# Towards an Application Programming Interface for Automated Testing of Artificial Intelligence Agents in Massively Multi-Player On-Line Role-Playing Games

**Markus Schatten, Bogdan Okreša Đurić, Igor Tomičić**
Artificial Intelligence Laboratory
Faculty áf árganization ánd ánformatics, University of Zagreb
Pavlinska á, á2000 áaraždin, ároatia
`{markus.schatten, dokresa, igor.tomicic}@foi.hr`

**Abstract.** *An initial implementation of an application programming interface (API) for automated testing of artificial intelligence agents in massively multi-player on-line role-playing games (MMORPGs) is presented and analyzed. The API, which is based on results from the Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Role-Playing Games (ModelMMORPG) project, is implemented in Smart Python Agent Development Environment (SPADE) and allows for the implementation of artificial agents which are able to play The Mana World (TMW), an open source MMORPG. The API provides access to basic player behaviours including but not limited to movement, fight, pick-up of items and interaction with non-player characters (NPCs) as well as more advanced features like interaction with other players through chat, creation of player parties and trade. The basic proposition of the API is to allow concurrent testing of various types of artificial agents in playing the game which allows for agent behaviour analysis as well as gameplay analysis.*

**Keywords.** application programming interface, automated testing, artificial intelligence, massively multi-player on-line role-playing game

## 1 Introduction

The gaming industry is recognized as a rapidly growing domain spanning various intended uses, from entertainment and education, to programming and design industries, to serious games and research. The importance and success of the gaming world is emphasized by the Electronic Sports (eSports) movement as well. As one of the identified application domains of large-scale multiagent systems (LSMASs), along with the Internet of Things (IoT), smart cities, smart grid, complex systems and smart transportation (Schatten, Tomičić, and Đurić, 2017), to name a few, the gaming industry gained a lot of attention in recent research trends, especially when MMORPGs are considered. Other than being a good test-bed for technologies such as virtual reality and augmented reality, the gaming domain succeeded in becoming an effective environment for social studies as well.

Computer games have been a source of training data for various approaches to development of a form of an artificial intelligence (AI) using different heuristics as well as machine learning methods. These forms of specialized AI were trained and developed for use with computer games that are of a simple point-and-click, first person shooter, or simple arcade game genre. Only recently has the gaming industry turned towards developing a form of a general AI, since games can be modified quickly, with many re-runs of experiments. Such a short response and customization time support the processes of learning when artificial agents are considered. In this context, Peng et al. have presented multiagent bidirectionally-coordinated network "with a vectorised extension of actor-critic formulation", which could "handle different types of combats under diverse terrains with arbitrary numbers of AI agents for both sides" (Peng et al., 2017). Authors used StarCraft game as a test-bed scenario and demonstrated that their network facilitates learning of various types of agent coordination strategies. A literature overview on AI techniques used in real-time strategy games was presented in (Robertson and Watson, 2014), with a focus on a StarCraft game. Authors have identified the main areas of current research: tactical and strategic decision making, plan recognition, and learning. Another application on StarCraft game was presented in (Synnaeve and Bessiere, 2016), where authors showed the use of Bayesian models in three distinct core components: micro-management/units control, tactics, and strategy, arguing the possibilities of probabilistic models on sources of uncertainty and incompleteness, which are inherent to the AIs in real-time strategy games.

This paper therefore presents initial steps towards implementation of an API for automated testing of AI agents in MMORPGs. Since testing an MMORPG is a tedious task that requires countless hours of gameplay, having it performed by an artificial agent greatly increases relevance of the received testing results, and the sole amount of such testing results.

The rest of this paper is organized as follows: firstly in section 2 we provide an overview of related work. Then in section 3 we show the architecture of the implemented API and in section 4 we discuss what additional steps have to be taken for the API to be complete. In the end in section 5 we draw our conclusions and provide guidelines for future work.

## 2 Related Work

The majority of related research is performed on simple games that usually use only two dimensions, and are linear in gaming style - either it's simply moving, collecting items, and avoiding obstacles, or almost only avoiding obstacles and moving in a single direction, or something similar.

The major distinctive difference between the published research and the research proposed in this paper, is the game genre that is put in the focus. While related research is focused primarily on games that are linear in the nature of their objectives, the API of this research is being built for an MMORPG which is a part of the role-playing games (RPGs) genre that usually utilizes non-linear story that is a combination of various quests and side-quests thus introducing a level of creativity in gameplay.

The topic of APIs for RPGs has received an increase in popularity when DeepMind and Blizzard (Vinyals et al., 2017) opened Blizzard's StarCraft 2 game as an AI research environment. The published set of tools designed for AI research combines the API developed by Blizzard, and a modified open source version of DeepMind's toolset that can be used to integrate Python-based agents with Blizzard's API. The combination of the two, along with the additional elements of the set of tools, can freely be used by researchers and developers in their AI-related pursuits. StarCraft 2 is a sequel and a modernized version of an old but highly popular and valued real-time strategy (RTS) computer game that has secured its popularity as a long-time champion in the domain of eSports. The cited research is an example and proof of the growing importance of computer games in research and their importance in AI development.

Another research on the topic of how RPGs can be used to further and enhance AI research is presented in (Tian et al., 2017), where a research-oriented platform is presented that conforms to a defined set of features of an ideal game environment for fundamental reinforced learning – one that is diverse in the context of properties of offered games, efficient in running simulations, and provides highly customized environment settings. RTSs are used as a way of providing richer environments for reinforced learning, ensuring that training using e.g. raw pixel data is supported, since not many RPGs exist that can be used for research directly (Tian et al., 2017). The research presented by Tian et al. (2017) therefore provides a platform for enhanced use of reinforced learning, in the context of computer games of RTS genre, which is significantly different than RPG games that are the focus of this paper's research.

Another take on the concept of machine learning, using visual input from a computer game for training the AI, is associated with a game of first-person shooter (FPS) genre (Kempka et al., 2017). *ViZDoom* is purposed to be used for developing bots that play Doom, the classical FPS computer game released in 1997. The game is played based on visual reinforced learning principles, with the software emulating actions corresponding to key presses and mouse movements used to control the gameflow, and providing the developer with some directly available in-game data. Doom, the computer game used in this context, is far more simple than an average modern RPG since the provided and requested gameplay is much simpler and does not require devised planning.

Classic machine learning approaches have been used in development of artificial agents that can play so-called platformer games, such as Super Mario World, Super Mario Kart, or Pacman. While all of these examples are interesting to observe from the point of view of using computer games in training an AI agent, they do not represent APIs by themselves.

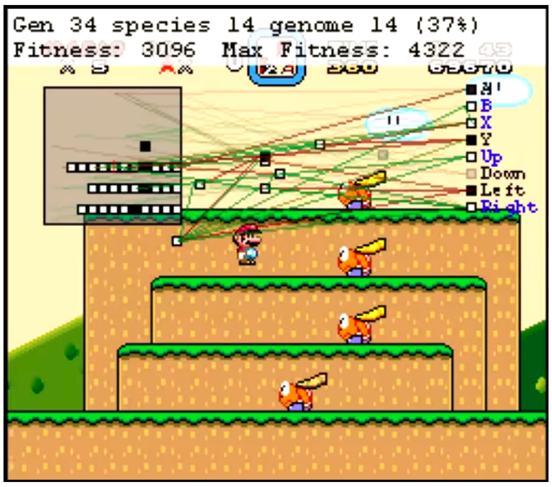

Gen 34 species 14 genome 14 (37%)
Fitness: 3096 Max Fitness: 4322
A
B
X
Y
Up
Down
Left
Right

**Figure 1:** MarI/O agent training, a screenshot from YouTube

MarI/O[1], shown on figure 1, is a piece of software developed by an Internet user known as SethBling that uses neural networks and generic algorithms to build an agent that is capable of successfully playing a simple Super Mario World game. The software interacts with the game using emulated keystrokes, just like seen in ViZDoom above. The use of visual input coupled with

[1]For further information, visit: https://www.youtube.com/watch?v=qv6UVOQ0F44, and https://pastebin.com/ZZmSNaHX

the use of recurrent neural networks is shown by the same author on another example, working with a driving computer game named Super Mario Kart – MariFlow[2]. As opposed to the earlier described example, this time the AI does not learn by genetic modifications, but is fed input data in form of sample games. The last such example presented here, for the sake of emphasizing the use of computer games in information communication technology (ICT) research, is another showcase of neural network methods being used to train and play a game, this time of Pacman[3].

## 3 Implementation

The importance and added value of the research laid out in this paper comes from the chosen application domain – MMORPGs. The difference of such a genre, when compared to some of the examples presented in Section 2, is obvious for those accustomed to computer games.

FPS is the most simple genre, when strategizing is considered, as it is based on destroying as many opponents as possible or demanded, and may be the simplest to implement, since the agent only needs to identify its target, and shoot at its most vulnerable point.

The genre of RTS is complex in terms of strategic thinking, but is mostly based on a single goal, packaged into successfully outrunning the opponent in the context of, usually, economic or militaristic prowess and power. While the possible ways of achieving victory are diverse and mostly depend on the nature of the given player and the way they want to interact with the game (e.g. emphasizing technological advancement, or building their armies in numbers), the goals are seldom more complex than those derived from the aforementioned context.

Finally, an RPG boasts a much more complex world in general, since many more decisive elements are present, more so than in either FPS or RTS genre. Furthermore, role-playing games, being based on developing a player's character (often referred to as an avatar), are built around a story that helps develop this avatar, most often defined by their various traits, features, abilities, inventory, etc. The in-game storyline is not as linear as that of a typical FPS or a typical RTS either. Although these two genre may provide a storyline to be followed during the game in extent, the gaming experience is heavily divided by levels that act as a whole. An RPG is more often modeled as an open-world game, allowing the players to roam the in-game world and choose objectives for themselves, while the game in general is directed by a set of quests that demand interaction with concepts of the in-game world. Thus the possible set of actions and their utilization is much larger than that of either of the aforementioned computer game genre.

In addition to computations in terms of devising plans on how to solve a quest, the context of MMORPGs demands solving social-level problems and challenges as well, since socialization is an important and integral part of the *massively multi-player* part of this genre. More so, socialization is very important in utilizing all the features of a MMORPG. Therefore, building an API for an MMORPG is not only about enabling an agent to play a game and interact with the in-game world, but to interact with other players as well, which is an element that is not present in research presented in Section 2.

During the ModelMMORPG project, which aimed on developing artificial agent organizations that are able to play MMORPGs, a byproduct was the development of an interface to a very particular open source MMORPG called TMW. The interface was implemented in Python and comprises two parts: (1) a lower-level interface – dealing with particular low-level communication with the game server; as well as a (2) higher-level interface – dealing with the actual implementation of a playing agent (Schatten, Tomičić, Đurić, and Ivković, 2017; Schatten, Đurić, Tomičić, et al., 2017; Schatten, Đurić, Tomičič, et al., 2017).[4] The architecture of the ModelMMORPG agent implementation system is shown on figure 2, on which the MMORPG plug-in is the interface to TMW. Detailed model of both lower- and higher level interfaces is shown in figure 3.

The lower-level interface, shown on figure 4 implements the network protocol of the TMW servers and emulates an actual client connecting to the server and playing the game. It also implements a number of functions that allow actual gameplay including navigation (moving the avatar including automated navigation like moving to some character or dropped item, following a player or mob etc.), attack, sitting down, standing up, whispering communication to other players, picking and dropping up items, equipping items, creating parties of players (including inviting, responding to invitations, leaving parties and communicating on the group chat), environment sensing (what is my location, what is the location of nearby players/items/NPCs), and trading of items with other players, interacting with NPCs.

The higher-level interface builds upon the lower-level by introducing an agent template implemented in SPADE (Gregori et al., 2006) and adding additional parts like a STRIPS-based planning system implemented in Prolog, as well as an agent knowledge base implemented using the SPADE knowledge-base system for SWI Prolog. The higher-level interface acts as a layer around the low-level interface by providing higher-level agent behaviours based on low-level func-

[2]For further information, visit https://www.youtube.com/watch?v=Ipi40cb_RsI

[3]For further information, visit: https://www.youtube.com/watch?v=t5--kLRI4UE

[4]The current version of the interface is available at https://github.com/tomicic/ModelMMORPG

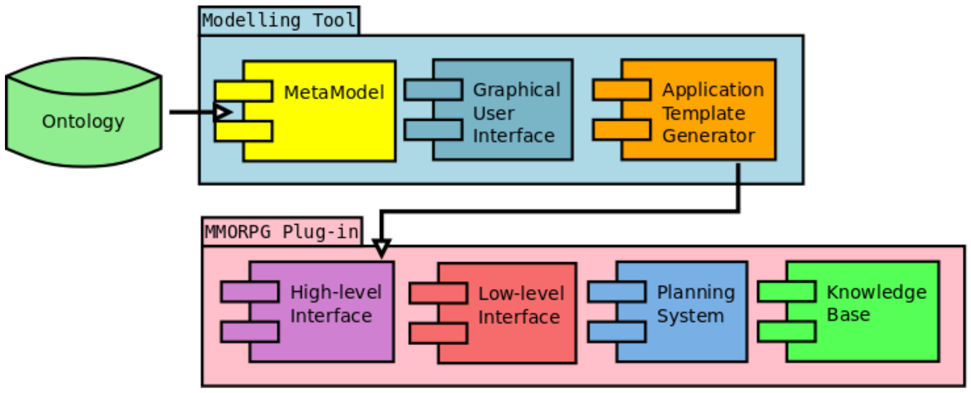


**Figure 2:** Architecture of the ModelMMORPG agent implementation system (Schatten, Đurić, Tomičić, et al., 2017)

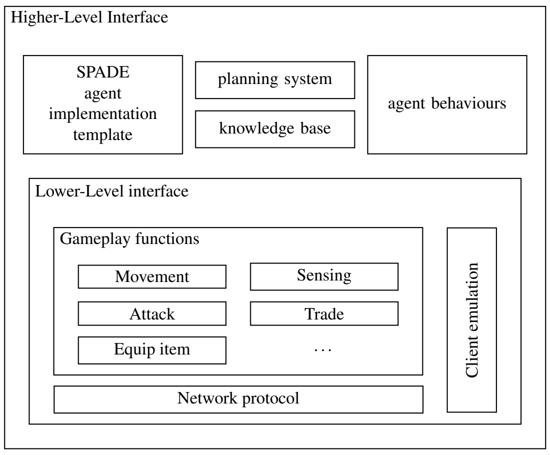


**Figure 3:** Detailed model of lower- and higher-level interfaces and their featuers

tion including but not limited to: navigation (including random walks), NPC conversation handling, fight handling, and party management. It is a typical belief-desire-intention (BDI) agent that firstly senses the environment, updates it's knowledge base, chooses an objective to accomplish next, generates a plan for this particular objective and then starts executing it, while the objective and the plan still make sense.

## 4 Discussion

In its current state, the interface cannot be really called an API, since a number of core functionality is missing, most prominently adequate documentation on how to implement an agent able to play TMW including a few simple examples as well as the abstraction of core functions from the actual implementation of an agent to be reusable. Currently, the implementation features a finished agent prototype that is able to play some of the initial quests of the game, as well as create, join or leave parties based on its assigned role. This prototype has to be dissected into parts that can be reused for new types of agents, possibly using different AI, reasoning and reaction technology like machine learning for example, since the prototype is based on BDI and STRIPS.

In order to do so, some basic rewriting of the code related to the high-level interface has to be done, for example: all prototype related code that was used for large-scale experiments has to be moved into an example usage, while core functionality like random walks, creation of parties, conversation with NPCs and similar have to be retained for future use. On the other hand, the low-level interface can stay more or less intact, since it is already used as a module for the high-level interface and acts as a low-level API.

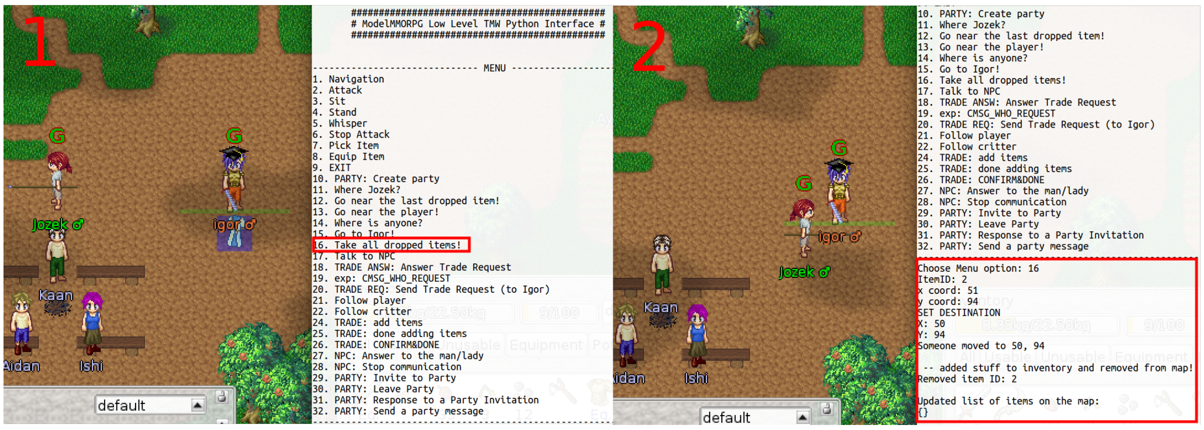
Also, the planning system as well as the knowledge base of the agent can be easily reused for future agents' implementation, but the interface towards them has to be made more accessible for new types of agents.

Even though the API is to be used for testing MMORPGs in the context of their logical, quest, and story soundness, and balanced gameplay, the basic use case of the work presented and proposed in this paper is to develop testing scenarios whose details depend on the developer and testing needs. Therefore the success of testing largely depends on the game developers and testers, and the implementation details beyond the scope of this research which aims to provide functions and agent behaviours necessary for successful interaction between agents and an MMORPG.

## 5 Conclusion

Herein we have presented an initial step towards the implementation of an API for the automated testing of AI agents for MMORPGs based on some byproducts of the ModelMMORPG project. We have implemented a lower-level interface, a higher-level interface, a STRIPS based automated planning system as well as a knowledge base for the future API that will allow for the implementation of AI agents for the open source MMORPG TMW.

The most important contribution of the API is its fo-



```
##############################################
# ModelMMORPG Low Level TMW Python Interface #
##############################################

------------------------------ MENU ------------------------------
1. Navigation
2. Attack
3. Sit
4. Stand
5. Whisper
6. Stop Attack
7. Pick Item
8. Equip Item
9. EXIT
10. PARTY: Create party
11. Where Jozek?
12. Go near the last dropped item!
13. Go near the player!
14. Where is anyone?
15. Go to Igor!
16. Take all dropped items!
17. Talk to NPC
18. TRADE ANSW: Answer Trade Request
19. exp: CMSG_WHO_REQUEST
20. TRADE REQ: Send Trade Request (to Igor)
21. Follow player
22. Follow critter
24. TRADE: add items
25. TRADE: done adding items
26. TRADE: CONFIRM&DONE
27. NPC: Answer to the man/lady
28. NPC: Stop communication
29. PARTY: Invite to Party
30. PARTY: Leave Party
31. PARTY: Response to a Party Invitation
32. PARTY: Send a party message
------------------------------------------------------------------
```

```
10. PARTY: Create party
11. Where Jozek?
12. Go near the last dropped item!
13. Go near the player!
14. Where is anyone?
15. Go to Igor!
16. Take all dropped items!
17. Talk to NPC
18. TRADE ANSW: Answer Trade Request
19. exp: CMSG_WHO_REQUEST
20. TRADE REQ: Send Trade Request (to Igor)
21. Follow player
22. Follow critter
24. TRADE: add items
25. TRADE: done adding items
26. TRADE: CONFIRM&DONE
27. NPC: Answer to the man/lady
28. NPC: Stop communication
29. PARTY: Invite to Party
30. PARTY: Leave Party
31. PARTY: Response to a Party Invitation
32. PARTY: Send a party message
------------------------------------------------------------------
Choose Menu option: 16
ItemID: 2
x coord: 51
y coord: 94
SET DESTINATION
X: 50
Y: 94
Someone moved to 50, 94

 -- added stuff to inventory and removed from map!
Removed item ID: 2

Updated list of items on the map:
{}
```

**Figure 4:** Lower-level interface to TMW (Schatten, Đurić, Tomičič, et al., 2017)

cus on a new genre, as opposed to current approaches. The RPG genre, and more recently the MMORPG genre hasn't been part of recent research for automated testing of AI agents, most probably due to the complexity of the domain, since such games include lots of complex behaviour of players to be exhibited in order to advance in the game.

Our future research will therefore be aimed towards completing the API by dissecting the prototype code into well established parts that can be reused in future AI agent implementation projects, by developing a number of simple AI agents for developers to start with as well as more profound documentation.

## Acknowledgments

This work has been fully supported by the Croatian Science Foundation under the project number 8537.

## References

Gregori, M. E., Cámara, J. P., & Bada, G. A. (2006). A jabber-based multi-agent system platform. In *Proceedings of the fifth international joint conference on autonomous agents and multiagent systems* (pp. 1282–1284). ACM.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaskowski, W. (2017). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. *IEEE Conference on Computatonal Intelligence and Games, CIG*. doi:10.1109/CIG.2016.7860433. arXiv: 1605.02097

Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*.

Robertson, G. & Watson, I. (2014). A review of real-time strategy game ai. *AI Magazine*, *35*(4), 75–104.

Schatten, M., Đurić, B. O., Tomičić, I., & Ivković, N. (2017). Agents as bots–an initial attempt towards model-driven mmorpg gameplay. In *International conference on practical applications of agents and multi-agent systems* (pp. 246–258). Springer.

Schatten, M., Đurić, B. O., Tomičić, I., & Ivkovič, N. (2017). Automated mmorpg testing–an agent-based approach. In *International conference on practical applications of agents and multi-agent systems* (pp. 359–363). Springer.

Schatten, M., Tomičić, I., & Đurić, B. O. (2017). A review on application domains of large-scale multiagent systems. In T. Hunjak, V. Kirinić, & M. Konecki (Eds.), *Central european conference on information and intelligent systems* (pp. 201–206).

Schatten, M., Tomičić, I., Đurić, B. O., & Ivković, N. (2017). Towards an agent-based automated testing environment for massively multi-player role playing games. In *Information and communication technology, electronics and microelectronics (mipro), 2017 40th international convention on* (pp. 1149–1154). IEEE.

Synnaeve, G. & Bessiere, P. (2016, December). Multiscale Bayesian Modeling for RTS Games: An Application to StarCraft AI. *IEEE Transactions on Computational Intelligence and AI in Games*, *8*(4), 338–350. doi:10 . 1109 / TCIAIG . 2015 . 2487743

Tian, Y., Gong, Q., Shang, W., Wu, Y., & Zitnick, C. L. (2017). ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games. (Nips), 1–14. arXiv: 1707.01067

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., . . . Tsing, R. (2017). *StarCraft II: A New Challenge for Reinforcement Learning*.