Designing a Platform to Support and Improve the Geometry Friends Game AI Competition

RICARDO COSTA, Instituto Superior Técnico, Universidade de Lisboa

This paper describes the development of a new online competition platform to support the future of the Geometry Friends Game AI competition, an artificial intelligence competition based on the Geometry Friends game. Geometry Friends is a cooperative 2D platformer whose main characteristics are visited in the beginning of this work, along with how the competition has been running until now. Several other AI competition platforms are studied, as well the practices which are considered to be state of the art to run good and prosperous game based AI competitions. Several informed requirements that the new platform should meet were laid out, which included facilitating competition participation online and automating several submission handling processes. The new platform was developed mostly from scratch, consisting of a new website and a background program in charge of fully handling the received submissions, and more. Virtualization was used to create secure, fair and reusable evaluation environments. Different features of the new platform were tested by several people across three different test scenarios. The new website was concluded to have an above average usability, while the submission handling program worked as expected throughout the testing phase.

1 INTRODUCTION

Video games make one of the best testbeds for artificial intelligence. They provide the researcher or developer with inexpensive virtual simulations where algorithms can be tested repeatedly. They can also be viewed as good platforms for iterative AI testing models, with the possibility for gradual improvements over time.

Geometry Friends [1] is one of such games. Initially designed to be a 2D platform puzzle game oriented towards cooperation, it was adapted to enable the implementation of artificial agents, which can be defined as virtual entities that act upon a virtual environment. Thus, the Geometry Friends game is relevant to the field of AI because it poses a rather unique set of challenges which are not considered to be completely solved. This means that there is room for improvement, where new algorithms and ideas can be explored in the future.

A Geometry Friends AI competition [2] exists and has been receiving submissions annually since 2013. Competitions allow different parties to present and compare their solutions. Several factors make a competition more attractive, such as game quality, unique challenges and good competition platforms. A competition platform is the system composed by several components such as the interfaces which the participants and even competition organizers interact with, participant registry, solution submission and consequent processing, community interaction, etc.

The platform of the Geometry Friends Game AI competition (which will be referred to as simply GFGAI in the rest of the document) is the main focus of this work. Improving a competition platform not only facilitates participation, but also cuts costs of running and maintaining a competition. It follows that improving the Geometry Friends Game AI Competition platform could increase the number of submissions per competition edition, as well as the amount of editions that could be held in the future.

Author's address: Ricardo Costa, Instituto Superior Técnico, Universidade de Lisboa, ricardo.gomes.costa@tecnico.ulisboa.pt.

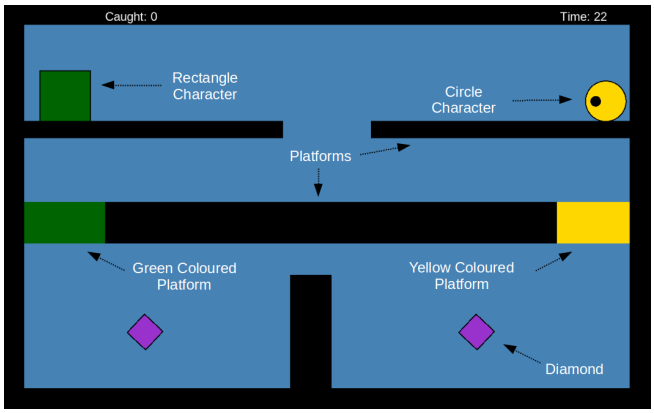

Fig. 1. A Geometry Friends level featuring all elements of the game: the circle and rectangle players, all three types of platforms and the purple diamonds.

1.1 Goal

The goal of this work is to design and implement a platform to support the Geometry Friends Game AI competition.

It mostly focuses on improving upon the existing procedures in charge of running and managing the competition.

2 GEOMETRY FRIENDS

This section introduces the Geometry Friends game (Section 2.1) and its competition (Section 2.2), the latter being the core subject of this work. A brief overview of current limitations revolving the competition are mentioned in Section 2.3.

2.1 About Geometry Friends

The Geometry Friends video game was created in the scope of a master's thesis by Rocha [1]. It is a 2D puzzle platformer where up to two playing characters - a yellow circle and/or a green rectangle - can move each with their unique set of actions to capture purple diamonds.

Figure 1 shows a sample level which not only includes the characters and collectibles mentioned above, but also the different platforms available: normal (black) impenetrable platforms, and coloured platforms. A player object will collide with a wall whose colour does not match its own.

One of the most interesting aspects of this game is the cooperation factor in levels/maps containing both the player characters. It is often required that both the characters combine their efforts and unique abilities to complete the levels.

2.2 Geometry Friends Competition

The Geometry Friends Game AI competition allows participants to implement agents which control one or both the player characters mentioned in the previous section, and then to compete against other participants in the same levels/maps.

Agent development can be done using a C# API, as the original game is also fully implemented in C#. Both the framework and sample agents can be found at the competition website[1].

Participants are then asked to zip their solution containing the source code developed and email it to the competition organizers, along with entry details such as team name, intended category (circle only, rectangle only or coop) and a 3-4 page technical report describing the solution.

A command line interface and a Geometry Friends Batch Simulator exist to help participants test their solutions or organizers to evaluate submitted solutions. However, in the latter case, the intervention of the organizers is required.

A competition has several public and private levels, to discourage overfitting solutions. Equation 1 determines the score of an agent on a single level, across R runs/simulations. It takes into account the level's time limit (T_{max}), the time elapsed (t_i), collectible (B) and completion (C_i) bonuses and the number of collectibles caught (N_i).

$$Score = \frac{1}{R}\sum_{i=0}^{R} C_i \frac{T_{max} - t_i}{T_{max}} + (B \times N_i) \quad (1)$$

More manual labour is then required to make all the results available on the competition website. There, all the results since the competition's first edition in 2013 can be viewed. Since then, the competition never grew much in popularity, averaging around 3-4 submissions per edition.

Several AI algorithms were explored across different editions so far. To name a few, the Dijkstra and MCTS algorithms were used in the edition of 2014, held at IEEE Conference on Computation Intelligence and Games (CIG) [2]; Subgoal A*, based on the result of a master's thesis [3], Rapidly-Exploring Random Trees [4] and Reinforcement Learning techniques [5] were used in 2015, an edition held at both at CIG and the Genetic Evolutionary Computation Conference (GECCO).

2.3 Current Limitations

While section Section 4.1 analyses and compares the current platform with the state of the art, this section mentions a few limitations that can be understood without much deeper knowledge on the subject.

Although the competition editions so far did not have many contestants, many of the competition management processes require manual labour which can scale with the amount of participants. These include the receiving of submissions, handling them (i.e. compiling and evaluating) and updating the results on the competition website.

Other drawbacks of this is that human intervention is prone not only to mistakes, but to delays as well.

[1] http://gaips.inesc-id.pt/geometryfriends (accessed October 13, 2017)

But before diving deeper into the problem of the GFGAI platform, it can be fruitful to investigate the state of the art regarding managing game based AI competitions and their platforms, to see how some of these limitations are being addressed by others.

3 RELATED WORK

Artificial Intelligence competitions are not new. They exist as a way to promote advances in the field by encouraging the development of AI based solutions and sharing with members of the community.

Popular examples lie in card games like Poker [6], real-time strategy (RTS) games such as StarCraft [7] and turn based games such as Pokemon [8].

In contrast to games, real-word applications are also explored in some competitions such as the Supply Chain Trading Agent Competition [9], transportation challenges such as the DARPA Grand Challenge [10] and human conversational intelligence such as the Loebner Prize [11].

Section 3.1 explores state of the art guidelines for managing and sustaining artificial intelligence competitions. Other AI competition platforms were analysed and the analysis of the most relevant platforms are present in Section 3.2.

3.1 Good Practices

Good practices for running AI competitions are explored and suggested by Togelius [12]. The author attempts to explain the reasons why some competitions fails and then suggests guidelines for the success of AI competitions.

As for why AI competitions fail, Togelius defends that the lack of continuity, stagnation and irrelevance are the main reasons. That is, competitions fail when they do not evolve and keep their challenges relevant to the field, and when low amount of effort is put into them, preventing continuity to the competition.

To run a successful AI competition, Togelius suggests that a competition should:

- be fully transparent in terms of rules and evaluation methods;
- be accessible on a wide variety of platforms and programming languages;
- be repeated to enable improvements over time;
- have a discussion group to encourage community interaction;
- have software that can run locally to test solutions more efficiently;
- have a game that can be sped up, useful to train learning algorithms;
- be easy on beginners, e.g. including sample agents and simple instructions;
- open-source everything, including solutions, to enable sharing and prevent cheating.

3.2 Game Competitions and their Online Platforms

During this work, several artificial intelligence competition platforms were studied. One of them, the Mario AI championship, is based on the Mario AI Benchmark, which is itself based on the Infinite Mario Bros game developed and made open-source by Markus Persson [13], [14]. Active between 2009 and 2012, the competition allowed agents to be developed in multiple languages, provided

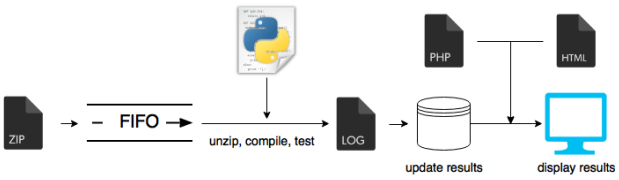

Fig. 2. A simplified diagram which describes the back-end processes of the gvgai competition (1-player tracks only)

Test Set 2-Player Games CEC 2017

Rank	Username	User ID	Country	Points	Avg. Glicko Score	Games Played	Controller
1	ToVo2	474	Slovenia	161	1183.41	5070	Download
2	ehauckdo	542	Brazil	111	1133.4	6338	Download
3	not2048	671	United Kingdom	111	1071.82	5170	Download
4	essex_acwebb	530	United Kingdom	110	1120.63	5070	Download
5	Number27	441	Germany	84	1124.69	5170	Download
6	sampleRS Sample	636	United Kingdom	78	1108.92	5349	Download

Fig. 3. Part of a table showing the 2-player track overall results of CEC 2017, as indicated by the label above the table. The first row explains what each column represents. Both the usernames and the "Download" are hyperlinks. *From http://www.gvgai.net/gvg_rankings_conf_2p.php?rg=2006 (accessed December 2, 2017)*

good documentation and community interaction via a Google site and a Google group. Submissions were sent via email.

Another studied platform is OpenAI's Gym, which is not necessarily a competition, but allows the submission, scoring and sharing of solutions based on reinforcement learning [15]. A more recent platform called Universe was developed on top of Gym to enable agents to mimic the actions of a computer user, i.e. by simulating keyboard and mouse events. An interesting feature of the OpenAI Gym's framework was the ability to automatically upload a solution to the platform using code.

The General Video Game AI (GVGAI) Competition, which like the OpenAI Gym promotes game agnostic solutions [16], is a modern competition platform that follows many of the guidelines suggested in Section 3.1. It allows submissions to be uploaded through the website. Its framework is cross-platform and agents may be developed in multiple languages (Java and Python). In a paper, the authors give an insight into the back-end processes of the platform [17] which we summarize and illustrate in Figure 2. An automated process such as this ensures high repeatability and low organizational costs.

As for the GVGAI results tables, they manage to act like a hub for lots of information regarding the submissions, such as score, participant profile and source-code download. This can bee seen in Figure 3.

Lastly, the Ms. Pac-Man Vs Ghost Team is a competition based on the popular arcade game Ms. Pac-Man [18]. Its website features a thorough step-by-step image based guide and also allows for submission uploads through the website. On of the features that stands out, however, is the existence of a controller packaging script, a Bash script that compresses the participant's solution into a single, submission ready file.

3.3 Mooshak

If we try to generalize and say that an AI competition is just like a general programming contest, we arrive at platforms such as Mooshak, an online platform to manage programming contests such as the ACM International Collegiate Programming Contest [19].

According to the authors, Mooshak is a web application running on a Linux based operating system. The core functionality is implemented as an Apache HTTP server and external programs written in Tcl.

Its strengths reside in completely automated evaluation of submissions, ability to run several contests and of big dimensions, multiple HTML based interfaces, etc. The mentioned interfaces exist not only for casual participants, but also for judges and competition managers to intervene and configure competition instances, respectively.

Limitations of this platform include, but are not limited to the facts that submissions are only evaluated according to predefined inputs and outputs and that only a single source file can be submitted as a solution. This and other limitations mean that configuring competitions of highly complex games with complicated submission files, complicated compilation and execution steps and so on, could prove not only hard if not even impossible.

4 FROM THE OLD TO THE NEW

Section 2 discussed the game and the competition. This section assesses the Geometry Friends competition according to what we have learned so far and pinpoints processes to improve, finishing with a conceptual overview of the new developed platform.

4.1 Weaknesses of the Current Geometry Friends Competition

When having Section 3.1 in mind, we can see that the Geometry Friends competition meets some of the studied guidelines, like having a discussion group (forum), being able to test the game locally, etc. However, it also has several flaws, such as (to mention but a few):

- organizational (human) work proportional to the amount of submissions per competition edition/instance;
- delays in the availability of competition related information, e.g. past submissions or technical reports, entire results, etc.;
- it is not open-source;
- only one language (C#) may be used in the development of agents.

While all these flaws represent important problems, solving the first two is of higher priority because they represent a threat to the continuity of the competition and a lack of transparency (information accessibility) to the participants and the general public.

4.2 How to Improve

There are three main groups of people that interact with the Geometry Friends competition: the organizers, the participants and the general public. When considering the main weaknesses previously presented, it is possible to present requirements that the new platform should meet for each of the three groups.

Most of this work focuses on improving the processes related to the organization of the GFGAI competition. With the new platform, an organizer should be able to:

- create new competitions and configure parameters such as name, start and end dates, evaluation formula parameters for each level, etc.;
- easily run predefined submissions to each competition to serve as baselines for the participants;
- have all the submission evaluation processes automated for them, including the reception of submission files, extraction, compilation, execution, obtaining results and making them publicly available, etc.;
- be able to update the GF game version in use when evaluating submissions, in case the game itself evolves in the future.

On the other side of things, a participant should be able to:

- understand if and what competition editions and categories are open for submissions;
- access competition details and parameters such as levels, time limits and bonuses for each level, etc.;
- have rapid access to the GFGAI competition framework and quickly create a simple agent;
- upload a submission for a desired competition instance automatically;
- understand what happened to his submissions (e.g. possible errors) and what their results were;
- submit multiple times to a single competition (given that only the latest submission counts);
- create and submit more complex submissions which may include C# dependencies and even platform specific dependencies.

Lastly, any person should be able to:

- have access to all current and past competition details (e.g. levels used, formula parameters) and results/scoreboards;
- have access to all submission files and technical reports submitted throughout the lifetime of the new platform (except for Competitions which have not finished yet for cheating prevention reasons);

4.3 Overview of the New Platform

In order to meet the requirements presented above, a new platform was developed. Due to the uniqueness of the Geometry Friends game and the complexities behind processing submissions for this game, it was concluded that solving the problem at hand would be a highly custom endeavour. As such, integrating the new platform with other existing platforms (such as the Mooshak platform described in Section 3.3) was deemed not only unnecessarily impractical but maybe even outright impossible. Therefore, the new platform was developed pretty much from scratch.

The new GFGAI competition platform was developed as a web application following a traditional LAMP model. Debian 9 was the specific Linux distribution chosen for deployment. The back-end was inspired by the GVGAI platform as shown in Figure 2. Conceptually, the platform's design can be split into two separate and almost independent components, as shown in Figure 4:

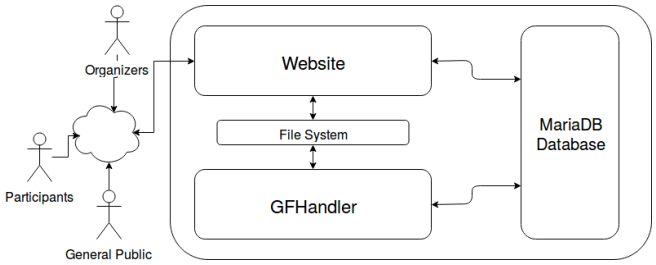

Fig. 4. Conceptualization of the two main components of the new platform.

Website the component which is meant to be interacted with, both by competition participants and administrators and is described in Section 5

GFHandler the background process which is in charge of handling submissions and more. Section 6 describes this component in more detail

They share a MariaDB database which stores most of the persistent competition related data, such as user information, submission details, existing competitions, etc. The two components also share files in the file system, such as the submission files.

Additionally, in the new platform's model, a competition is defined as a specific instance (e.g. edition) and a specific category/track. This is because each unique set of Geometry Friends levels is defined in an XML file, and so it makes sense to associate each competition with a single XML file. For the following chapters, most of the uses of this word will take this definition.

5 THE NEW WEBSITE

The new website is one of the two main components introduced in the previous section. It is the only component which interacts directly with its users: organizers, participants and the general public.

In the interest of providing more details about each of the participants and to enable the re-use of participant data during continuous interaction with the website, a new account based system was developed for the new website.

Given that there is a clear separation between organizational and non-organizational tasks as per the requirements established under Section 4.2, it also makes sense to separate the functionalities related to those requirements on the website. This resulted in a feature segregation via multiple interfaces similar to what Mooshak does (Section 3.3), some of which are detailed below.

5.1 Management Interfaces

The management interfaces are meant to be accessed and used by the organizers alone. Because of this, a special type of user account exists, called an administrator account, which enables the use of these interfaces.

Organizers may create competitions and then configure them (see Figure 5) by editing, for instance, the number of simulations executed per level, the maximum submission size and level specific parameters (visibility and formula related parameters).

Pages exist which enable organizers to create and run presets, which can be thought of special "participants" controlled by the

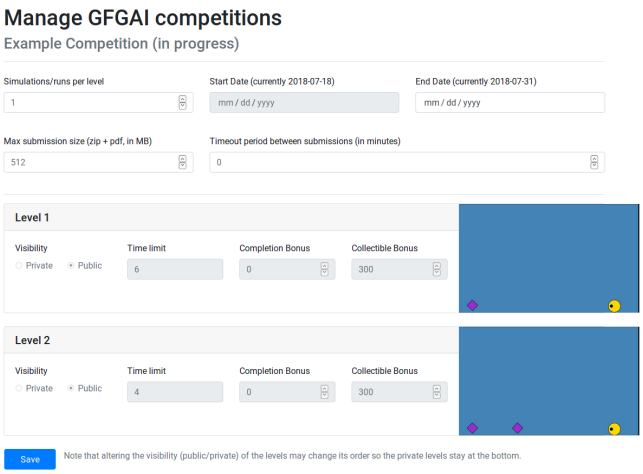

Fig. 5. Management interface to edit competition parameters. Different form fields can be disabled/enabled depending on the stage of the competition.

organizers which can "submit" a predetermined submission on command. This is useful to have constant baselines across the different competitions.

Lastly, in the interest of facilitating maintenance and future-proofing the whole system, the game version used when automatically evaluating agents may be updated by an organizer on the website.

5.2 Participating

A participant may pick an competition from a list of competitions and download what we call a competition package. This contains:

- a copy of the Geometry Friends game;
- a sample agent to help the participant get started;
- the XML world file describing the levels of the competition (only the public levels in case the competition in question is still ongoing);
- two packaging scripts (explained in Section 5.4.2).

After developing their solutions, participants may upload their submissions (requires logging in) through the website (see Figure 6 and track their results in real time on their profile page, which shows both the score and state of the user's submissions. This state indicates feedback from the GFHandler component, such as if the submission is still being processed, or if it had errors (the system may even make error logs downloadable), etc.

Finally, users may check the actual scoreboard (Figure 7) of the competition they submitted to in order to compare their results against other participants, as well as see their level specific scores and even request videos (see Section 6.4.2).

5.3 Additional Features

In the interest of transparency, anyone may download the source code and technical reports of all the submissions placed to any *finished* competition via download buttons on the scoreboards, thus centralizing a lot of information in one place.

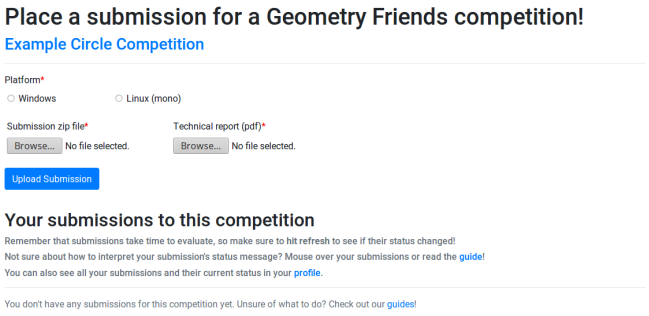

Fig. 6. Submission form to upload a ZIP solution.

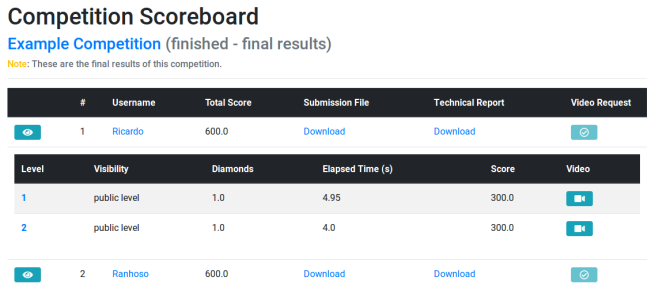

Fig. 7. Scoreboard with two entries. The first entry is expanded to show the results on each individual level.

Several image-based guides were also created to teach people how to use the new platform and how the server evaluated their submission. The organizers also have their own tutorial to learn how to manage competitions, and more. Lastly, a video tutorial was also created which shows the whole process of picking a competition, creating a simple agent and participating (submitting) using the website.

5.4 Implementation Details and Challenges

The developed website is a simple Flask[2] application, a web application capable of handling requests from an Apache HTTP web server through the use of WSGI technology. Bootstrap[3] was used as the front end.

5.4.1 Security Measures. The Geometry Friends competition is not highly popular at the moment, its visitors and participants are often academics and most of the valuable information is made public for everyone to see. Thus, it is hard to imagine the new competition platform as a big target for malicious attacks.

However, some simple security measures were still implemented, such as:

- SQL injection prevention, which means that the system will not interpret SQL queries attackers may write in text fields;
- both client and server side form validation, ensuring that the data received remains consistent with what is expected by the system;

[2]http://flask.pocoo.org/ (accessed August 7, 2018)
[3]https://getbootstrap.com/ (accessed August 7, 2018)

- the use of HTTPS, a protocol that improves the security of data transmitted between client and server.

5.4.2 Packaging Scripts. In the interest of minimizing network activity to reduce submission times and disk usage by the server, participants are encouraged to exclude the game files from their submission ZIP files. The game files occupy a significant amount of space and the server already has an up to date copy of it anyway.

To facilitate this process, two packaging scripts inspired by the Ms. Pac-Man vs Ghost Team packaging script were written: one in Bash (Linux, MacOS, etc.) and one in Batch (Windows). The scripts basically archive a user's submission into a ZIP file while automatically excluding redundant files. The Batch script relies on the open-source program 7-Zip[4] because the "Compress-Archive" PowerShell utility did not offer the needed functionality as the "zip" command line utility available in most *Nix systems.

6 GFHANDLER - THE SUBMISSION HANDLING PROGRAM

The most complex component of the developed platform is a background program which we called GFHandler, since it is meant to handle Geometry Friends submissions. This ended up being a very custom program tailored for the quirks of the Geometry Friends game and framework. To maintain consistency with the website, it was also written in Python 3.

6.1 The Main Loop of the GFHandler Program

The way it works is: it regularly polls the database for new (unprocessed) submission entries. These submissions are the ones that are uploaded using the new website, using forms such as the one in Figure 6. Each submission entry in the database possesses all information required by the GFHandler program, one of the most important being the path where the uploaded submission was stored to.

If there is a new submission to handle, the GFhandler starts a new submission environment, as detailed in the continuation. The environment is populated with the necessary files—namely the submission ZIP file, a fresh copy of the game and the XML world file for the competition.

Once everything is in place, a series of events occur automatically, including extraction of the ZIP file, compilation of the source code, simulation of the compiled agent on the several levels (usually multiple times per level, a parameter which can be configured by an organizer using the new website) and extraction of the results from a game generated Results.csv file. Each of these actions may be implemented differently in each submission environment. Figure 8 shows a simplified diagram of the main loop of the program.

This means that the GFHandler program is always on. Being deployed in a Debian operating system, the GFHandler component is managed as a `systemd`[5] service, which enables it to start automatically at boot, along with other benefits.

[4]https://www.7-zip.org/ (accessed August 7, 2018)
[5] https://www.freedesktop.org/wiki/Software/systemd/ (accessed August 2, 2018)

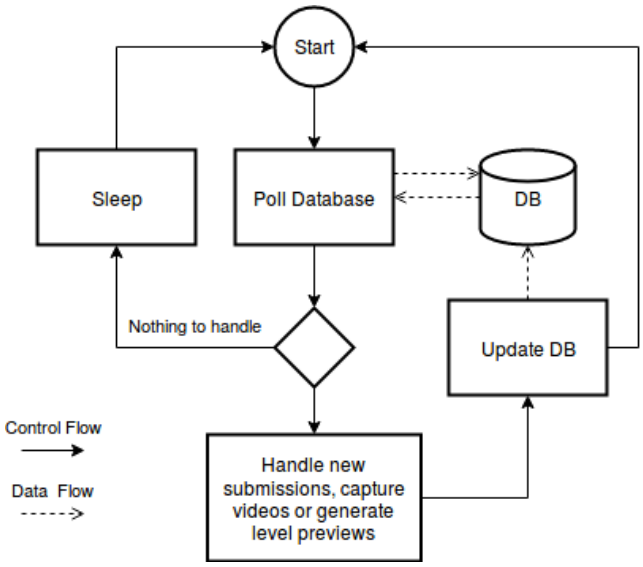

Fig. 8. A flowchart containing the core elements and procedures of the main polling loop of the GFHandler program.

6.2 Simulation Environments

As mentioned above, submissions are mostly handled in special environments which were given the name of submission environments. The idea is that encapsulating the submission handling processes may allow for:

- a standardization of the mentioned processes, which may enable all submissions to be handled in the same way and thus simplifying the system;
- fairness, if the environment is guaranteed to be equal for every submission;
- modularity, in the sense that the environments may change (e.g. by adapting to future changes of the Geometry Friends framework) without the need to modify the main program/loop of the GFHandler;
- isolation, whose magnitude may vary depending on the environment chosen, but grants a greater security for the host system in case a malicious submission is sent.

However, there is one requirement that highly influenced the environments to use, which is the fact that participants may resort to platform specific third-party libraries to build their agents. Given that the platform was designed to be deployed under a Linux based system, this meant being able to run Windows specific .NET dependencies under Linux.

Some attempts with containers were made to try and meet some of the characteristics of the envisioned environment, such as isolation. However, containers work by sharing the kernel of the host system, which means that it is impossible to create a Windows container on a Linux platform.

Eventually, virtualization was chosen as a way to accomplish the desired outcome. With virtual machines, we can:

- run entirely different operating systems on the same machine (e.g. Windows on Linux) and thus avoid the problem of platform specific dependencies;
- have isolated environments with no access to the host system where even malicious submissions can be executed safely;
- save a clean state (snapshot) of a virtual machine and then restore a machine to that snapshot every time we handle a submission to ensure a constant and fair environment for each submission.

VirtualBox[6] is an open-source virtualization product that possesses all the necessary features (e.g. snapshots), and thus, it was used in this work. The problem now was deciding what virtual machines to use and how to have them do what is required for the submission handling system.

6.2.1 Choosing the Operating Systems. The majority of Geometry Friends agents are written either using Microsoft .NET Framework (under Windows) or the Mono[7] open-source and cross platform .NET framework (commonly under a Linux distribution or macOS). Thus, it made sense to create two virtual machines, one with a Windows release, and other with a Linux distribution running Mono.

Either way, since the machines were meant to be deployed in a headless server with limited storage, preference was given to lightweight operating systems.

The Linux distribution chosen was Ubuntu Server 18.04, a recent operating system based on the popular Ubuntu distribution, but even lighter.

Finding a lightweight Windows operating system is not easy. Microsoft Nano Server[8] is an installation option of Windows Server 2016 which is lightweight and meant for headless systems. However, it only supports 64-bit programs, which was a deal breaking concern since it would place limitations on the participants. In the end, a standard Windows Server 2016 installation was used.

6.2.2 Communicating with the virtual machines. In order to process submissions inside the VMs, it was necessary to tell the VMs to run commands. SSH is a protocol that allows our host system to run commands remotely and to transfer files (SCP).

Because the machines are reset to a clean snapshot every single time before they handle each submission, they are powered off when inactive. Therefore, a constant SSH connection to the machines cannot be kept. Before re-establishing the connection, a safely large sleep is performed on the GFHandler when booting the machines.

As for the specific implementation, OpenSSH was setup on both virtual machines.

6.3 Handling the Submissions

The default shells of Ubuntu and Windows are Bash and cmd, respectively. They are both quite different. However, Windows also has a second shell, PowerShell. Due to some of PowerShell's command aliases, basic commands can be used interchangeably between the two shells (PowerShell and Bash). In summary, the GFHandler program runs commands remotely (via SSH) in either Bash or PowerShell, depending on the environment currently in use.

A status attribute of the submissions (also stored in the DB) not only allows the GFHandler to see which submissions have already been processed, but also to update their progress in real time. For instance, a submission may be "new", "processing", have an "extraction error", or it may have been evaluated on all public levels, in which the status becomes "handled public", and more.

After detecting the existence of an unprocessed submission, GFHandler decides what levels the submission needs to be simulated on by looking at whether the respective competition has ended or not. Only when a competition ends will the submissions be evaluated on that competition's private levels.

Another flag in the database indicates whether the submission should be run using Windows or Linux. The respective VM is restored to a clean snapshot and is booted. After a while, the GFHandler connects to the VMs via SSH and begins a series of steps:

(1) sends the submission ZIP to the VM via SCP;
(2) extracts the ZIP (with "unzip" in Ubuntu or "Expand-Archive" in Windows PowerShell);
(3) removes unwanted files from the user's submissions (e.g. the game files, in case the participant also sent them in their submission) to prevent cheating with altered files;
(4) sends a fresh copy of the game and an XML file containing only the levels that the submissions is supposed to be evaluated on (i.e. only the public levels if the competition has not ended yet);
(5) builds the solution using MSBuild[9], which should generate a DLL file containing the submitted agent;
(6) simulates the agent on the levels, R runs per level, computing the score using equation 1 by parsing the results from a game generated CSV file;
(7) powers off the VM.

Step 4 allows the game to be updated in the host system (e.g. due to future improvements to the original Geometry Friends game) without the need to change the snapshots of the virtual machines.

On step 6, the score on each level is computed and updated on the database before the agent is simulated on the next level, making it possible to track the progress in real time.

6.3.1 Error Handling. The GFHandler program is expecting errors at any stage of this recipe. In high profile steps such as extraction or compilation, submissions may fail and be assigned a status message accordingly, like "extraction error" or "compilation error".

In unexpected cases, mostly in steps unrelated to the participant's files specifically, the program may still fail. This causes the submission to have a "server error", which may indicate a critical and urgent issue with the program. In this situation, an automated email is sent to the organizers containing the description of the error and the latest log file of the GFHandler program.

Contrary to errors in other stages, the GFHandler does not scrub the entire progress after encountering runtime errors during game

[6] https://www.virtualbox.org/ (accessed August 1, 2018)
[7] https://www.mono-project.com/ (accessed July 25, 2018)
[8] https://docs.microsoft.com/en-us/windows-server/get-started/getting-started-with-nano-server (accessed July 25, 2018)
[9] https://msdn.microsoft.com/en-us/library/dd393574.aspx (accessed July 26, 2018)

simulation. This is to give a chance to agents who just crash occasionally to still get some score for the levels where they performed well.

For both compilation and runtime errors, the GFHandler program stores the output of the executed commands in a separate text file which can later be displayed to the participant through the website. This text file may contain, for example, the output of a MSBuild's compilation, or even runtime error traces for each individual level the submission failed on.

6.4 Capturing Level Previews and Submission Videos

Some pages of the new website show pictures of the different levels of a given competition, such as the competition configuration page (Figure 5) and others. These are generated automatically after a competition manager creates a new competition by uploading a new world file.

These previews need to be generated dynamically because the world file is just an XML file describing the levels, which means that no pictures of the levels exist at the moment of competition creation.

6.4.1 Level Previews. There are several ways to take screenshots of a window, but fortunately the Geometry Friends game already included a feature which allows agents to take screenshots of the game screen. This is useful for vision based algorithms.

A simple agent was created which does not move and at the start of the game captures a screenshot, saves it in a specific location and then exits the game. Since this agent can be run on Linux and there are no fairness or security concerns in this situation, its execution is done in the host system to save time by avoiding booting up a virtual machine.

6.4.2 Generating Submission Videos. The Geometry Friends Game AI competition of 2013 included embedded YouTube videos of the submitted agents. Although only one submission was received, thirty videos were made: ten (one per level) for each of the three categories. Submission videos are good for mainly two reasons:

- it allows participants to see how their agents are running in the competition server;
- it allows the general public to better see how the algorithms used in the different submissions perform in the different levels, without having to download and run the source code themselves.

The GFHandler also has the ability to generate videos of submitted agents. It does so by using the capability of the game to generate screenshots as described above. Generating a video of a submission is just like evaluating an agent on all levels (but only a single run per level), but with the following variations:

(1) environment setup as described in Section 6.3
(2) 24 screenshots are captured per second during execution of a submitted agent on a single level, and are named "image_xxxxx.png" in ascending order (4 digits would only allow for 7 minutes of video at 24 frames per second, so just to be safe 5 digits are used here, which allow for roughly 70 minutes);
(3) the captured screenshots are extracted from the virtual machine using SCP and placed in a temporary directory;
(4) a WebM video is generated from the images using FFmpeg[10]
(5) the video is copied to a folder accessible by the website, the database is updated accordingly, the temporary folder is deleted and if there are any levels left, GFHandler goes back to step 2.

However, because taking screenshots is a feature of the framework, it is something that must be done in code. To avoid injecting code into the participant's submissions, the original source code of the game itself was modified.

6.5 A Step Into Open-sourcing the Geometry Friends Game

As Section 6.4.2 mentioned, the original code base of the Geometry Friends game was modified in order to complete the video generation feature of the GFHandler. Other modifications were also made to the Geometry Friends code which represent a move forward towards open-sourcing the game. These modifications enable the Geometry Friends code to be compiled in a cross-platform environment, since it previously only compiled under Windows. The modifications mostly came down to:

- adapting file name references to case-sensitive systems;
- changing hardcoded Windows-specific path references to dynamically constructed cross-platform paths.

As Section 3.1 mentioned, open-sourcing is considered a good practice to run game-based AI competitions.

6.6 Additional Implementation Details and Challenges

The development of the GFHandler did not happen without setbacks or challenges. This section describes four of the most significant challenges faced.

6.6.1 Windows Forms and the Absence of a Display. Windows Forms is a GUI library which is part of the .NET Framework. Geometry Friends uses it when creating the game window or, when specifying through the GF's command line interface a `--no-rendering` option, a single window with a single button to stop the simulation.

When attempting to run the game in a Ubuntu Server headless VM, it would not work. While initially this was thought to be due to the absence of graphic libraries, which is common in most Linux based server distributions, it was later concluded that the problem was, in fact, due to the absence of a connected display (a headless Lubuntu 16.04.4 with all the standard graphics packages was used to confirm this hypothesis).

Since even the `--no-rendering` option of the Geometry Friends game rendered a window, the first solution to tackle the problem was to modify the source code of the Geometry Friends game to remove this simple window, as its functionality was not critical to the simulations.

However, it was later found out that a previous submission, which was based on computer vision, performed worse when the game was told not to render the game. The image capturing functionality

[10] https://www.ffmpeg.org/ (accessed July 26, 2018)

of the game simply did not work as a consequence of not rendering the game.

It was now certain that the game had to be rendered, but in rendering the game, windows forms would definitely be used, which caused the simulation under a headless Ubuntu Server not to work. No errors were found under Windows, however. Since Windows is a closed system, it was too difficult to determine how it tackled this problem. It could come down to differences in the .NET Framework implementation, or in the operating system itself.

Finally, the solution which ended up being implemented was to use a virtual display under the Ubuntu Server machine. The use of this virtual display may be the reason why the videos generated under the Linux VM tend to have a lower frame rate that the ones generated under the Windows Server VM. The specific virtual display program used was Xvfb[11].

6.6.2 What the GFHandler Does When Simulation Hangs Indefinitely. The majority of the errors the GFHandler is capable of detecting during the handling of a submission are detectable because commands which fail return an error code, which the Python module the GFHandler uses to run external commands, "subprocess"[12], can detect.

However, no error code is returned from commands which fail but hang indefinitely. This can be the case when simulating agents in the Geometry Friends game, which may have errors that cause the game to stop responding but never exit.

The solution was to use timeouts. Under the Ubuntu Server VM, commands were run in Bash via SSH. Hence, the "timeout"[13] utility of the GNU coreutils was used to detect when the execution exceeded a certain time limit and subsequently kill the Geometry Friends process. The timeout command then exits with a 124 error code, enabling the GFHandler to distinguish between casual runtime errors and hanging.

In Windows PowerShell, the same functionality proved unnecessarily difficult (maybe even impossible) to accomplish. As such, the GFHandler itself (using Python's subprocess module) checks for timeouts on the game execution command. If it times out, a second command is required to kill any process named "Geometry Friends". While a specific process identifier would have been preferred here, it simply was not feasible. A third command forces PowerShell to exit with a 124 status in this scenario to retain consistency with the Linux behaviour.

It is important to note here that while the subprocess module may indeed kill the process it generated after a certain timeout is achieved, the only process that would be killed would be the SSH process, and not the process generated by the SSH commands.

Lastly, a generous global timeout is applied to all external commands (not just agent simulation) to ensure the GFHandler never stops working. But in this case, we do not worry about killing any process which might still be running inside the VM.

6.6.3 Recuperating Submission Handling Procedures. In the event that the GFHandler program may be restarted, either due to server maintenance, power outage, or other reasons, the GFHandler may be able to resume submission handling procedures.

By looking at the database and inspecting the status code of a submission, as well as the number of levels an interrupted submission has already been evaluated on, the GFHandler is made so it can resume the evaluation process from where it left off.

However, this feature is really hard to test, as there are many different points along the evaluation procedures where an unexpected halt of the GFHandler program may occur.

6.6.4 Tackling a Resolution Inconsistency Originated by the GF Game. A significant amount of different operating systems were used throughout the development of the new platform and it was noticed that the Geometry Friends game window would have different sizes. A quick look at the Geometry Friends code did not prove useful, but it is important to recall that the original Geometry Friends code was not the subject of this work.

This problem only seems to impact the screenshots taken during gameplay, which then impact the level previews and submission videos. As such, some larger images captured during runtime inside the VM's are then being cropped after being collected by the host system to fit a certain dimensions. The `convert` and `mogrify` commands of the ImageMagick [14] image manipulation tool suite are used for that end.

7 EVALUATION

The platform was evaluated mostly in terms of ability to complete tasks as the ones mentioned in Section 4.2, as well as general usability using a System Usability Scale[15] (SUS) questionnaire. Three test scenarios were designed, one testing the organizational side of things, and the other two the participants side.

The competition management (CM) test had testers create and edit a competition. It also had them create and run preset submissions and track their results.

A thorough user (TU) test had testers act as participants by creating and logging in with a new account, downloading a competition package (game + sample agent + competition specific levels), creating a simple agent in VisualStudio or MonoDevelop, submitting it to a certain competition and tracking the results.

A simple participant (SU) test made available two ready to upload submissions and had the testers create and login with a new account, submit two submissions and track their progress. Testers were also asked to explain what had happened to their first submission after submitting the second.

The three tests were concluded with an SUS questionnaire. Google Forms was used to distribute the questionnaires and collect the responses.

Each person asked to evaluate the system could do only two of the scenarios, since both TU and SU scenarios had overlapping tasks. As such, each person could test the competition management scenario and one of the competition participant's scenario.

Lastly, people were invited to do the tests using their own computers, as a way to also test the platforms across different systems

[11] https://www.x.org/archive/X11R7.6/doc/man/man1/Xvfb.1.xhtml (accessed July 30, 2018)
[12] https://docs.python.org/3/library/subprocess.html (accessed July 30, 2018)
[13] http://man7.org/linux/man-pages/man1/timeout.1.html (accessed July 30, 2018)
[14] https://www.imagemagick.org/script/index.php (accessed July 30, 2018)
[15] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (accessed August 3, 2018)

Scenario	# tests	Average SUS	σ
CM	18	76.6(6)	17.1499
TU	9	78.3(3)	16.0078
SU	9	84.4(4)	10.8813
Total	36	79.027(7)	15.4605

Table 1. SUS results for all the test scenarios.

and browsers. However, since some people could not bring their own or did not have VisualStudio installed on theirs (in the case of the TU scenario), a Windows 10 computer on campus, with VS2017 installed, was used in many tests. From this point forward, we will refer to this computer and this setting as the LAB.

7.1 Pre-tests

Three people pre-tested the system in the LAB on July 9, 2018, resulting in three CM, two TU and one SU tests. While the sample is quite small for statistical inferences, the feedback provided enabled some small changes to be made to the system, mainly:

- the video tutorial was updated to clarify the usage for the Windows submission packaging script;
- several helpful tooltips were added across the website to clarify things like submission status, video status, etc.;
- an incorrect HTML form was occasionally causing a "bad request" error, which was then fixed;
- a JavaScript alert was introduced in the system which alerts users of unsaved changes when exiting the competition editing page.

7.2 Final Tests and Results

A total of 18 people participated in the tests, from July 20 to September 14. The number of people on each test can be found on Table 1, along with the SUS scores.

All but one testers were students between 18 and 25 years old, and 2 were female. Over half of the people tested studied IT related fields. One researcher of a relevant field of study, AI for games, also participated. While most people used the LAB computer, some personal computers were still used, which provided significant diversity in terms of browsers (Microsoft Edge, Google Chrome, Mozilla Firefox, Safari) and operating systems (Windows 10, Ubuntu 16.04, MacOS).

According to results of 500 SUS evaluations[16] [17], the average SUS score is 68 and achieving over 80.3 is considered a very good score. This means that an above average score was obtained in all our scenarios, and the SU scenario obtained a very good usability score. This is not surprising as it was designed to be the simplest scenario.

However, because the main difference between the TU test and the SU test is the fact that the TU test required interaction with the GF framework, the difference in scores may indicate that the website is more usable than the framework.

As for the tasks of each scenario, everyone was able to complete them, and the majority was able to give the expected answers, meaning they were able to create a simple agent successfully (TU) and understand what happened to their submissions/presets (all scenarios). On the CM scenario, everyone created and configured a competition correctly as expected.

However, in a couple of tests (a SU and a CM tests), people did not give expected answers when reporting on the status of their submissions/presets, mostly because they did not realize that the system was evaluating their submissions in real time.

Based on this and many other problems and suggestions, several changes were made on the new website. These were mostly intended to improve the website navigation, as well as clarify certain aspects that the testers would find confusing. This was accomplished mostly by implementing better labels, helpful messages, and introduce more hyperlinks to relevant pages.

A researcher, who had experience using platforms such as Mooshak, complimented the new platform, more specifically the modern design of the website.

7.3 Successfully Implemented Requirements

The tests above were meant to directly address and test many of the requirements listed in Section 4.2. The features that were not only successfully implemented but also tested by people include:

- the creation and configuration of new online competitions;
- the ability to create and run predefined baseline submissions;
- the fully automatic evaluation of new submissions, from the reception of new submission files, to the displaying of the final results on the new competition website;
- ability to discern what competitions exist and whether they are still ongoing and open to new submissions or not;
- accessibility to information and parameters of existing competitions;
- the participant's access to the framework and ability to create a simple agent with the provided tools;
- submission upload by the participant;
- display of evaluation results to the participants, including possible errors;
- ability to submit multiple times to the same competition;

In fact, only four of the requirements were not directly tested by any of the three scenarios, but were tested and concluded to be working during the development phase of the project:

- the CM scenario did not include updating the game version, so this feature was only tested during the development phase;
- the TU scenario asked participants to develop a simple agent which would work on either platform, so the only tested submissions which used platform specific libraries were submissions from previous years tested during the development of the platform;
- no scenario included checking details and results of competitions (including finished competition) without being logged in either as a participant or administrator, but these pages are nonetheless accessible without logging in;
- no scenario asked people to download either the source code or the technical reports of submissions of finished competitions, but it is possible to do so.

This means that all the requirements mentioned in Section 4.2 were successfully implemented.

[16] https://measuringu.com/sus/ (accessed August 12, 2018)
[17] https://uiuxtrend.com/measuring-system-usability-scale-sus/ (accessed August 12, 2018)

One of the most important requirements was the automation of the submission handling processes and the GFHandler never failed during any of the tests. As a matter of fact, all the improvements made to the platform after receiving feedback from the testing phase were only related to the new website, and not to the GFHandler component.

Additionally, it is interesting to note that the GFHandler was operational uninterruptedly for the whole evaluation phase, which span across multiple months.

Other additional features which were implemented along the way, such as a password reset feature, sending error logs via email, etc., were tested and proved to work during the development phase of this project.

8 CONCLUSION

A new and functional platform for the Geometry Friends Game AI competition was developed successfully during this work.[18]

From the evaluation conducted in Section 7, we can see that its main features were tested by several people and multiple improvements to the new website were made having valuable feedback in mind. The usability of the website was considered above average using a System Usability Scale (SUS). No improvements were required on the platform's second main component, the GFHandler, as it functioned exactly as intended throughout all the tests.

All the requirements specified in Section 4.2 were successfully implemented, and most of them tested repeatedly on the aforementioned tests with people. Thus, we can conclude that the main goal was accomplished, since a new platform was indeed designed, implemented and deployed successfully.

Moreover, considerable care was put into producing helpful documentation not only for the participants and competition organizers (extensive guides on the website), but also to whomever will be in charge of maintaining or continuing the development of the platform itself. With respects to the latter case, a technical report/manual of considerable length and detail was written.

The implemented solution takes a different approach from the current state of the art, mostly by using virtualization during submission evaluation processes to guarantee security against malicious foreign code, fairness and more freedom for the participants to develop platform specific code, if they so desire. While this provides a small overhead in submission evaluation times mostly due to booting up the virtual machines, this is still small compared to the rest of the necessary evaluation steps.

8.1 The Future of the New GFGAI Competition Platform

Although the new platform is working correctly and as expected, time did not allow for a more extensive and complete feature list to be implemented. For instance, if the number of created competition begins to rise by a lot, it may be difficult to navigate through all of them. Although active competitions appear first on the web page, implementing a search or filter feature would make it more future proof.

While the guide states that all times are in Lisbon time, this may not be very international friendly. A solution would be to convert automatically all dates (e.g. competition start and end dates), and perhaps store them all in UTC inside the database.

Another limitation of the new platform is that the game may only be updated when there are no open/ongoing competitions, to avoid inconsistent results. A way around this would be to enable updating the game version and store it in the server under a temporary location until all open competitions ended. Then, the GFHandler could detect this and swap the game versions automatically.

Although the new platform is quite secure due to its architecture based on virtual machines, it may still be possible to cheat. For instance, the GFHandler reads the results out of a CSV file generated by the Geometry Friends game. The developed agents are permitted access to files because several AI implementations may depend on external files, but a malicious agent could attempt to write its own score on the CSV file. A solution for this problem could be to cryptographically sign the results file. This would likely require a customized version of the Geometry Friends game to exchange a cryptographic key with the GFHandler, which the agent would not have access to.

Lastly, several limitations of the Geometry Friends game could be more easily addressed in the future if it were open sourced. Some steps were taken in this work towards that end.

8.2 The Way Forward for AI Competition Platforms

Unfortunately, because of how the Geometry Friends game is implemented, the new developed platform had to be quite specific and tailored to the quirks of the game.

However, picking on some of the ideas explored in this work, it is possible to envision a more general platform for game based AI competition hosts who may want to use games with special needs similar to the Geometry Friends game:

(1) a competition specification containing some meta-parameters could describe what a competition is about, how it is named, and more (not to be confused with specifications of competition instances which would include, in the case of the Geometry Friends competition, starting and end dates, maps, etc.);
(2) a modern and open-source website template that could be cloned from a repository and, along with the specification in (1), immediately take the shape of a website of the specified competition;
(3) a program similar to but far more generic than the GFHandler could be developed and made open-source, meaning that people had to configure a virtual machine image containing the exact software they wanted (sample generic images could be provided and technologies such as Vagrant [19] could be explored here, instead of just using VirtualBox);
(4) the program mentioned in (3) would be configurable in terms of what commands should be executed, similarly to what Mooshak does.
(5) these components, (2) and (3), could be packaged together along with a sample database containing generic competition

[18] The platform was deployed and is accessible at https://geometryfriends.gaips.inesc-id.pt/

[19] https://www.vagrantup.com/ (accessed August 14, 2018)

related tables (e.g. users, scores, etc.) and scripts to install everything.

This would hopefully lower the barrier to create and manage game based AI competition platforms, resulting in more of them to make an appearance.

Then, a meta-platform could be created in order to index all of these competitions into one single website. This could allow AI enthusiasts to search for the ideal competitions that better match their work or that simply interest them the most. It could also provide visibility for less known competitions.

REFERENCES

[1] J. B. G. Rocha: "Geometry Friends", Master's thesis, University of Lisbon. 2009

[2] R. Prada, P. Lopes, J. Catarino, J. Quitério and F. S. Melo: "The Geometry Friends Game AI Competition", in proceedings of CIG'2015 – IEEE Conference on Computational Intelligence and Games, pp. 431-438, Tainan, Taiwan. August 2015. IEEE.

[3] D. Fischer: "Development of search-based agents for the physics-based simulation game Geometry Friends", Master's thesis, Maastricht University. 2015

[4] R. Soares, F. Leal, R. Prada, F. S. Melo: "Rapidly-Exploring Random Tree approach for Geometry Friends" in proceedings of DiGRA/FDG'16 – Proceedings of the First International Joint Conference of DiGRA and FDG, nº 1, vol.13, Dundee, Scotland. August 2016. Digital Games Research Association and Society for the Advancement of the Science of Digital Games.

[5] J. Quitério, R. Prada, F. S. Melo: "A Reinforcement Learning Approach for the Circle Agent of Geometry Friends", in proceedings of CIG'2015 – IEEE Conference on Computational Intelligence and Games, pp. 423-430, Tainan, Taiwan. August 2015. IEEE.

[6] D. Billings, D. Papp, J. Schaeffer, and D. Szafron: "Poker as a testbed for machine intelligence research", in Advances in Artificial Intelligence, pp. 228–238. 1998

[7] M. Buro, D. Churchill: "Real-time strategy game competitions", in AI Magazine, vol. 33, no. 3, pp. 106–108. 2012

[8] S. Lee, J. Togelius: "Showdown AI Competition", in Proceedings of the IEEE Conference on Computational Intelligence in Games, pp. 191-198. 2017

[9] R. Arunachalam, N. M. Sadeh: "The supply chain trading agent competition", in Electronic Commerce Research and Applications, vol. 4, no. 1, pp. 66–84. 2005

[10] G. Seetharaman, A. Lakhotia, E.P. Blasch: "Unmanned Vehicles Come of Age: The DARPA Grand Challenge", Computer 39, 26–29. 2006

[11] D. Powers: "The Total Turing Test and the Loebner Prize", in Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning, NeMLaP3/CoNLL '98, Association for Computational Linguistics, pp. 279-280. 1998

[12] J. Togelius: "How to run a successful game-based AI competition", in IEEE Transactions on Computational Intelligence and AI in Games, vol. 8, pp. 95-100. 2014

[13] S. Karakovskiy, J. Togelius: "The Mario AI Benchmark and Competitions", in IEEE Transactions on Computational Intelligence and AI in Games, 4(1):55–67. 2012.

[14] J. Togelius, N. Shaker, S. Karakovskiy, G. N. Yannakakis: "The Mario AI Championship 2009-2012", in AI Magazine, vol. 34, no. 3, pp. 89–92. 2013.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba: "OpenAI Gym", in arXiv preprint arXiv:1606.01540. 2016

[16] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas: "General Video Game AI: Competition, Challenges and Opportunities", in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. 2016

[17] D. Perez, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas, A. Couetoux, J. Lee, C. Lim, T. Thompson: "The 2014 General Game Playing Competition", in IEEE Transactions on Computational Intelligence and AI in Games, DOI: 10.1109/TCIAIG.2015.2402393. 2015

[18] P.R. Williams, D. Perez-Liebana, S.M. Lucas: "Ms Pac-Man Versus Ghost Team CIG 2016 Competition", in Proceedings of the IEEE Conference on Computational Intelligence and Games. 2016

[19] J. P. Leal, F. Silva: "Managing Programming Contests with Mooshak", Universidade do Porto. 2001