O_HAI(4)Games

Orchestration of Hybrid Artificial Intelligence Methods for Computer Games

Case Study 1 - MMORPGs

This project was funded by the Croatian Science Foundation

Principal investigator:

Markus Schatten

PUBLISHED BY ARTIFICIAL INTELLIGENCE LABORATORY,
FACULTY OF ORGANIZATION AND INFORMATICS, UNIVERSITY OF ZAGREB

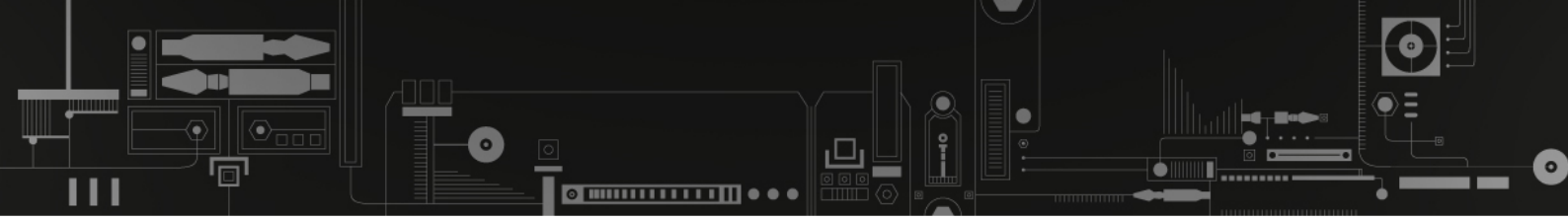
HTTP://AI.FOI.HR/OHAI4GAMES

Some of the results presented in this deliverable have been published in [48, 62].

Technical Report No. AIL202101 – First release, May 2021, edit September 2022

Document compiled by: Markus Schatten with inputs from other project team members

This work has been supported in full by the Croatian Science Foundation under the project number IP-2019-04-5824.

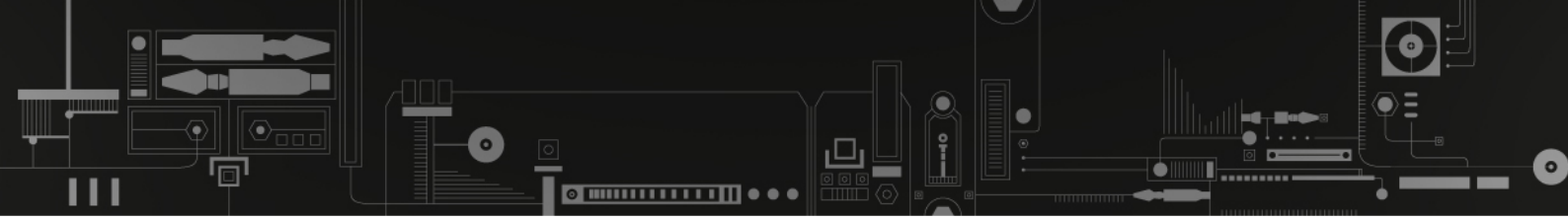


hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

Contents



hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

1. Project Description

1.1 Abstract

Hybrid artificial intelligence (HAI) methods, which can be defined as the orchestration of complementary heterogeneous both symbolic and statistical AI methods to acquire more precise results, are omnipresent in contemporary scientific literature. Still, the methodology of developing such systems is in the most cases ad-hoc and depends from project to project. Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games, which have often been judged by the quality of their AI and praised if they used an innovative approach. Computer games allow us to test AI methods, not only for fun and leisure, but also for numerous other fields of human activity through the fields of serious games and gamification. The project proposes to establish a new framework for the orchestration of hybrid artificial intelligence methods with a special application to computer games. Therefore an ontology of hybrid AI methods as well as a meta-model shall be developed that would allow for creating models (ensembles) of hybrid AI methods. This meta-model would be implemented into a modular distributed orchestration platform which would be further enriched with a number of modules to be tested in four gaming related environments: (1) MMORPG games, (2) gamified learning platform, (3) serious game related to autonomous vehicles and (4) a game for a holographic/volumetric gaming console which would also be developed during the project.

1.2 Introduction

The application of HAI which can be defined as the orchestration of heterogeneous artificial intelligence (AI) methods including both statistical and symbolic approaches in various domains is omnipresent in current scientific literature. It is largely overlapping with the term hybrid intelligence (HI) that has been defined as *"the combination of complementary heterogeneous intelligences (...) to create a socio-technological ensemble that is able to overcome the current limitations of (artificial) intelligence."* [20]. HI lies at the intersection of human, collective and

artificial intelligence, with the intent of taking the best of each.

There have been numerous studies recently addressing issues related to HAI and HI methods in a multitude of application domains including but not limited to land-slide prediction [38], drug testing [13], forecasting crude oil prices [64], prediction of wildfire [31], evaluation of slope stability [36], modeling of hydro-power dam [9], wind energy resource analysis [24], industry 4.0 and production automation [4], airblast prediction [3], heart disease diagnosis [39] and these are just a few references from 2018 until the time of writing this proposal. Most of these and such studies report building HAI systems by combining various AI methods to acquire better and more precise results. However, when it comes to methodology of the actual orchestration of HAI methods the usual approach is ad-hoc and depends from project to project. The lack of methodology in orchestrating HAI shall be addressed in the proposed project.

In a previous project sponsored by the Croatian Science Foundation (Installation Project No. HRZZ-UIP-2013-11-8537 entitled Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Playing Games - ModelMMORPG - see [53] for details) a comprehensive methodology for modelling large-scale intelligent distributed systems has been developed that includes a graphical modelling tool and code generator (described in [52] and in more detail in [42]). The implemented toolset allows for modelling complex multi-agent organizations and could be applied to numerous applications domains [49, 50]. Herein, we would like to apply and incorporate this methodology to the development of the HAI orchestration platform.

Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo™ in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games. Computer games have often been judged by the quality of their AI and praised if they used an innovative approach like the ghosts in Pacman™ which had individual personality traits (1980), Creatures™ which used neural networks for character development (1996), Black & White™ which used the belief-desire-intention (BDI) model (2000), F.E.A.R.™ which used automated planning algorithms (2005) and many others (see [69, pp. 8–15] for a very detailed overview). Artificial intelligence in games is not only used for non-playing character (NPC) or opponent implementation, but also for various other parts of games [69, pp. 151–203] including but not limited to **generation of content** (graphics including levels and maps, sound, narratives, rules and mechanics or even whole games like the Angelina game-generating system [17]), **player behaviour and experience modeling** [69, pp. 203–259], as well as **bot development and automated game testing** [69, pp. 91–151]. Due to their complex nature and endless possibilities of creative design, computer games present us with an excellent opportunity to study the orchestration of HAI in various scenarios – not only for fun and leisure but also for other domains in form of serious games and/or gamification.

In the previously mentioned ModelMMORPG project, we have already used an open source massively multi-player on-line role-playing game (MMORPG) called The Mana World (TMW) for which we have implemented a high-level interface to test intelligent agents playing the game. Additionally a number of connected game quests have been developed for various scenarios which allowed us to build an automated game testing system [51]. Herein we would like to use this interface to test orchestrated HAI methods, but also develop other testbeds for the planned platform.

Therefore, the main contribution of the proposed project shall be: (1) a comprehensive framework for the orchestration of hybrid artificial intelligence methods for computer games allowing to define models of HAI for various purposes, (2) an open source distributed cloud platform that will allow to implement such models based on existing HAI methods and connect them directly from game development platforms, (3) a set of best practices in developing HAI ensemble models tested in at least four specific testbeds described bellow.

1.3 Team Members

Markus Schatten (Principal investigator)
Head of Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Jaime Andres Rincon Arango
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Bogdan Okreša Đurić
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Carlos Carrascosa
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Damir Horvat
Department of Quantitative Methods
Faculty of Organization and Informatics,
University of Zagreb

Vicente Julian
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Tomislav Peharda
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Glenn Smith
College of Education,
University of South Florida

Igor Tomičić
Center for Forensics, Biometrics and Privacy
Faculty of Organization and Informatics,
University of Zagreb

Neven Vrček
Department of Information Systems Development
Faculty of Organization and Informatics,
University of Zagreb

2. MMO IF and Game Engine Implementation

2.1 Introduction

Computer games which have a textual user interface and in which players interact with the game using text commands are usually called IF, text adventures, gamebooks and even in some cases, if an additional graphical interface is present, visual novels [27]. Interactive fiction games consist of narrative worlds often organized in so called rooms which players can explore by using various commands. The term "room" here is quite broadly defined and can encompass any type of space an avatar can exist in, including but not limited to actual rooms, houses, caves, open spaces, vehicles, physical dimensions or even states of mind [1]. These rooms are usually connected through doors which represent the gateways between spaces. "Doors" are again very broadly defined and can include various pathways like actual doors, streets, portals, staircases, time machines etc. A lot of IF games fall into the role-playing game (RPG) genre, but there are exceptions.

On the other hand MMO games which allow for thousands and sometimes even hundreds of thousands players playing simultaneously, are an interesting contemporary phenomenon due to the possible mutual interactions between players, especially in the form of organizing their behaviour to perform certain game-related tasks [52]. Additionally MMORPGs which combine the massive multiplayer aspect with the RPG genre, provide us with an opportunity to study the possibility for integrating IF and MMO [51]. IF games in most cases fall into the RPG genre, but there are exceptions.

In this chapter we present an initial attempt on creating a game engine for MMO IF games. We base our implementation upon Inform 7[1] IF which is a declarative programming language for the development of IF based on natural language. We have developed a Python interface[2] to the Glulxe IF interpreter in order to be able to extend Inform 7 capabilities with various AI related methods. One application example of this interface was using multiagent system (MAS) inside IF games which has shown very promising characteristics on which we will build upon herein by introducing a game engine layer that allows interaction and synchronization between players inside the same narrative world.

[1] Available here: `http://inform7.com/`
[2] Available here: `https://github.com/AILab-FOI/python-glulxe.git`

The rest of this chapter is organized as follows: firstly in section 2.2 we provide an overview of related work. Afterwards in section 2.3 IF games are introduced and described in more detail. In section 2.4 we give an introduction to MMO games with a special focus on elements that can and should be part of IF games. In section 2.5 we introduce our game engine layer for MMO IF games and provide insights into implementation specific details. Then in section 2.6 we provide an example MMO IF game and discuss other possibilities of implementation. In the end in section 3.5 we draw our conclusions and give guidelines for future research.

2.2 Related Work

There are not many papers describing game engines for MMO IF, and up to our best knowledge there are none using an agent based approach in implementing an MMO IF game engine.

For example, in [32] a persistent system called True Story that has dynamically generated and contextually linked quests is explored. The implemented game tried to persuade the player to be a maker of its own story by completing quests and taking non-scripted paths trough a game, thus creating a unique user experience. The quest generation engine is based on a number of different constraints such as memories (what the player character did in the past), attributes (what an object is capable of), actions (what an object can do), layer (relationships established between objects) and proximity (area in which objects can interfere). Using these constraints, the system generates quests based on the player character's values of each constraint and state of the world. They achieve a system that can generate numerous quests that are specific for a player and its impact in the game environment. Whilst the implemented game does have elements of the RPG genre and IF, some elements like communication and interaction between players seem not to be addressed and the game cannot be considered an actual MMORPG since network play has been left for future research.

The paper [33] examines the Twine[3] platform which is and open source platform for developing interactive fiction. The author compares games made in Twine with mainstream games with focus on what emotional experience can game produce in player, and among other things, outlines the capability of Twine for implementing multi-player games. One of the first such games was Naked Shades[4], which was developed for the Ludum Dare gamejam in 2013. Sadly, the game isn't available any more, but nevertheless presents some interesting aspects of IF multiplayer games like how the narrative world is influenced by previous and on-line players.

The authors in their article [18] compare and analyze visual novels that use virtual reality technology as part of storytelling and enhanced immersion into a story. As they conclude, good graphics are needed for visual novels to be a good game for using virtual reality; otherwise, the player will not experience fully immersed in the story. While the games mentioned in the article do not have MMO elements, some important aspects of MMO IF games are outlined like multi-linear storytelling and player interaction.

2.3 Interactive Fiction

IF games are usually adventure, RPG and other narrative based games written in words in simple scenarios in which players can create stories through interacting with in game characters, items and the environment, usually by issuing defined sets of commands in a textual interface [8].

IF narrative worlds besides having "rooms" and "doors" connecting them usually describe objects or things that are placed in various rooms which can be examined and interacted with. Such object can for example be:

[3] Available here `https://twinery.org/`

[4] Description available here `http://ludumdare.com/compo/ludum-dare-26/?action=preview&uid=23541`

- NPCs that the player can communicate with,
- containers that might have other objects within,
- edibles that can be consumed,
- wearables that can be used as clothes or equipment, etc.

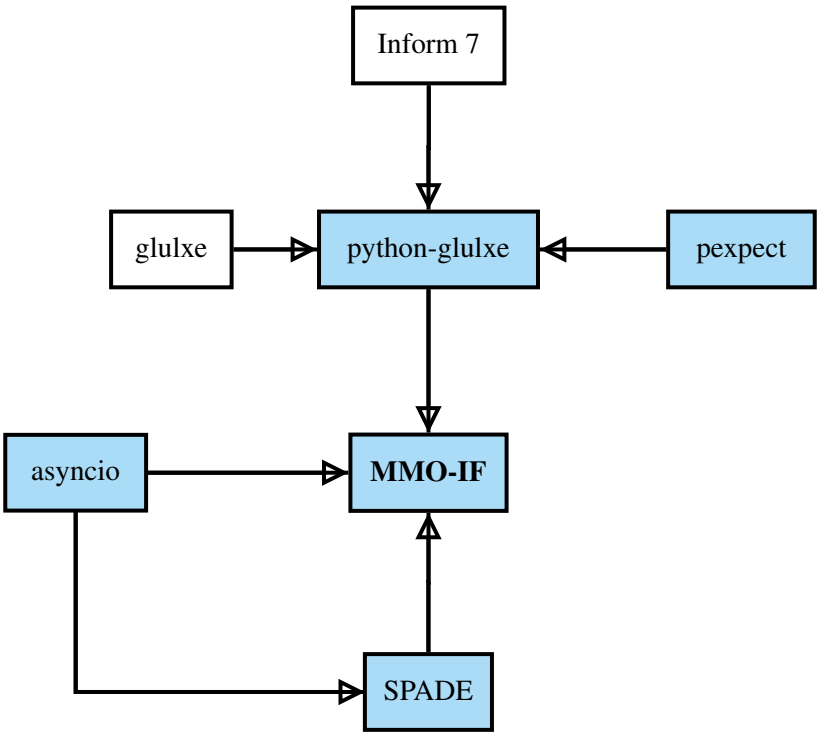


Figure 2.1: Dependency graph of the implemented game engine layer (blue: Python modules; white: external tools)

As opposed to most computer games focused on graphics, IF is focused on the story and narrative which makes it an interesting and different medium similarly as printed novels differ from movies. While some authors consider it a literary genre [72] other argue that due to the users interaction and other game elements IF works are mostly computer games [29].

2.4 Massively Multiplayer On-line Role Playing Games

Role-playing video or computer games (commonly referred to as only role-playing games or RPGs) are a game genre in which the player controls the actions of some protagonist (or potentially several party members) in a world which is well defined [15]. A massively multi-player on-line game (MMOG) is a (computer) game that supports a great number of players playing on-line simultaneously causing or even fostering interaction among them [45]. Massively multi-player on-line role playing games are thus a mixture of these two genres allowing players to control the action of their protagonist (avatar) by interacting with a potentially large user-base on-line [6].

The global market for MMO games is growing rapidly with an estimate of 21.94 Bn € in 2025 [59]. While the economic importance of MMORPGs is obvious, another aspect is of equal importance: it allows us to investigate two aspects of large-scale computing - (1) social interaction of (large numbers of) players through a computing platform as well as (2) the design and implementation large-scale distributed artificial intelligence (in form of non-player characters –

NPCs, mobs – various monsters to be fought, as well as AI players – bots).

2.5 MMO-IF

In the following we shall layout our implementation of the MMO IF game engine layer. The two main characteristics of MMO that the proposed implementation attempts to achieve are enabling communication between players and notifying players when another player enters the current room. Whilst these features are basic for MMO games, using them in an IF environment which is limited to a text only interface, has its particularities.

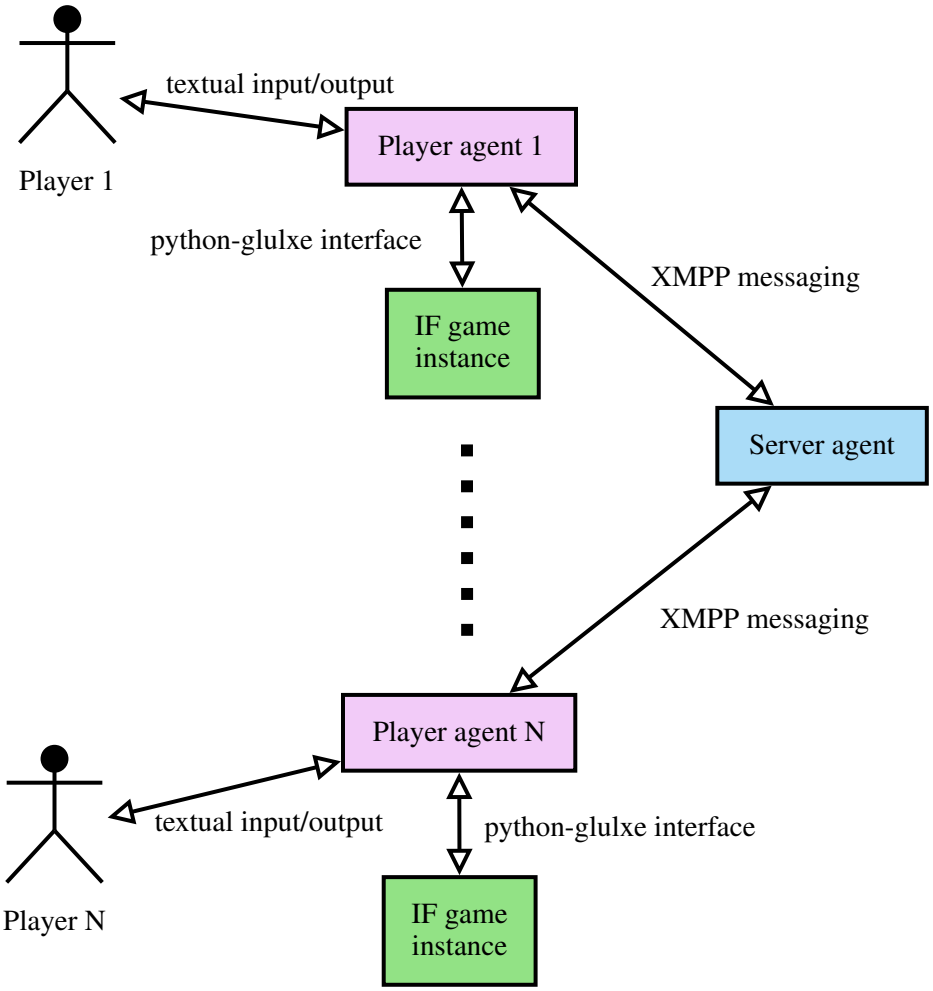


Figure 2.2: Architecture of interconnected game instances

The communication between players refers to the ability to send a message either privately, to a single player or broadcast to everyone. To make the experience of communication feel more natural, a restriction has been put in place that allows players to send their messages only to players that are located in the same room as the player. To start any type of communication, it is required from a player to execute it by typing in a special command (in our case the command has to start with the `@` symbol). The format is `@player message`, where `player` must be a valid player username to send the message directly, or `everyone` to send it to everyone. Obviously, `message` refers to a string (which might include white spaces) that is to be previewed to the receiver. On the receiving

side the agents connected to the MMO server have to monitor communication and react in case a message is sent to the player in question.

The second implemented MMO characteristic relates to notifying a player when another player enters the current room. Contrary to the communication which must be executed by players themselves, notifications about players entering the room are dispatched automatically, every time a player moves between rooms.

We have used a number of technologies to build the described layer[5]. The primary technologies could be narrowed down to Inform 7, Python, and the Extensible Messaging and Presence Protocol (XMPP) through the Smart Python Agent Development Environment (SPADE)[6] framework [2, 44]. As the idea is to develop the game engine layer on top of Inform 7, a developer is not required to add any additional code to its IF game, but only to set up a room structure (which is a usual requirement for IF games). Figure 2.1 shows the various dependencies. The layer is implemented in form of a MAS [43] in which individual players are represented as agents (game avatars) which are aware of their local environment whilst to anticipate the actions of other players in the game they communicate with a server agents that synchronizes the global state of the game.

To process input or output to the IF game in order to accommodate stated MMO characteristics, a Python interface has been put in place serving as a layer between a player and the game. By intercepting the player's input, the interface is used to check if the input is in a format which would indicate that a player is attempting to dispatch a message. On the other hand, by intercepting the game output, the role of the interface is to detect whether a player has changed rooms. On each room change, the game returns an output saying `You entered [name] room` where `name` refers to the name of the room. That way, by parsing the output, the interface keeps track of the player's location.

The XMPP facilitates the communication between players. This is achieved by having a centralized server agent that is aware of the active players and their locations. The two types of the XMPP messages that the server reacts to are when a player changes the location, and when a player wants to send a message. The location change type of the XMPP message is significant as it is involved in the restriction logic that defines who will be the receivers of a message that the server dispatches. Furthermore, when the server receives this type of message, it sends a notification to all players located in the same room as the sender, indicating what player changed its location. The other type of the XMPP message relates to ability to exchange messages between players. On the server end, this includes logic to determine who all shall be receivers of the message. Eventually, the player agent accepts the XMPP messages that the server dispatches, and shows them accordingly in each player's interface.

Figure 2.2 gives a visual description of how the described system components are connected, as well as how the communication is established.

2.6 Implementation Example

Below is a basic IF game developed in Inform 7 to support our MMO implementation.

```
"RoOms" by Tomislav Peharda
Release along with an interpreter.
Release along with a website.

When play begins:
       say "You got into the roOm! Try communicating with the other people
          ↪ in here with @"
```

[5]The full code is vailable here: `https://github.com/AILab-FOI/MMO-IF`

[6]Available here: `https://spade-mas.readthedocs.io`

```
The yellow is a room. The description is "You entered the yellow room.".
The blue is a room. It is east of the yellow. The description is "You
    ↪ entered the blue room.".
The green is a room. It is west of the yellow. The description is "You
    ↪ entered the green room.".
The red is a room. It is south of the yellow. The description is "You
    ↪ entered the the red room.".
The orange is a room. It is north of the yellow. The description is "You
    ↪ entered the orange room.".
```

When the IF game is started, a player is positioned in one of the available rooms. In our example, there are five rooms that a player can move between and all of them are named by colors. By default, a player starts in the yellow room. In each direction, there is a single room available as shown on figure 2.3.

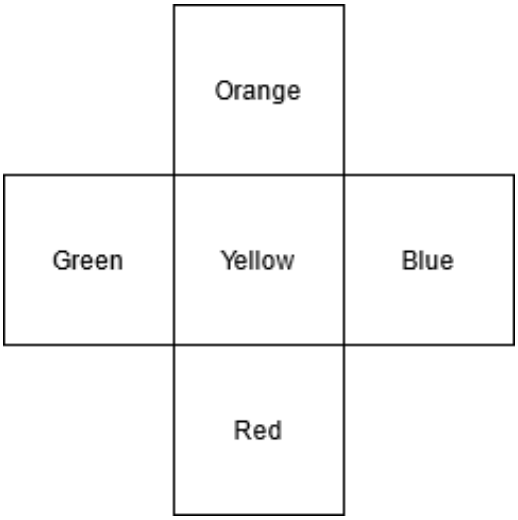


Figure 2.3: IF example game room structure

The proposed MMO-IF implementation can work along with any Inform 7 game without any adjustments needed. The only requirement that the game must fulfil is that the output format indicating room change also remains unchanged (example: "You entered the yellow room").

In this example we have chosen to showcase three players, but an arbitrary number of players could join the game having the adequate game interface. Below are three blocks showing the output of each player. The players and their corresponding output are in this order tpeharda_agent_alpha, tpeharda_agent_beta, and tpeharda_agent_omega. The output previews both the use-cases, communication between players, and notifications when a player enters a room.

```
# tpeharda_agent_alpha
You entered the yellow room.

tpeharda_agent_beta has entered the yellow room.
tpeharda_agent_omega has entered the yellow room.

> go west
You entered the green room.
```

```
tpeharda_agent_beta has entered the green room.
tpeharda_agent_omega has entered the green room.

> @everyone hey everyone!

> @tpeharda_agent_beta hey beta!
```

```
# tpeharda_agent_beta
You entered the yellow room.

tpeharda_agent_omega has entered yellow room.

> go west
You entered the green room.

tpeharda_agent_omega has entered the green room.

tpeharda_agent_alpha said: hey everyone!
tpeharda_agent_alpha said: hey beta!
```

```
# tpeharda_agent_omega
You entered the yellow room.

> go west
You entered the green room.

tpeharda_agent_alpha said: hey everyone!
```

Player tpeharda_agent_alpha has joined the game before everyone else hence it is notified when tpeharda_agent_beta and tpeharda_agent_omega joined the same room. The player in question also dispatches two messages, one directed to everyone with content "hey everyone!", and another one directed precisely to tpeharda_agent_beta with content "hey beta!". Thus, the other two players receive a message depending if they are the specified receiver.

2.7 Conclusion

In this chapter we have provided a novel game engine layer that introduces MMO to IF games using a MASs approach. It can be used in any existing IF game developed in Inform 7 without any (or sporadically minor) adjustments. We have provided an initial implementation of a MMO IF game to test the implemented layer by using the Inform 7 IF programming language, Python to create an interface and particularly SPADE for the implementation of agents. Whilst the proof-of-concept game is simplistic, it shows a promising social interaction and user experience not usually present in IF games. The MASs approach taken herein allows us to implement numerous AI techniques in modelling player interaction as outlined in [47].

In the current implementation only communication and player presence are implemented, but there are many other important features that should be addressed mostly related to synchronization of the game state, items and in game characters. Such and similar features will be subject to our future research.

3. MMORPG bot detection

3.1 Introduction

For quite some time, computer games have been part of everyday life for a significant number of people, and according to a news report published by DFC Intelligence, there were more than 3 billion gamers by the 2020 [21]. Nowadays, almost every computer game provide some sort of online component; either as comprehensive as playing with other players completely online, or as simple as providing a scoreboard containing results from other players [66].

Considering that games require an input - as in, a user action - and process it in order to provide a reaction to it in terms of an output, they are prone to exploits [16]. Exploiting a computer game is more often referred to as cheating [65] and indicates an act where a player obtains some form of an unfair advantage against the other legitimate players. In competitive games, where the goal may potentially be to obtain score points, cheating would refer to an illegal action that player performs to get better points. Some games may incorrectly implement functionalities that cunning players leverage in their favour. The described type of oversight falls into a group of bugs and glitches [63]. The other type, where a player influences the game-play directly thus obtaining advantage requires application of different techniques that often include advanced computer science skills.

The more complex a computer game is, the higher number of vulnerabilities it might have. In a simple 2D snake game [68], the most influencing exploit would be related to score points. Contrary, in a 3D multi-player first-person shooter (FPS) game [56], a player has much more functionalities to exploit. That could relate to items a player brings, geo-location, behaviour towards other players and so on. Besides for numerous game functionalities that may be exploited, there are also several ways to develop cheats [65]. Some cheats directly affect player capabilities that could, for example, result in enabling a player to see locations of opponent players that otherwise shall not be visible. Different kind of cheats for the multi-player type of games are focused on malforming network requests that are dispatched to the opponent players. That way, a player hypothetically misinforms its opponents about actions it performs. In MMORPG [57] there is increasing use of bots [58] that also fall into group of cheating. Bots are primarily used to do simple repeating tasks on behalf of a human player in order to gain some form of an advantage. As [34] argue, "game bots destroy the game balance and consume game contents fast. They cause honest users to feel deprived, lose

interest and eventually leave the game." Consequentially, this leads to the decrease of the profit in the gaming industry. Limelight [60] argues that 57% of gamers will not continue to make purchases or play games on a website that has previously suffered a security breach.

Clearly, computer game producers do not want their games to be exploited, as that could negatively impact them from different aspects [14]. Therefore, a lot of effort has been put into securing games as much as possible. In computer games themselves, game producers have been adding advanced mechanisms to detect if player in any way attempts to change the game-play. In order to prevent malmorfed network requests, additional focus has been put towards better communication encryption and advanced identification/authentication techniques [71]. The stated improvements are dealing directly with hardening the game security. However, a more human-like type of cheating includes development of bots which are harder to catch as they are developed to work within the game boundaries, without exploiting them. Therefore, different behaviour analysis mechanisms [58] are put in place to differentiate a human player from a bot.

Because the modern research is focused on developing bots that seem to be more human in their behaviours, thus deceiving the passive detection methods as will be argued in the Related Work section, we propose a method of an active bot detection in the form of an automated Turing test, where our security bot approaches a player within the game and tries to communicate with it. The answers of the approached player ("the suspect") would be evaluated, and based on the given answers (or the lack of the same), our security bot would assess if the "suspect" is a human player, or an artificial one.

3.2 Related Work

As stated previously, cheating in computer games is a large concern for any type of audience [14]. Different parties attempt to take various strategies in order to prevent cheating. S. Ferretti and M. Roccetti [22] focus on cheating detection of malformed actions in peer-to-peer games. Essentially, what happens is that players perform actions that do not complete immediately, but instead take time to resolve. An opponent is informed about the execution of an action including time it took to perform it. The problem here is that any player may malform an action data that is sent to their opponent and fake execution times, therefore, influencing the game-play. What authors propose in order to detect such cheating is to measure the execution time of each available action in the game, and store it on each players' end. That way, when players are engaged in a match, an automated mechanism built in each players' game engine evaluates the opponent action execution times to detect potential cheating.

C. Zhao in [71] speaks about a similar problem, but on a more general level. In the article, Zhao emphasises on the development of cheats that exploit the game and trojans that can be injected to peers over network. Considering the both listed type of cheating is performed online, the solutions that are proposed for avoiding the associated risks are based on implementing proper encryption systems such as the RSA algorithm and/or enhanced identification/authentication mechanisms so that players could not easily spoof their identity, as well as on higher performance servers and bandwidth, in order to prevent any unplanned server downtimes, that could open up more space for additional harm in sense of leveraging network instability.

In [58], authors deal with a type of cheating in MMORPG that introduced bots that play a game on behalf of a player. Essentially, the bots are designed to do simple repeating tasks, such as battling NPCs or trading possessed items to gain further advantage. Authors suggests that by the analysis of bots behaviour, it is possible to detect them. In particular, frequencies of the stated actions should distinguish human players from bots, as bots tend to perform certain actions much faster.

Python-based bots that can play a game instead of the human player, by leveraging an unsecured network protocol, is presented in detail in [61]. These bots use a low-level game interface to connect

to the game and control a game character through specifically crafted network packets, and a higher-level interface for basic reasoning.

A group of authors [35] examine how game logs can potentially be used to track down bots in a MMORPGs game. Their assumption is that when players and bots engage in a party play, each type has different goals. A player engages in a party play to complete quests that are otherwise harder to complete for a single player, while behaviour of bots is geared towards items acquisition. That also implies that party plays last much longer (even indefinitely) when bots are playing, which is one of the main indications of bots involvement. Authors propose that the analysis of party play logs that contain information about game events, especially repetitive player actions, could unveil bots.

Similarly, in [37] research is also geared on unveiling bots in MMORPGs based on the log analysis. Concrete technique they vouch for is the analysis of player actions from a game log as a function of the time lag. On a low level, that includes comparison of a human player actions against bot actions to construct a model with capabilities to detect a bot behaviour.

Depending on a game genre, there are some key actions and characteristics that can be looked for to distinguish a bot from a human player. In 2008 and 2009 there was a BotPrize competition [30] organised with a goal to develop a human-like bot. Bots were developed for a FPS game called Unreal Tournament 4. Judges, whose role was to give their verdict whether a player in a game was a bot or a human, would analyse behaviour and game-play. Behaviour of bots were not so convincing, therefore, judges were able to properly classify players. However, based on comments from judges, what uncovered bots were lack of capabilities to plan their actions as well as inconsistencies in their intentions, static movements, and aggressive behaviour. As for lack of planning capabilities, bots would engage into one action and before completing it, move onto another one. Any advanced planning, that would include applying strategy to combat opponents was fully missing. In terms of static movements, bots would show signs of not having smooth movements, but rather stiff. Not only that, at times they would get stuck and were unable to proceed further. Aggressive behaviour is another indicator that makes it easier to detect bots. More specifically, bots would shoot a lot, and be very accurate at it. The speed and accuracy can be remedied trivially however; as [30] argue, "bot that shoots too quickly and accurately is easily identified as non-human, but it is simple to slow the bot down and make its shots less accurate".

To identify bots, they might be even tested in specifically crafted scenarios. For example, putting a bot in a location where there are plenty of obstacles would much faster reveal it than putting it in a location with lots of open space. On a similar note, having a bot battle multiple opponents at the same time may potentially result in bot resolving the situation in its favour with greater success comparing to how a human player would perform [30]. Thus to minimise time required to identify bots, scenarios where bots tend to show weaknesses could be designed with indications on bot reactions to given scenario.

The primarily approach to detect a bot is by behaviour analysis [58]. Bots can be seen as regular players with a concrete goal set. Often times, they are developed in such a way that they do not hold superior capabilities to regular players. Identifying bots depends on a context. In MMORPGs some bot types are defined to do repeating tasks indefinitely [58] thus to identify them, key indicators to look for would be how long is a bot engaged in a game (if information is provided) and what sort of tasks it executes. If a game is taking longer than some chosen time reference point, that could be a relevant indicator. If a bot does a singular action continuously, such as trading items, that also might be a relevant indicator. Game events and player actions are usually stored in game logs, therefore, analysis of game logs can be used to identify them [37]. One of the possible ways to analyse game state changes is by applying machine learning algorithms, such as Bayesian network approach [70].

A similar approach involves analysis of the network traffic [10]. A player pragmatically informs either a server, or its peers about actions it performs. When examining network traffic, the main focus should be on the execution time and the traffic magnitude. What that means is that a bot

may potentially inform server or its peers about completion of an action by malmorfing a network request and modification of execution times [22] so that the execution time takes shorter time than it originally does.

Besides action execution times and repetitive actions, another aspect that could be analysed is player movement [41]. Movement is also communicated over different channels either as a network traffic, in logs or shared exclusively with the server. By extracting waypoints, it is possible to recognise paths that are repetitive. If a bot always takes the same path between concrete outset location and a destination, it may indicate that it has calculated the path as the optimal one.

With all the reviewed research body in the domain of bot detection in mind, an automatic bot detection methods where an artificial system may differentiate between the human and the artificial player still seems to be in its relative infancy. The use of CAPTCHA tests are considered in [26], but this brings along the unnatural disruption of the game flow. Others, as mentioned, perform an analysis of behaviour patterns, such as patterns of movement ([11], [12]), but as previously argued, behaviours can be adapted to slip through such techniques [30], [55], [54]. Moreover, this year for the first time on the BotPrize competition, two bots achieved humanness ratings of over 50%, whereas the human players achieved average humanness ratings of just 40% in the form of a Turing test which is conducted within this competition [7].

In light of all this bot behavioural advancements towards bot humanness, we propose a method of an active bot detection in the form of an automated Turing test, where our artificial player (security bot) approaches the player and tries to communicate with the player. The answers of the player would be evaluated, and based on the given answers (or the lack of them) our security bot would assess if the "suspect" is a human player, or an artificial one.

3.2.1 Types of Bots

Capabilities and complexity of bots depend on their purpose which also indicates how challenging is it to develop them. In this context, bots can be classified in groups based on their level of advancement. A bot that performs a single repetitive task, such as, writing a message in the chat can be completely agnostic to its environment, as it requires no input. To develop this type of a bot, there is no reasoning skills to be involved, therefore, development shall be easy [58].

Bots that require to process the input in order to properly react to an event require reasoning capabilities to some degree. If they are developed to perform a simple action, such as battling NPCs or trading items, in order to make an advantage in favour of a player, they need to be able to recognise patterns that lead them to it. Usually this mean comparison conditions that evaluate player state and input attributes [35].

A higher degree of reasoning capabilities involve machine learning algorithms [70]. That could either be utilised to develop complete behaviour of a bot so that it may participate in all actions provided in a game or to enhance capabilities of performing a single task. One of the possible approaches to develop behaviour of a bot is by application of Bayesian network. The approach includes programmatic analysis of attributes that describe different states of a player. The outcome of the process is a model that is capable to react to state changes in the real-time. In simpler games, where there is a limited number of actions available, a bot could be specialised to execute only some of the actions rather than all of them. For example, in Pictionary [67], where the main tasks consist of properly drawing a suggested item and for opponents to guess them, various computer vision tools may be facilitated to enhance bots capabilities [5].

Previous types of bots are primarily developed to gain advantage on behalf of a player, however, they do not put much significance on how easy or hard is it to detect them [30]. Human-like bots do not only intend to be precise in their actions, but the goal of this type of bots is also to replicate human behaviour as much as possible, so that opponents and detection mechanisms have hard time detecting them [55]. In an FPS game that was discussed previously, that implies smooth

movements, planning capabilities and environment awareness. To develop a bot with such degree of advancement, it is likely that sole application of machine learning algorithms may not be sufficient thus requiring manual adjustments of bot behaviour.

In this chapter, we have assumed "the worst case" scenario, where a bot is advanced to the level that it can imitate human behaviour - both in playing the game and in conversational capabilities - based on the currently available research and their limitations.

3.3 Requirements

The model proposed within this chapter relies on a game API through which we could infiltrate our security bot into the game, and on the game chatting platforms, through which our bot can communicate with other players. The game API on which this model partly relies is based on the work presented in [46], which currently supports one MMORPG game (The Mana World). In order to be able to distinguish the legit human player from an artificial one, our security bot needs to be equipped with the ability to perform a form of a Turing test to a suspect player. Because of this, we tried to identify types of questions that a more advanced AI bot playing the game would fail to answer correctly. In most cases, bots are designed to perform specific actions, and rarely are equipped with advanced chat capabilities; such bots are programmed to do only a few certain things, and can easily be detected by starting a simple, general-topic conversations with them. For example, a security bot can ask a simple questions like "what is the colour of the clear sky?", or "give me any number that is not the result of adding two plus two", and even a simple response to the "Hello" can yield results if there is no answer, as the real players are obliged to answer to the security bot. In this work, we anticipate the "worst case scenario", i.e. a bot that can imitate human chat responses.

A group of authors have summarised passive and active bot detection methods, and more significantly for our work, strategies for actively interrogating suspected chatbots [40].

As for the passive detection, authors analyse message size and inter-message delays in order to successfully distinguish humans from bots. According to [25], in contrast to most bots, human inter-message delays (times between sequential message transmissions) “appeared to follow a distinct power law distribution, and human message sizes seemed to follow an exponential distribution (with $\lambda = 0.034$)”. As for the active detection, authors list possible strategies detailed in several other papers. We have initially dismissed a few of them (questions with very informative answers, keyword targeting, evasiveness, using rating games, using ambiguous questions) as such tactics either do not seem feasible for the implementation of our security bot because of the automated nature of the proposed interrogation method, whereas these tactics would require a more complex answer analysis and reasoning, or they might be too intrusive for the game play, breaking the game flow and irritating human players.

Other listed tactics proved to be more useful for our security bot, such as:

- Challenging the syntactic engine, which include questions based on elementary logic (for example: “if New York is north of Atlanta, is Atlanta south of New York?”), common typing shortcuts (for example: “do u like to go 2 dinner b4 going to c a movie?”), using figures of speech or slang, and requesting enumerations to simple questions (for example: “name three things you can do with a ball”).
- Using so called “URL questions” (Understanding, Reasoning, and Learning), where the idea is to probe basic human intelligence. Examples are listed as follows.
  UNDERSTANDING: “What shape is a door?”; “What happens to an ice cube in a hot drink?”
  REASONING: “Altogether, how many feet do four cats have?”; “What does the letter M look like upside down?”;
  LEARNING: “What comes next after A1, B2, C3, . . . ?”; “PLEASE IMITATE MY TYPING STYLE!!!!”

The bots would answer these questions nonsensically, evade or ignore the questions according to [23].

- Using sub-cognitive questions, which rely on the “internal, physical, and/or personal-historical experiences of human beings”, which computer lacks. In [19], authors suggest questions aimed at physical structure and not on the higher-level cognition, but on the low-level sensations and perceptions, for example: “What happens to your clothes if you fall into a pool?”; “If you touch a hot pan, what does it feel like?”
- Using a guessing game based on internal human experiences. For example: “Would you like to play a game? Then try to guess what I‘m thinking of. . . " We than provide a series of hints, such as: "It digests food"; "It sometimes aches"; "Most people cannot pat their head and rub this at the same time” (the answer is stomach). According to [19], bots should fail a Turing Test based on such questions, because again the lack of common human experiences.
- Using a guessing game on general and common sense knowledge. For example, the "suspect" would have to guess what is being hinted at: "You play this game with a black and white ball, your feet, 2 nets, and 11 players on each team" (the answer is soccer). Or "You use this to talk to people, you hold it in your hand, and you dial numbers on it" (the answer is telephone).
- Using emotional-based questions. For example: “How would you feel if you won the lottery”; “Can you describe how you would feel if you were fired from your job for no obvious reason?”
- Using intentional misspellings. For example: "Doo yuo knowe whut thyme it iz?"; "ha+ is y0re 8irthday?"; "Whhat iss yerr favvorite memmorie?" "Can you raed these wrods taht I‘ve tyepd?"

Following these tactics we can extract a significant set of questions that our security bot can use, and most of them could have a relatively simple answers which could be verified by simple if-then logic and regular expressions with practically no need for more advanced reasoning, making the automated Turing test more feasible.

3.4 A Conceptual Model

The proposed model relies on the work presented within the [46], implementing so called lower and higher-level game interfaces for playing the game with artificial players.

The "lower-level interface" presented in the chapter enables the emulation of a legitimate game client connecting to the server and playing the game. It does this by re-creating and manipulating network packets being sent between the game server and the client, where the game server could not differentiate between the packets sent by the Python script from those sent by the legitimate game client [61].

The idea is to imitate the actions of the real human player. By following this idea further, the artificial security bots could be infiltrated within computer games through such interfaces. The reasoning and autonomy of such an security bot could be implemented within the "higher-level interface" [46], which builds upon the lower-level by introducing an agent template implemented in SPADE [28]. Authors have implemented a STRIPS-based planning system in Prolog, and an agent knowledge base implemented using the SPADE knowledge-base system for SWI Prolog, which could be refitted for different use-cases - for example, for new types of agents that may require a different AI method, such as machine learning. There is a constant interaction between the two interfaces; low-level interface is providing higher-level agent with actionable behaviours such as navigation, NPC conversation handling, fight handling, party management, etc. The agent autonomy is based on the belief-desire-intention (BDI) model, where the agent initially senses the environment, updates it’s knowledge base, chooses a goal to accomplish, generates a plan for this particular goal and then starts executing it.

Since the focus of our proposed security bot is on other players using potentially illegal activities, the sensing part of the BDI would include observing the actions of other players, and upon detecting

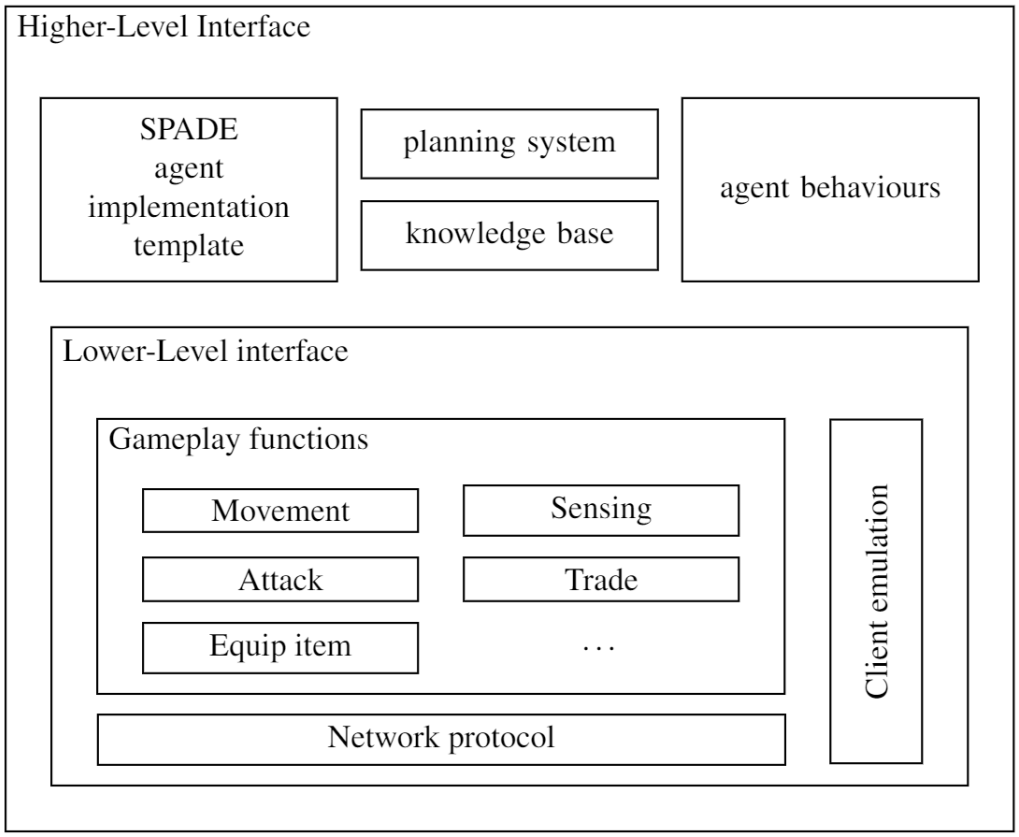


Figure 3.1: A possible architecture for an agent-based security bot model [46]

the potential illegal action, the bot would set the goal to locate the player in question, approach it, and test it for being either human player or artificial one in some form of the reverse Turing test. In order for this to work, the security bot would need more detailed insight into the in-game data - other player locations, actions, statistics, and other metadata, depending on the game itself, in order to passively identify suspicious behaviours. This part may be implemented with any of the aforementioned methods for detecting game bots. On the other hand, the security bot may also randomly pick players and "interrogate" them. The approached "suspect" must be able to identify the security bot as such, because every player would be obliged to respond to the security bot, otherwise risk being kicked out of the game.

Should the security bot identify the suspect as the game cheating bot, it could report it to the game administrator, or autonomously ban the player from the game. Should the suspect pass the test and prove itself to be human, the security bot departs and goes back to the loop of wandering, sensing and detecting illegal activities, and randomly approaching players.

The game administrators would have a simple interface for upgrading their security bots with new questions and tests in order to have the possibility to be ahead of the game bot designers which could in time download all the challenges and hard code the answers into their bots.

3.5 Conclusion

The existing body of research is abundant with passive bot detection tactics, analysing game logs, network traffic, bot behaviours and patterns, action execution times, message size and inter-message delays in communication, and other relevant indicators, but rarely the literature answers the question: what if more advanced, human-like cheating bot passes through all these mechanisms and remains

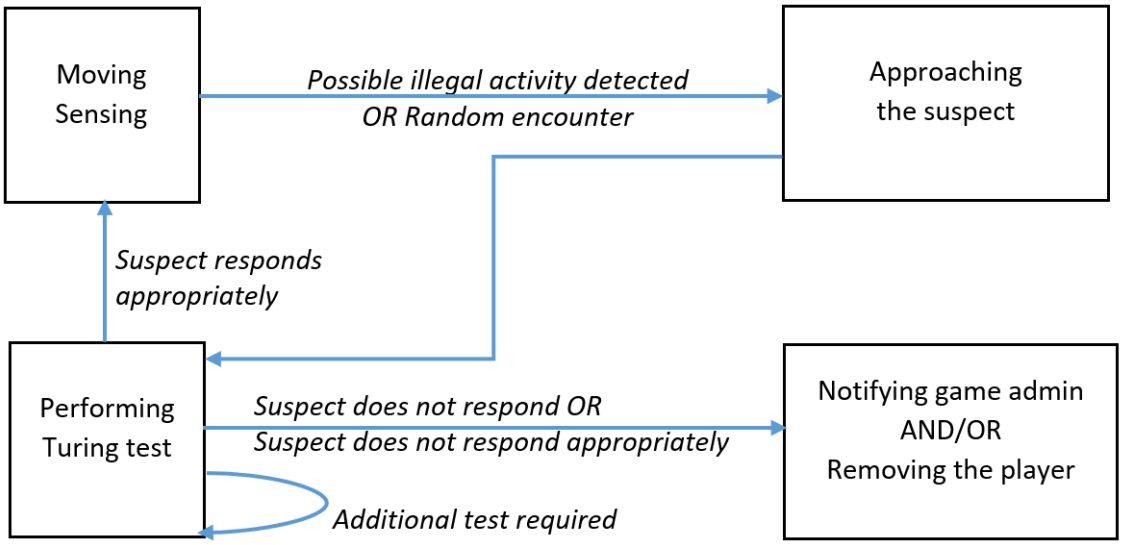


Figure 3.2: A high level concept of the security bot model

within the game undetected? The security bot model proposed within this chapter is an active in-game mechanism that would aim to detect and prevent illegal artificial players or scripts from playing the game, should those get by other security mechanisms. It would use a pool of carefully designed questions to conduct a form of an automated Turing test on players with a sole purpose of distinguishing a legit human player from an artificial one. Depending on its designed autonomy, a security bot might report an illegal player, or even permanently remove it from the game. The research so far includes the conceptual model of such a mechanism, with feasible implementation methods through previously developed MMORPG game interfaces which would enable the security bot to login to the game and play it as a regular player, tactics to be included in the development of the question database, and a higher-level interface which would enable the security bot with basic reasoning and planning capabilities. Further research will include the full implementation of such a bot and proof of concept on a single computer game, with the aim of detecting other bots within the game and measuring its efficiency comparing with the existing passive detection methods that were previously implemented.

Bibliography

[1] Jim Aikin. “The Inform 7 Handbook”. In: (2009) (cited on page 5).

[2] Estefania Argente et al. “Supporting agent organizations”. In: *International Central and Eastern European Conference on Multi-Agent Systems*. Springer. 2007, pages 236–245 (cited on page 9).

[3] Danial Jahed Armaghani et al. “Airblast prediction through a hybrid genetic algorithm-ANN model”. In: *Neural Computing and Applications* 29.9 (2018), pages 619–629 (cited on page 2).

[4] Aydin Azizi. “Hybrid artificial intelligence optimization technique”. In: *Applications of Artificial Intelligence Techniques in Industry 4.0*. Springer, 2019, pages 27–47 (cited on page 2).

[5] Giorgia Baroffio, Luca Galli, and Piero Fraternali. “Designing bots in games with a purpose”. In: *IEEE Symposium on Computational Intelligence and Games*. Volume 2015. IEEE, 2015 (cited on page 16).

[6] Roberta Biolcati, Virginia Pupi, and Giacomo Mancini. “Massively Multiplayer Online Role-Playing Game (MMORPG) Player Profiles: Exploring Player’s Motives Predicting Internet Addiction Disorder”. In: *International Journal of High Risk Behaviors and Addiction* 10.1 (2021) (cited on page 7).

[7] BotPrize authors. *BotPrize Humanness results*. [Online; accessed 13-June-2021]. 2021. URL: `https://botprize.org/results/` (cited on page 16).

[8] Mary Ann Buckles. “Interactive fiction: The computer storygame" Adventure"”. PhD thesis. University of California, San Diego, 1985 (cited on page 6).

[9] Kien-Trinh Thi Bui et al. “A novel hybrid artificial intelligent approach based on neural fuzzy inference model and particle swarm optimization for horizontal displacement modeling of hydropower dam”. In: *Neural Computing and Applications* 29.12 (2018), pages 1495–1506 (cited on page 2).

[10] K.-T. Chen et al. “Identifying MMORPG Bots: A Traffic Analysis Approach”. In: *EURASIP Journal on Advances in Signal Processing volume* (2009) (cited on page 15).

[11] Kuan-Ta Chen and Li-Wen Hong. “User identification based on game-play activity patterns”. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*. 2007, pages 7–12 (cited on page 16).

[12] Kuan-Ta Chen et al. “Identifying MMORPG bots: A traffic analysis approach”. In: *EURASIP Journal on Advances in Signal Processing* 2009 (2008), pages 1–22 (cited on page 16).

[13] Wei Chen et al. “Novel hybrid artificial intelligence approach of bivariate statistical-methods-based kernel logistic regression classifier for landslide susceptibility modeling”. In: *Bulletin of Engineering Geology and the Environment* (2018), pages 1–23 (cited on page 2).

[14] Y.-C. Chen et al. “Online gaming cheating and security issue”. In: *International Conference on Information Technology: Coding and Computing (ITCC”05) - Volume II*. 2007, pages 518–523 (cited on page 14).

[15] Sawyer Collins and Selma Šabanović. “" What Does Your Robot Do?" A Tabletop Role-Playing Game to Support Robot Design”. In: *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE. 2021, pages 1097–1102 (cited on page 7).

[16] Benjamin D. Cone et al. “A video game for cyber security training and awareness”. In: *Computers & Security* (2007), pages 63–72 (cited on page 13).

[17] Michael Cook, Simon Colton, and Azalea Raad. “Inferring Design Constraints From Game Ruleset Analysis”. In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2018, pages 1–8 (cited on page 2).

[18] Rebecca Crawford and Yuanyuan Chen. “From hypertext to hyperdimension Neptunia: The future of VR visual novels: The potentials of new technologies for branching-path narrative games”. In: *2017 23rd International Conference on Virtual System Multimedia (VSMM)*. 2017, pages 1–7. DOI: 10.1109/VSMM.2017.8346298 (cited on page 6).

[19] Jamie Cullen. “Imitation versus communication: Testing for human-like intelligence”. In: *Minds and Machines* 19.2 (2009), pages 237–254 (cited on page 18).

[20] Dominik Dellermann et al. “Hybrid intelligence”. In: *Business & Information Systems Engineering* (2019), pages 1–7 (cited on page 1).

[21] DFC Intelligence authors. *Global Video Game Consumer Segmentation*. [Online; accessed 10-June-2021]. 2021. URL: https://www.dfcint.com/product/video-game-consumer-segmentation-2/ (cited on page 13).

[22] Stefano Ferretti and Marco Roccetti. “Game Time Modelling for Cheating Detection in P2PMOGs: a Case Study with a Fast Rate Cheat”. In: *The 5th Workshop on Network & System SUpport for Games 2006 - NETGAMES 2006* (2006) (cited on pages 14, 16).

[23] RM French. “Subcognitive probing: Hard questions for the Turing Test”. In: *Proceedings of the Tenth Annual Cognitive Science Society Conference*. 1988, pages 361–367 (cited on page 18).

[24] Tonglin Fu and Chen Wang. “A hybrid wind speed forecasting method and wind energy resource analysis based on a swarm intelligence optimization algorithm and an artificial intelligence model”. In: *Sustainability* 10.11 (2018), page 3913 (cited on page 2).

[25] Steven Gianvecchio et al. “Measurement and Classification of Humans and Bots in Internet Chat.” In: *USENIX security symposium*. 2008, pages 155–170 (cited on page 17).

[26] Philippe Golle and Nicolas Ducheneaut. “Preventing bots from playing online games”. In: *Computers in Entertainment (CIE)* 3.3 (2005), pages 3–3 (cited on page 16).

[27] Mercedes Gómez-Albarrán et al. “Authoring and playing interactive fiction with conventional web technologies”. In: *Multimedia Tools and Applications* (2021), pages 1–43 (cited on page 5).

[28] Miguel Escrivá Gregori, Javier Palanca Cámara, and Gustavo Aranda Bada. “A jabber-based multi-agent system platform”. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 2006, pages 1282–1284 (cited on page 18).

[29] Matthew Hausknecht et al. “Interactive fiction games: A colossal adventure”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 34. 05. 2020, pages 7903–7910 (cited on page 7).

[30] Philip Hingston. “A new design for a turing test for bots”. In: *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE. 2010, pages 345–350 (cited on pages 15, 16).

[31] Abolfazl Jaafari et al. “Hybrid artificial intelligence models based on a neuro-fuzzy system and metaheuristic optimization algorithms for spatial prediction of wildfire probability”. In: *Agricultural and Forest Meteorology* 266 (2019), pages 198–207 (cited on page 2).

[32] Scott Brodie James Pita Brian Magerko. “True story: dynamically generated, contextually linked quests in persistent systems”. In: *Future Play ’07: Proceedings of the 2007 conference on Future Play* (November 2007), pages 145–151 (cited on page 6).

[33] Friedhoff Jane. “Untangling Twine: A Platform Study”. In: *DiGRA ’13 - Proceedings of the 2013 DiGRA International Conference: DeFragging Game Studies*. Aug. 2014. ISBN: ISSN 2342-9666. URL: `http://www.digra.org/wp-content/uploads/digital-library/paper_67.compressed.pdf` (cited on page 6).

[34] Ah Reum Kang, Huy Kang Kim, and Jiyoung Woo. “Chatting pattern based game BOT detection: do they talk like us?” In: *KSII Transactions on Internet and Information Systems (TIIS)* 6.11 (2012), pages 2866–2879 (cited on page 13).

[35] Reum Kang et al. “Online game bot detection based on party-play log analysis”. In: *Computers & Mathematics with Applications* (2013), pages 1384–1395 (cited on pages 15, 16).

[36] Mohammadreza Koopialipoor et al. “Applying various hybrid intelligent systems to evaluate and predict slope stability under static and dynamic conditions”. In: *Soft Computing* (2018), pages 1–17 (cited on page 2).

[37] Eunjo Lee et al. “You Are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild”. In: *The 5th Workshop on Network & System SUpport for Games 2006 - NETGAMES 2006* (2016) (cited on page 15).

[38] Mengshan Li et al. “Prediction of pKa values for neutral and basic drugs based on hybrid artificial intelligence methods”. In: *Scientific reports* 8.1 (2018), page 3991 (cited on page 2).

[39] Gunasekaran Manogaran, R Varatharajan, and MK Priyan. “Hybrid recommendation system for heart disease diagnosis based on multiple kernel learning with adaptive neuro-fuzzy inference system”. In: *Multimedia tools and applications* 77.4 (2018), pages 4379–4399 (cited on page 2).

[40] John P McIntire, Lindsey K McIntire, and Paul R Havig. “Methods for chatbot detection in distributed text-based communications”. In: *2010 International Symposium on Collaborative Technologies and Systems*. IEEE. 2010, pages 463–472 (cited on page 17).

[41] Stefan Mitterhofer et al. “Server-side Bot Detection in Massive Multiplayer Online Games”. In: *IEEE Security and Privacy Magazine* (2009) (cited on page 16).

[42] Bogdan Okreša Đurić. “Organizational Modeling of Large-Scale Multi-Agent Systems with Application to Computer Games”. PhD thesis. Faculty of Organization and Informatics, University of Zagreb, 2018 (cited on page 2).

[43] Bogdan Okreša Đurić et al. “MAMbO5: A new Ontology Approach for Modelling and Managing Intelligent Virtual Environments Based on Multi-Agent Systems”. In: *Journal of Ambient Intelligence and Humanized Computing* (2018) (cited on page 9).

[44] Javier Palanca et al. “SPADE 3: Supporting the New Generation of Multi-Agent Systems”. In: *IEEE Access* 8 (2020), pages 182537–182549 (cited on page 9).

[45] Lisa Raith et al. “Massively Multiplayer Online Games and Well-Being: A Systematic Literature Review”. In: *Frontiers in Psychology* 12 (2021), page 2369 (cited on page 7).

[46] Markus Schatten, Bogdan Okreša Đurić, and Igor Tomičić. “Towards an Application Programming Interface for Automated Testing of Artificial Intelligence Agents in Massively Multi-Player On-Line Role-Playing Games”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2018, pages 11–15 (cited on pages 17–19).

[47] Markus Schatten, Bogdan Okreša Đurić, and Tomislav Peharda. “An Agent-based Game Engine Layer for Interactive Fiction”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2021, (in print) (cited on page 11).

[48] Markus Schatten, Tomislav Peharda, and Juraj Rasonja. “A Game Engine Layer for the Implementation of Massively Multiplayer On-line Interactive Fiction”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2021, pages 11–16 (cited on page 2).

[49] Markus Schatten, Jurica Ševa, and Igor Tomičić. “A roadmap for scalable agent organizations in the internet of everything”. In: *Journal of Systems and Software* 115 (2016), pages 31–41 (cited on page 2).

[50] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “A review on application domains of large-scale multiagent systems”. In: *Central european conference on information and intelligent systems*. 2017 (cited on page 2).

[51] Markus Schatten et al. “Agents as bots–an initial attempt towards model-driven mmorpg gameplay”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 246–258 (cited on pages 2, 5).

[52] Markus Schatten et al. “Automated MMORPG Testing–An Agent-Based Approach”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 359–363 (cited on pages 2, 5).

[53] Markus Schatten et al. “Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Role-Playing Games–A Summary”. In: *Central European Conference on Information and Intelligent Systems*. 2017 (cited on page 2).

[54] Jacob Schrum, Igor V Karpov, and Risto Miikkulainen. “UT$_\lambda$ 2: Human-like behavior via neuroevolution of combat behavior and replay of human traces”. In: *2011 IEEE Conference on Computational Intelligence and Games (CIG’11)*. IEEE. 2011, pages 329–336 (cited on page 16).

[55] Bhuman Soni and Philip Hingston. “Bots trained to play like a human are more fun”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pages 363–369 (cited on page 16).

[56] technopedia contributors. *First Person Shooter (FPS)*. [Online; accessed 10-June-2021]. 2021. URL: https://www.techopedia.com/definition/241/first-person-shooter-fps (cited on page 13).

[57] technopedia contributors. *Massively Multiplayer Online Role-Playing Game (MMORPG)*. [Online; accessed 11-June-2021]. 2021. URL: https://www.techopedia.com/definition/1919/massively-multiplayer-online-role-playing-game-mmorpg (cited on page 13).

[58] R. Thawonmas, Y. Kashifuji, and K.-T Chen. “"Detection of MMORPG bots based on behaviour analysis”. In: *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACE. 2008, pages 91–94 (cited on pages 13–16).

[59] The Insight Partners. *MMO Games Market to 2025 - Global Analysis and Forecast by Genre, Platform and Revenue Model*. 2021. URL: https://www.theinsightpartners.com/reports/mmo-games-market (cited on page 7).

[60] *The State of Online Gaming - 2018*. URL: https://www.limelight.com/resources/white-paper/state-of-online-gaming-2018/. (accessed: 23.02.2019) (cited on page 14).

[61] Igor Tomičić, Petra Grd, and Markus Schatten. “Reverse engineering of the MMORPG client protocol”. In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2019, pages 1099–1104 (cited on pages 14, 18).

[62] Igor Tomičić, Tomislav Peharda, and Andrija Bernik. “An Active Game Bot Detection with Security Bots”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2021, pages 25–31 (cited on page 2).

[63] David Velasco and Daniel Delgado. *Gaming dictionary for newbies*. [Online; accessed 10-June-2021]. 2021. URL: https://www.megainteresting.com/techno/gallery/gaming-dictionary-for-newbies-771573460415/1 (cited on page 13).

[64] Minggang Wang et al. “A novel hybrid method of forecasting crude oil prices using complex network science and artificial intelligence algorithms”. In: *Applied energy* 220 (2018), pages 480–495 (cited on page 2).

[65] Wikipedia contributors. *Cheating in video games — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-June-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Cheating_in_video_games&oldid=1024175185 (cited on page 13).

[66] Wikipedia contributors. *Online game — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-June-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Online_game&oldid=1020769453 (cited on page 13).

[67] Wikipedia contributors. *Pictionary — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-June-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Pictionary&oldid=1023653082 (cited on page 16).

[68] Wikipedia contributors. *Snake (video game genre) — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-June-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Snake_(video_game_genre)&oldid=1028120569 (cited on page 13).

[69] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*. Volume 2. Springer, 2018 (cited on page 2).

[70] S. F. Yeung and John C.S. Lui. “Dynamic Bayesian approach for detecthing cheats in multi-player online games”. In: *Multimedia Systems* (2008), pages 221–236 (cited on pages 15, 16).

[71] Chen Zhao. “Cyber Security Issues in Online Games”. In: *AIP Conference Proceedings 1955*. 2018 (cited on page 14).

[72] Richard Ziegfeld. “Interactive fiction: A new literary genre?” In: *New Literary History* 20.2 (1989), pages 341–372 (cited on page 7).