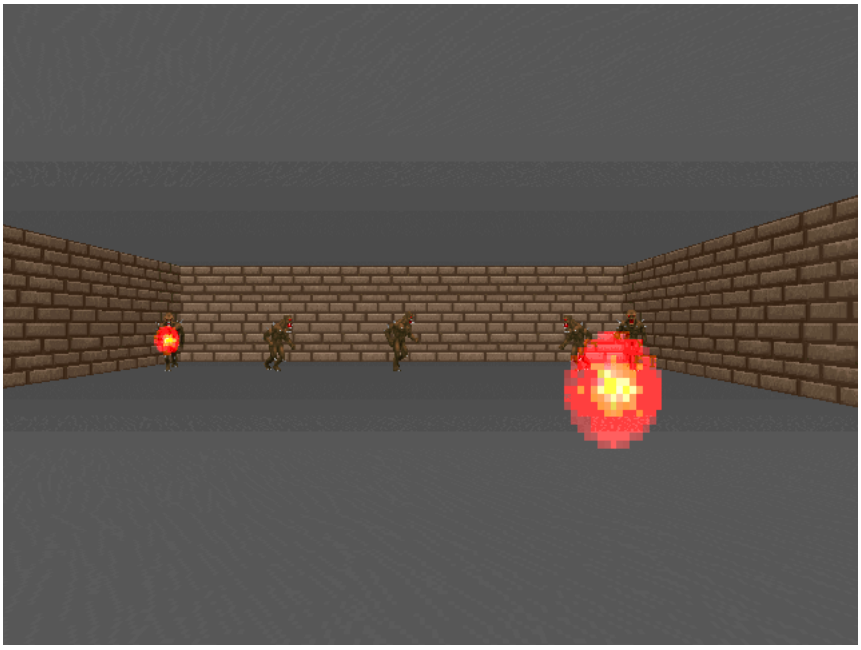

FIGURE 4.8: Doom gameplay frame from ”take cover” scenario

- motivation
- description

Chapter 5

Experiment

5.1 Goal of the Experiment

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

5.2 Experimental Settings

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

5.3 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? (ommit if short on time)
- 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)

5.4 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?
- Multiplayer (Added by MrJ.)

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/`

Appendix B

Building

B.1 Prerequisites

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

B.2 Compilation

cmake, make and they lived happily ever after

Appendix C

Methods and Structures Handout

C.1 Methods

C.2 Structures and Enumerations

Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Typeset using LaTeX in Computer Modern.

BibTeX:

```
@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",
}
```