TÉCNICO LISBOA

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

Ricardo Filipe Gomes da Costa

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Francisco António Chaves Saraiva de Melo
Prof. Rui Filipe Fernandes Prada

Examination Committee

Chairperson: Prof. Mário Jorge Costa Gaspar da Silva
Supervisor: Prof. Francisco António Chaves Saraiva de Melo
Members of the Committee: Prof. António Manuel Ferreira Rito da Silva

October 2018

Acknowledgments

I would like to thank my thesis supervisors Francisco Melo and Rui Prada for extremely valuable guidance and orientation throughout most of my work. Their constant feedback was a huge help both in the development phase of this work and in the writing of all the necessary documentation.

I would also like to mention my colleague and lab manager Luís Silva for his help deploying the developed system on a machine inside university campus.

Lastly, I must thank all the people who participated in the testing phase of this work for allowing my work to be evaluated and validated.

Abstract

This document 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. The Geometry Friends game 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. After analysing how the current Geometry Friends competition fairs against the state of the art, several 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, which both participants and competition organizers may interact with, 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. It was also possible to conclude that the main objective was accomplished, since a new platform was successfully developed.

Keywords

Artificial Intelligence, Video Games, Competitions, Geometry Friends, Automation

Resumo

Este documento descreve o desenvolvimento de uma nova plataforma de competição online para suportar o futuro da Geometry Friends Game AI competition, uma competição de inteligência artificial baseada no jogo Geometry Friends. O Geometry Friends é um jogo de plataformas 2D de cooperação, cujas características principais são abordadas no início deste trabalho, juntamente com pormenores sobre como a competição tem funcionado até hoje. Várias outras plataformas de competições de IA são estudadas, assim como as práticas que são consideradas como estado da arte ao gerir boas e prósperas competições de vídeo jogos e IA. Após analisar a maneira como a competição do Geometry Friends se compara com as práticas referidas e com outras plataformas modernas, vários requisitos que a nova plataforma deve cumprir foram descritos, incluindo facilitar a participação online e a automação de vários processos relacionados com a manipulação das submissões. A nova plataforma foi maioritariamente desenvolvida de raiz, consistindo de um novo website, o qual tanto participantes como organizadores da competição podem interagir com; um programa encarregue de lidar completamente com as submissões recebidas; e mais. Virtualização foi usada para criar ambientes de avaliação seguros, justos e reutilizáveis. Várias pessoas testaram, recorrendo a três cenários diferentes, diferentes características da nova plataforma. Concluiu-se que o novo website possui uma usabilidade acima da média, enquanto que o programa que lida com as submissões funcionou como esperado durante a fase de testes. Foi também possível concluir que o objetivo principal foi cumprido, sendo que uma nova plataforma foi desenvolvida com sucesso.

Palavras Chave

Inteligência Artificial, Vídeo Jogos, Competições, Geometry Friends, Automação

Contents

List of Figures

List of Tables

Listings

1

Introduction

Contents

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. Winning parties are usually offered rewards, although in the case of the Geometry Friends Game AI competition this has not happened yet, since the competition has yet to receive a significant number of entries. Rewards and prizes are considered to be an incentive for people to participate in competitions.

However, prizes are not the only incentive for participation. Several other 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 GF-GAI 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.

1.1 Goals

The main goal of this project is to design and implement a platform to support the Geometry Friends Game AI competition. The design shall take into account three stakeholders: competition organizers, participants and the general public. The following more specific sub-goals are defined:

- liberate the competition organizers from all tasks related to submission handling/evaluation by automating those tasks;
- develop online interfaces which allow organizers to manage the GFGAI competition and configure its future editions;

- develop online interfaces which allow participants to compete in the GFGAI competition;
- the platform shall offer automatic feedback to participants regarding their submissions to the competition;
- knowledge (participant's solutions, etc.) generated by the competition shall be made available automatically on the new platform to increase accessibility of information and avoid delays;
- achieve an above average usability score on the System Usability Scale[1].

Research on other competition platforms and on the state of the art of the best practices surrounding competitions of this genre shall guide the design of the new platform by allowing system requirements to be established.

The system shall be evaluated according to the implemented requirements and to usability tests.

1.2 Outline of the Document

Chapter 2 starts by describing the Geometry Friends game and its competition, which is the main focus of this project. Details about the competition include the existing framework, its current submission process, evaluation method, past editions and so on. After this, some current limitations of the platform are pointed out (Section 2.3).

The related work is approached in Chapter 3, which starts with an overview of competitions in the context of artificial intelligence (Section 3.1). It then identifies good practices when running game competitions (Section 3.2). These are guidelines that the new platform should follow. After this, a few competition platforms are studied (Section 3.3) in order to ascertain if and how they meet some of the good practices mentioned above. Lastly, we analyse a more generic programming contest platform, Mooshak (Section 3.4).

With all the other platforms and good practices studied in the related work, we shall revisit the Geometry Friends in Chapter 4 to address its limitations (Section 4.1) and to establish what needs to be done in order to improve it (Section 4.2).

Having some requirements already on the table, Chapter 5 introduces the new Geometry Friends competition platform developed to meet these requirements. This includes giving an overview of the new system (Section 5.2), where two main components are introduced: the new website, explained in greater detail in Chapter 6, and the GFHandler, which is detailed in Chapter 7.

The evaluation of the new platform can be found in Chapter 8, which includes tests with people (Sections 8.1 to 8.3) and an assessment of the implemented features and completed requirements.

[1] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

Lastly, some conclusions about the newly developed platform are drawn in Chapter 9, which ends with some insights about what could still be done in the future (Section 9.1).

2

Geometry Friends

Contents

This chapter begins by describing the origins, characteristics and importance of the Geometry Friends game. It gives a basic understanding of the game without the need to play it. It then approaches the Geometry Friends competition, its several iterations so far and the current participation process. Lastly, it mentions a few main difficulties of the current platform.

2.1 About Geometry Friends

The Geometry Friends video game was created in the scope of a master's thesis by Rocha [1]. Developed for use with the Wii Remote controller, this minimalist looking game features simple graphics but several physics concepts, such as mass, gravity, acceleration and friction. For a 2D puzzle platformer, these physical characteristics are a step up from the usual physics present in most games of the same genre. One of the key elements of this game, however, is its focus on cooperative gameplay, which will be explained later in this section.

The game currently features six different objects that can populate the map. Two of them, a circle and a rectangle, are actually the player characters. Fig. 2.1 shows a custom built map, using the in-game level editor, which is populated by all possible types of objects. These, along with their actions or effects, are now explained in greater detail.

Circle The yellow circle is one of the two possible player characters. It can move sideways (roll), and jump.

Rectangle The green rectangle is the second existing player characters. While it can also move sideways (slide), it cannot jump. However, it can morph, that is, change its shape by stretching horizontally or vertically, all the while retaining its surface area.

Black platforms These platforms are the basic, most simple platforms of the game. They are impenetrable by both characters and have a fixed position.

Coloured platforms A player object will collide with a wall whose colour does not match its own. Therefore, the circle will be able to get through yellow platforms while the rectangle will be able to go through green platforms. Similarly to the black platforms, these also do not move.

Diamonds These purple objects are the main objective in the game. They can be placed wherever in the map as long as they do not overlap with platforms. When all diamonds in a map have been caught, the player(s) win(s).

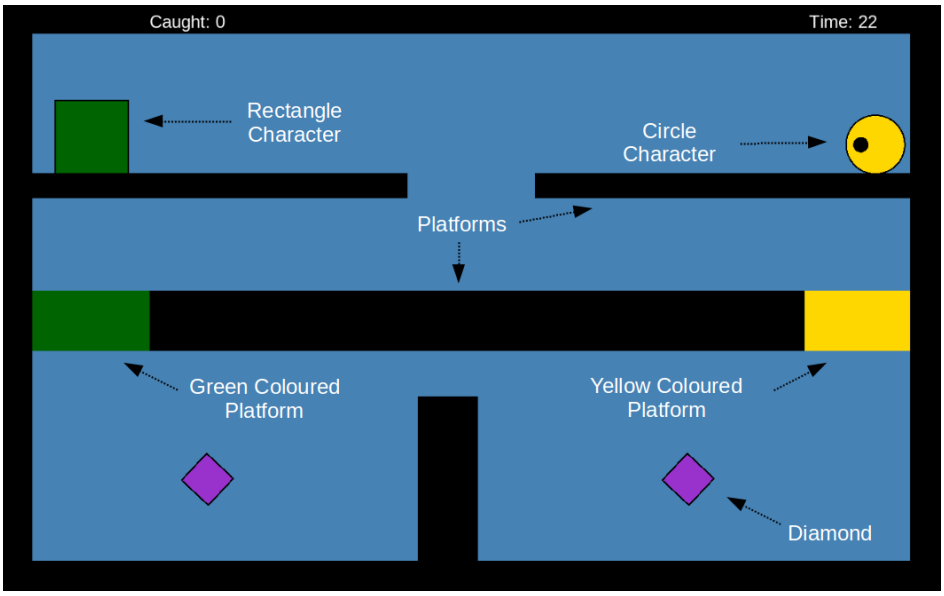


Figure 2.1: A Geometry Friends level featuring all elements of the game: the circle and rectangle players, all three types of platforms and the purple diamonds.

The goal of the game is to catch all the diamonds present in the map. Additionally, there is a time limit. The score, which will be described in greater detail in the section below, depends on both the number of diamonds caught and the time elapsed until level completion.

There are two general types of maps: single player and cooperative. Like the name suggests, single player maps feature only one of the player characters, i.e. circle or rectangle. Cooperative maps are the highlight of the game, featuring both characters. Good cooperative maps are designed so they cannot be solved by only one of the characters. This way, both the circle and the rectangle characters are forced to work together to achieve the goal of collecting all the diamonds.

If we take the same level shown in Fig. 2.1 as an example, it is easy to understand how both the circle and rectangle players are needed to complete the level, since only the rectangle agent can reach the collectible to the left, and only the circle agent can reach the collectible to the right. However, more intricate cooperation scenarios can exist. For example, placing the circle on top of the rectangle and then stretching the rectangle vertically can allow the circle to reach collectibles that are very high up.

2.2 Geometry Friends Competition

The Geometry Friends AI competition and its organizational procedures are the main focus of this project. In order to improve upon them, it is important to understand how it all currently works. Thus, this section includes a general overview of the tools given to competitors, the process by which they submit their agent's implementation, how they are ranked, and a brief summary on previous Geometry Friends

competitions.

2.2.1 Agent development and test tools

Competitors can make use of a framework which allows them to implement their own agents for Geometry Friends. An agent controls one of the player characters mentioned in section 2.1. Currently, the framework only provides a C# API. Both the framework and sample agents can be found at the competition website[1].

As is common practice while implementing an agent for a game, the developer must follow a specific structure, which implies inheriting from provided abstract classes and overriding specific methods called by the game. The agents developed using this API have access to a set of sensors and actions. The actions correspond to the available movements of each player character, as described in section 2.1. As for the sensors, they provide a lot of information which includes, but is not limited to:

- The number of collectibles (diamonds) left in the level.
- The number of platforms (any type) in the map.
- The position and velocity of all agents, including itself, as well as the position of all obstacles in the game.
- The colour of these obstacles.

The competitors are, of course, free to implement whatever algorithm they see fit. The game binaries along with five public levels per category are made available to enable local testing. There are three categories/tracks:

- single player tracks with the circle as the playing character;
- single player tracks with the rectangle as the playing character;
- cooperation tracks with both the circle and rectangle as playing characters.

An implementation can be split across multiple C# source files (.cs). However, these should be able to be compiled into a single library file (.dll). This new file is where the game will retrieve the agent's implementation from.

The framework enables the developers to test their agents in order to perfect them. Furthermore, it comes packed with several testing features which can help both the competitors and the competition organizers. These features are made available through a command line interface, where it is possible to specify options such as world (i.e. category/track), level, agent file, rendering, etc. For instance, the command

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

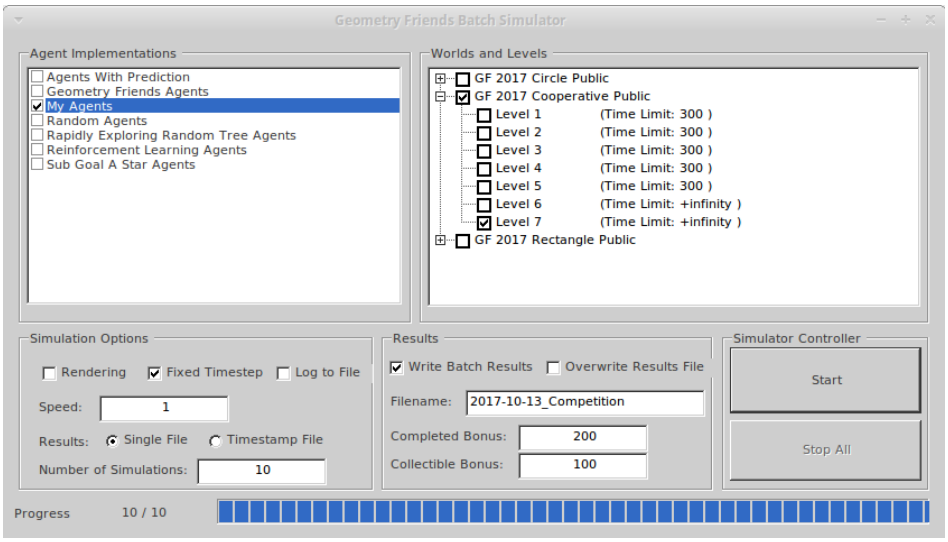


Figure 2.2: Geometry Friends Batch Simulator GUI

```
GeometryFriends.exe --no-rendering -st 1 7 --simulations 10 -a ./MyAgent.dll
```

would test (10 times) the agent compiled to `MyAgent.dll` in the current directory on the level 7 of the second (`-st 1`) world file present in the `Levels` folder, alphabetically ordered. This world *id* is zero indexed. The world file is an XML file containing the description of one or more levels. The command also specifies that the game should not be rendered, which means that the computer can focus its resources on computing the agent's algorithms and the game's core physics without having to use resources to display the game.

Another option allows the simulation to be sped up by lowering the time between each game tick or frame. However, this feature has some problems because the scores obtained from a sped up simulation differ from the ones obtained during a normal simulation.

Moreover, a Graphical User Interface (GUI) called Geometry Friends Batch Simulator can be summoned using the `--batch-simulator` argument. This GUI not only provides many of the functionalities that the command line provides, but it also extends upon them. Fig. 2.2 shows how the Batch Simulator can run several simulations, change the values of the objective bonuses (these will be explained below) and produce a more customized results file. To reiterate, both the command line arguments and the Batch Simulator are tools both the competitors and the competition organizers can use to enhance their testing environment.

2.2.2 Submission

After having implemented their agents, the competitors are ready to submit their work for evaluation. They do so by emailing[2] their solution's source code, along with the participants' identification and email contacts, team name, intended category to compete in and a description of the developed agents in the form of a two to four page report. Additionally, they can submit auxiliary files to be used by their implementation, which can be useful, for instance, for machine learning implementations.

2.2.3 Evaluation

Evaluation of the agents in each competition category covers the performance of agents across ten different levels, where five of them are the public levels mentioned above, and the other five are private levels released only after submissions are closed. This is a common practice meant to promote generalized solutions, which is why the score obtained on public levels is halved. The score given to an agent on its performance for a given level can be represented using the following function:

$$Score = \frac{1}{R}\sum_{i=0}^{R} C_i \frac{T_{max} - t_i}{T_{max}} + (B \times N_i) \tag{2.1}$$

R number of runs performed for each level. This number is currently 10.

C_i a bonus greater than zero if the agent completed the level during the run i.

T_{max} the time limit of the level, in seconds.

t_i the time, in seconds, the agent required to complete the level in run i.

B a bonus value for each collectible caught in the level

N_i the number of collectibles/diamonds caught during the run i.

During the evaluation process, a 2.50 GHz Quad Core CPU with 4 GB memory computer is currently used to run the game along with the submitted agents. This hardware, however, can be changed. Multi-threaded agents are permitted, as well as local file saving and loading for machine learning purposes.

2.2.4 Community interaction

Communication within the community is made available through an online forum[3]. This can be useful to discuss AI topics, facts about the game itself, the different available levels and general troubleshooting.

Questions can also be sent directly to the competition organizers via email.

[2] Current submission email: gfcompetition@gaips.inesc-id.pt (accessed October 13, 2017)
[3] http://gaips.inesc-id.pt/geometryfriends/?post_type=forum (accessed October 13, 2017)

2.2.5 Previous competition editions

The Geometry Friends game AI competition has been receiving submissions since its first edition in 2013, albeit there was only one entrant for that year's edition. But the competition grew, hosting yearly editions at several different events, including the IEEE Conference on Computation Intelligence and Games (CIG), Genetic and Evolutionary Computation Conference (GECCO). In the year of 2017, the competition was held at the EPIA Conference on Artificial Intelligence.

The edition of 2014 was more successful than the previous edition [2]. Held at the CIG, it featured a total of six submissions spread across the three different categories. All of the participants were from Asia, more specifically, Sejong University, the National Kaohsiung University of Applied Sciences and Osaka Prefecture University. This means that some competed in more than a single category. The South Korean entry from Sejong University, named CIBot, won all categories that year. Their solution for the circle track consisted of mapping the level onto a graph, applying Dijkstra's algorithm to plan the order of diamonds to catch based on how far away they are from the circle, and using a rule-based system for movement. As for the rectangle track the solution was similar, except they used Monte Carlo Tree Search (MCTS) for pathfinding. Lastly, for the cooperation category, both the circle and the rectangle would start by acting as if the level was single player, until no more diamonds could be caught this way. After this, they always assume that any leftover diamonds can only be caught by using the rectangle as a jumping aid for the circle.

Both Sejong University and Osaka Prefecture University also submitted agents in the following edition of 2015. Held at both CIG and GECCO, this edition saw some new submissions from Maastricht University and INESC-ID. For the rectangle track, the entry from Maastricht University took the lead. Based on the result of a master thesis by Daniel Fischer [3], this agent used a variant of the A* algorithm called Subgoal A*. As for the circle category, RRT Agent, submitted by INESC-ID, won first place. The agent's name comes from one of the techniques used, which is called Rapidly-Exploring Random Trees [4]. Another entry from INESC-ID, called PG-RL Agent that year proposed a reinforcement learning approach which successfully tackled the problem of over specialization to the public levels [5].

Similarly to the first edition of 2013, the 2016 edition, held at CIG, had just a single entrant. 2017 saw a new entry compete for the Circle Track. It was named KIT and was submitted by the Kyoto Institute of Technology. It scored higher than all of the agents submitted by the organizers.

2.3 Current Limitations

Before doing a research on good practices for running a competition (Section 3.2) and an in-depth analysis of whether the Geometry Friends competition abides by those good practices (Section 4.1), it is possible to point out a few difficulties and bottlenecks of the current platform.

Regarding the submission process, although it is quite simple for the competitor (as simple as sending an e-mail), this translates to manual labour to the organizers. Moreover, the amount of work required to handle submissions is proportional to the amount of submissions.

Even when considering some of the tools which automate part of the process of evaluating submissions, such as Geometry Friends Batch Simulator, the whole process is still bottlenecked by the human who uses the tools. Even after the evaluation is complete, the results still need to be updated and displayed manually on the website. This means that the results update phase is not immediate, that is, it may take a long time before competitors can be informed of their results.

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

Contents

This chapter starts with an overview of video game AI competitions, how they can be related to the field of AI, and why they are relevant (Section 3.1). It then presents a few good practices required to run a successful competition (Section 3.2). After this, it describes details about a few current AI competitions and, more importantly, the underlying systems that manage them (Section 3.3).

3.1 Competitions and AI

Artificial Intelligence advances can have numerous positive applications. Artificial Intelligence competitions are meant to promote these advances. Besides prizes, competitions can present baselines and benchmarks, that is, unchanging challenges that invite researchers to beat previous winners, thus presenting new and better solutions.

A popular example of researchers attempting to beat previous winners in competitions is in folk board games, such as the notorious game of chess. IBM's Deep Blue supercomputer managed to win against the world's greatest chess player Garry Kasparov in 1997. A more recent accomplishment in classical board games is the victory of AlphaGo, DeepMind's Go program, against Lee Sedol in 2016. Lee was considered one of the highest ranked Go players in the world. In May 2017 AlphaGo beat Ke Jie, who held the number one rank in the world.

The aforementioned board games present challenges with discrete and very large state spaces. Card games have similar properties, but usually have the added challenge of uncertainty, which is the case of the famous game of Poker [6].

Besides board and card games, video games are often more complex since they usually have a larger set of rules and goals, and can be continuous dynamical systems. They are also more versatile in the challenges that they can present because they are computer simulations. Several types of games are thus available in AI competitions such as the widely known Real-Time Strategy (RTS) game StarCraft [7] and the turn based battle game Pokemon [8].

While the competitions mentioned so far focus on games or virtual simulations, other competitions face real-world challenges, like the Supply Chain Trading Agent Competition [9].

Another very popular real-world area for AI advances is transportation. This area brings about several different challenges for AI, such as real-time image recognition, navigation, etc. A widely known competition on ground vehicle autonomy was the DARPA Grand Challenge [10]. Its 2005 edition was won by the famous Stanley car, Stanford's University entry for the competition. Stanford made available all the source code that went into Stanley. In 2007, DARPA held another competition known as the DARPA Urban Challenge in California, USA. The winner of that competition was Boss, a robot which featured a layered control system and was the result of a collaboration between Carnegie Mellon University and General Motors [11].

Also in the subject of transportation, the International Aerial Robotics Competition[1] promotes advances in autonomous aerial vehicles, which could have implications, for example, in package delivery to the home. Additionally, RoboSub[2] is aimed at underwater unmanned vehicles. Autonomous vehicles can have great impact on common people's lives since they provide potentially safer transportation due to perks of the digital world such as lower reaction times or precise real world measurements. Removing the need for drivers of goods transportation vehicles, such as trucks, would lower the costs for the end customer.

Modern AI techniques based on deep learning in neural networks have won several competitions and contests on pattern recognition, handwriting segmentation and recognition, human activity detection in surveillance videos, visual object detection, image segmentation, among others [12]. One of the most relevant applications of object detection in images could be in identifying tumours in images of human tissue. These biomedical automations could cut healthcare costs significantly.

Different competitions can be aimed at all sorts of goals. For instance, in sports there is the RoboCup [13] annual robot soccer competition. The Netflix Prize [14] competition falls under the category of entertainment and user experience since it challenges competitors to make user based film ratings predictions. The Loebner Prize [15] rewards the competitors that best perform on a conversational Turing test.

The immense variety of competitions offers a wide range of unique challenges. Thus, there is a lot of diversity, but let us now focus our attention on subjects more relevant to our problem.

3.2 Good Practices

This section introduces good practices for running AI competitions. This is a topic explored in detail by Togelius [16]. In the mentioned paper, Togelius starts by defending the usefulness of game AI competitions for academic reasons since they "provide fair, transparent and reusable means of benchmarking algorithms". He also states that researchers can benefit from API's specifically adapted for the task of testing AI in games. He mentions, as an example, the ability some interfaces have to run without displaying graphics, which allows for faster testing. This is currently possible in Geometry Friends.

The paper then discusses two relevant topics: cases of success and failure of competitions, and guidelines for success. As for the former, the author describes a few reasons why some competitions fail. These are:

Lack of Continuity Some competitions run very few times, and sometimes only even once. The author suggests that organizers may fail to give competitions continuity due to the high

[1] http://www.aerialroboticscompetition.org (accessed October 28, 2017)
[2] http://www.robonation.org/competition/robosub (accessed October 28, 2017)

amount of effort required in maintaining a competition running.

Stagnation Other competitions fail due to the low effort put into maintaining the competition. This includes low advertising efforts and poor sharing of information regarding previous editions, such as results and source code of previous entries.

Irrelevance Lastly, the author believes that some competitions fail due to the lack of evolution or relevance of the challenge. He defends that, for competitions to stay relevant, they must be constantly improving in terms of usability and reliability at the software and rules level. Also, the challenge presented by the competition must not be considered solved. Lastly, it cannot be so complex of a challenge that improvements upon the current baselines are not seen as feasible.

As for the guidelines, Togelius suggests that the competitions should, among other things:

- be fully transparent in terms of rules, evaluation and scoring methods, submission terms and conditions (e.g. whether submissions are to be open-sourced or not) and to keep all this information up to date;
- be accessible on a wide variety of platforms and programming languages;
- be repeated to enable the improvement of submissions and to attract competitors reeled in by previous editions of the competition;
- have a discussion group to allow the community to share and answer questions and information relative to the competition software, rules, etc.;
- have software that can run locally, in contrast to remote applications, to enable maximum performance and minimum latency;
- have a game that can be sped up to enable algorithms, more specifically reinforcement learning algorithms, that can test and learn faster;
- be easy on beginners, this is, to have samples that can quickly be set up and to have “less than a page” of instructions;
- open-source everything. This includes competition entries, so that knowledge can be easily shared and cheating better prevented, and the competition software.

Section 3.3 analyses a few competition platforms and concludes each of them with a table showing if and how the platform approaches some of these guidelines, as well as some other points. For instance, regarding the third guideline presented above which states that a competition should be repeated, the analysis will take into account factors that facilitate the repeatability of a competition, such as the automation level of submission handling processes.

3.3 Game Competitions and their Online Platforms

The competitions and contests mentioned in Section 3.1 are but a few of all the competitions that are currently running, or that ran during some time period in the past. This section describes in more detail a significant number of game AI competitions and, especially, the online platforms behind them. Each analysis presents a table showing if and how each platform meets the guidelines introduced in Section 3.2. Competitions were chosen by the amount of information that is available on them, as well as relevance to the problem.

3.3.1 GVGAI Competition.

The General Video Game AI (GVGAI) competition promotes generalized and game agnostic solutions [17]. The organizers question how much game-specific algorithms impact AI research, that is, how solutions specifically tailored for a certain game can represent any kind of breakthrough in the field of artificial intelligence. So they propose a competition to evaluate artificial agents across a wide variety of video games. These video games present 2-dimensional levels and are defined in text files according to specifications of a Video Game Description Language (VGDL) [18]. Older competitions which share a similar goal of generalizing environments and solutions include the General Game Playing (GGP) [19] and the General Video Game Playing (GVGP) [20] competitions.

The GVGAI framework allows the development of agents able to play in any game defined in VGDL [17]. The framework is implemented in Java and interacts with the developed agents using an object-oriented interface. However, the mechanics of the different games are never revealed to the player directly. That is, the player must infer characteristics of the game such as rules, physics and victory requirements during playtime.

Initially, the competition featured only a single player planning track. With time, it expanded by adding a 2-player planning track, a level generation track, a rule generation track and a single player learning track. Several instances of the competition were held at different events which include the GECCO and IEEE CIG mentioned in Section 2.2.5, as well as the IEEE Congress on Evolutionary Computation (CEC) and the IEEE Computer Science & Electronic Engineering Conference (CEEC), among others.

Competitors who wish to participate on a GVGAI competition need to sign up in the official competition website[3]. A software package is available for download which includes the framework and some sample controllers. Currently, all developed solutions must be written in Java, since that is the language the framework is written in. The competitors are instructed to name their developed Java package after their username and to zip their Java code before uploading it through the submission page, which is only available while logged in. As of 2017, Python 3.5 agents can also be developed for the single player

[3] http://www.gvgai.net (accessed November 3, 2017)

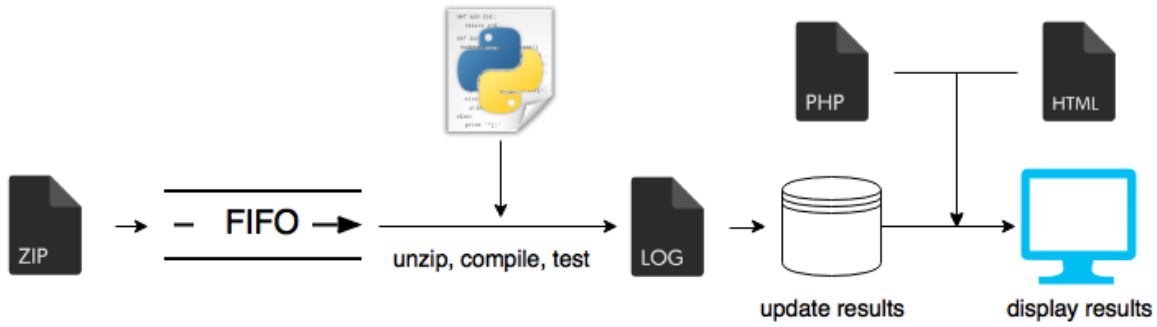


Figure 3.1: A simplified diagram which describes the back-end processes of the gvgai competition (1-player tracks only)

learning track. Users can also use TensorFlow to train their agents.

A paper on the 2014 GVGAI edition describes the back-end of the GVGAI competition [21]. It uses a server infrastructure where submitted zip files are stored into a queue. The server processes the entries of the queue as quickly as it can. This means, for instance, that the server does not follow any particular schedule where it would only evaluate entries at a certain time. To ensure that CPU time is adequately attributed to each entry, only one entry is popped from the queue and processed at a time. Once the head of the queue is popped, the corresponding zip file of that submission is automatically unzipped, compiled and tested by a script written in Python. The test is based on running the compiled controller in a predetermined set of games and levels. The same script is also in charge of reporting on errors it may find during the different phases of the entry processing (e.g. compilation errors, crashes). After the testing phase ends, a log file containing all the results is produced. A database is then updated with data based on this log file. Finally, all the resulting information from the submission is displayed in an organized and pretty view powered by HTML. Figure 3.1 summarizes these processes in a simplified diagram. The authors also reveal having used the Python and PHP languages in scripts written to aid in this process.

The process for dealing with submissions for the 2-player track actually features a dual server architecture [22]: the GVGAI server (which is most likely the one mentioned in the previous paragraph) and the GVGAI 2 Player server. The latter is in charge of running tests on pairs of controllers and reporting back the results to the former. To do this, another queue is used which is accessed frequently through a polling system. Back on the GVGAI server, the results are used to compute ratings based on the Glicko-2 rating algorithm[4]. Lastly, the databased is updated.

The tables showed in Figs. 3.2, 3.3 and 3.4 display results about the 2-player track of CEC 2017, in increasing order of specificity. Clicking a username in any of the first two tables will lead to the profile page of that user. Clicking on the *Download* hyperlink on the CEC 2017 table redirects the user to the

[4]The Glicko-2 rating algorithm was invented by Mark Glickman and measures relative skill levels. A condensed description can be found in `http://www.glicko.net/glicko/glicko2.pdf` (accessed December 30, 2017)

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

Figure 3.2: 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)*

Total | Arkanoid 2P | Butterflies | Conquer | Decks | Explore | Pacoban | Pong 2P | Prisoners Dilemma | Slide | Unbalanced

Rank	Username	Country	Description	G-1	G-2	G-3	G-4	G-5	G-6	G-7	G-8	G-9	G-10	Total
1	ToVo2	Slovenia	Description	**25**	15	12	4	18	18	18	18	15	18	**161**
2	ehauckdo	Brazil	Description	0	10	15	1	8	12	15	**25**	10	15	**111**
3	not2048	United Kingdom	Description	15	8	10	10	15	8	12	15	18	0	**111**
4	essex_acwebb	United Kingdom	Description	18	4	4	2	12	0	**25**	8	**25**	12	**110**
5	Number27	Germany	Description	0	18	**25**	6	6	2	2	0	0	**25**	**84**
6	sampleRS Sample	United Kingdom	Description	10	1	0	**25**	1	15	8	12	0	6	**78**

Figure 3.3: Part of a table showing results for all test games. The *Description* button displays a text box with information about the controller. *From http://www.gvgai.net/gvg_rankings_conf_2p.php?rg=2006 (accessed December 2, 2017)*

website homepage instead of downloading the solution's zip file. Given that the competition is already over and there is no risk of copying solutions, this is seemingly an unwanted result. Thus, this might be an indicator that the process is not fully automated yet. It is important to note, however, that for previous editions, the *Download* hyperlink does download a zip file containing the source code of a solution. This download requires a user to be logged in.

Fig. 3.3 shows the score obtained in each of the testing games, which are numbered from 1 to 10. Each one of these ten games has their own results table, as shown in Fig. 3.4, which can be accessed by clicking on a game's name above the table. The table shown in Fig. 3.3 also features a button labeled as *Description*. Clicking this displays a text box with the solution's description, as shown in Fig. 3.5. Another interesting fact about both Fig. 3.2 and 3.3 is that both include entries of sample agents that are clearly marked by a *sample* label next to their username, to allow them to be easily distinguished from external submissions.

The framework for the competition is open-source[5]. Besides presenting a FAQ/troubleshooting sec-

[5] `https://github.com/EssexUniversityMCTS/gvgai` (accessed December 2, 2017)

Total | Arkanoid 2P | Butterflies | Conquer | Decks | Explore | Pacoban | Pong 2P | Prisoners Dilemma | Slide | Unbalanced

Arkanoid 2P:

Game	Username	Country	Points	Glicko Score	Winner % (N / D)	Game Score Avg.	Timesteps Avg.
Arkanoid 2P	ToVo2	Slovenia	25	0	2% (510 / 0)	37.15	493
Arkanoid 2P	essex_acwebb	United Kingdom	18	0	2% (510 / 0)	35.84	498

Figure 3.4: Part of a table showing game specific information and scoring. *From http://www.gvgai.net/gvg_rankings_conf_2p.php?rg=2006 (accessed December 2, 2017)*

Controller description: ToVo2

A Sarsa-UCT algorithm, using TD backups and space-local Q-value normalization. Builds a tree, does not compute transpositions. Most primitive opponent model: random player, assumed as part of the environment – keeping value estimates only for own agent. No expert knowledge, no handcrafted strategies, no heuristics.

Figure 3.5: A text box with the name and implementation details of a submitted GVGAI controller. *From http://www.gvgai.net/gvg_rankings_conf_2p.php?rg=2006 (accessed December 2, 2017)*

tion in the main page of the repository, their GitHub wiki page also contains an extensive amount of information on the competition. The community can also communicate via a Google group[6].

A summary of how the GVGAI competition meets the good practices mentioned in the previous section is shown in Table 3.1.

3.3.2 Ms. Pac-Man Versus Ghost Team.

The Ms. Pac-Man Vs Ghost Team is a competition based on the popular arcade game Ms. Pac-Man [23]. Ms. Pac-Man, based on the original Pac-Man, is a 2D maze video game populated by five agents: Ms. Pac-Man and four Ghosts. Ms. Pac-Man's objective is to traverse through the maze while collecting white pills and evading wandering ghosts. A special pill, the power pill, enables the ghosts to be caught by Ms. Pac-Man. While in the original game the movement of these ghosts was deterministic, the Ms. Pac-Man game states that the ghosts movements are stochastic. To summarize, the goal of the game for Ms. Pac-Man is to maximize its score, which can be obtained by eating pills or ghosts (while under the effect of the power pill). The goal of the ghosts is to catch Ms. Pac-Man.

In the Ms. Pac-Man vs Ghost Team competition, controllers can be developed for both Ms. Pac-Man and the ghosts. More recently, controllers can be developed separately for each ghost [23], as the result of a revamp of the previous competition of the same name. Besides, the authors state that partial observability was implemented in the game engine using a line of sight (LOS) technique. LOS attempts

[6] https://groups.google.com/forum/#!forum/the-general-video-game-competition (accessed December 2, 2017)

GVGAI Competition			
Transparency	Rules	✔	Website and Github Wiki
	Scoring	✔	
Beginner Friendly	Instructions	✔	short 7 step guide
	Samples	✔	-
	Documentation	✔	Organized GitHub Wiki
Accessibility	Cross-platform	✔	JVM
	Multi-language	✔	Java, Python (only on 1P learning track)
Enables local testing		✔	-
Open-source	Game(s)	✔	VGL description
	Entries	✔	Downloadable through results table
Submission process	Sign-up required	✔	Can sign-up with GitHub
	Automatic controller packaging	✖	-
	Upload through website	✔	-
Community communication		✔	Google groups
Results display	Past results	✔	-
	Controller's description	✔	-

Table 3.1: Summary of the good practices followed (or not) by the GVGAI competition

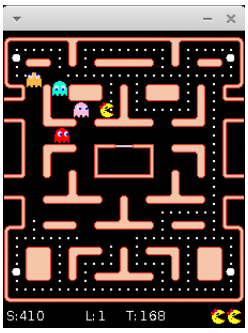


Figure 3.6: A screenshot of the game provided in the Ms. Pac-Man Vs Ghost Team competition software (starter version 0-1-8)

to make the game more challenging by limiting an agent's perception of the game in a more realistic manner, such as obfuscating parts of the maze that are far away from an agent, or that are on the other side of walls relative to an agent. Moreover, the ghosts can communicate with each other via a game enabled messaging system.

This version of the competition had its debut at the IEEE Computational Intelligence and Games Conference (CIG) in 2016. A second iteration was also held at CIG, but in 2017.

The official Ms. Pac-Man vs Ghost Team competition website[7] welcomes users to register on their competition server[8]. Besides a regular sign up, users can choose to sign up with a GitHub account. Competitors then need to download the necessary files to start developing their controllers. These

[7] http://www.pacmanvghosts.co.uk (accessed November 5, 2017)
[8] http://competitions.pacmanvghosts.co.uk/competitions (accessed November 5, 2017)

CIG 2017 Results

Ms. PacMan Results

Agent	Average Score
SubtleBattle	10260.38
giangrocker	9147.60
thunder	8861.12
ToSc	8388.80

Figure 3.7: Part of the Ms. Pac-Man Vs. Ghost Team competition results, CIG 2017 edition. There is also a similar table with the results of the Ghost Team. *http://www.pacmanvghosts.co.uk/results.html (accessed December 2, 2017)*

come in a single zip file which includes Java code in the form of a Maven project, a script, and a readme file. The first can be edited only in Java and the package must be named after the user's username on the website. This name must also be updated in Maven's configuration file (pom.xml). The second is a Bash script that packages the project into a zip file for submission purposes. The website provides instructions to setup the development environment in either Eclipse or IntelliJ, popular Java Integrated Development Environments (IDE).

In order to register for a competition, a user must sign in to the competition server and choose which track they want to compete on: the Ghost Team track or the Ms. Pac-Man track. After completing a submission form with both name and description, the user can then upload the solutions that the user wrote and packaged using the provided Bash script.

The evaluation is done in one of two different ways, depending on the amount of submissions received [23]. If the number is low enough that performing matches between all submitted controllers is manageable, then a round-robin tournament is used. If not, then a rating based on the Glicko-2 rating algorithm will be given to submitted controllers, which will serve as the criteria to select future matches. However, the final results are obtained through a small round-robin tournament where the participants are the top ten rated Ghost controllers and the top ten rated Ms. Pac-Man controllers.

In the main competition website, one can access past year results, which are displayed in a simple table as shown in Fig. 3.7.

The game code currently used for the competitions is not publicly available. However, the compiled game is (through Maven repositories), which means that developed controllers can be tested locally, without having to constantly upload different versions of a solution to the competition website. The website provides a step-by-step image-based guide covering most subjects, from registration to submission. The website also presents a table with rules and constraints.

A summary analysis on this competition is presented in Table 3.2.

Ms. Pac-Man Versus Ghost Team Competition			
Transparency	Rules	✔	Website
	Scoring	✔	Paper [23] available on the website
Beginner Friendly	Instructions	✔	step-by-step image based guide
	Samples	✔	-
	Documentation	✔	Short description of available methods
Accessibility	Cross-platform	✔	JVM
	Multi-language	✖	Java only
Enables local testing		✔	-
Open-source	Game(s)	✖	-
	Entries	✖	-
Submission process	Sign-up required	✔	Can sign-up with GitHub
	Automatic controller packaging	✔	Bash script (prepare_submission.sh)
	Upload through website	✔	-
Community communication		✖	-
Results display	Past results	✔	-
	Controller's description	✔	Only on the competition server

Table 3.2: Summary of the good practices followed (or not) by the Ms. Pac-Man vs Ghosts Team competition

3.3.3 OpenAI Gym.

Similarly to the GVGAI competition, OpenAI Gym[9] focuses on generalized problem solving. However, OpenAI Gym is not a competition. It is, as the authors describe it, "a toolkit for reinforcement learning research" [24]. This toolkit is made of two major components: an open-source software library which is meant to aid the development of reinforcement learning (RL) algorithms and a service which enables the community to compare the performance of developed agents. Both these components will be described in greater detail, but it is the latter that is more pertinent to this project.

Written in Python, OpenAI Gym's library offers a wide variety of environments (games and tasks) under a common interface. Between agents and environments, only the latter, which can be described as partially observable Markov decision processes (POMDP), are focused by Gym [24]. The authors state that this allows for a more flexible agent development with less constraints. The goal when developing an agent for Gym is to maximize the agent's performance on Gym's tasks while minimizing the amount of iterations it takes to achieve maximum performance. It is intended for the agents to be general enough so they can achieve good results across different environments, that is, so they do not overfit to a specific environment. Environments include board games, algorithmic tasks, classic control tasks, etc. As an

[9] https://github.com/openai/gym (accessed November 10, 2017)

example, a simple agent would, for every iteration in a cycle, poll the environment for an observation and a reward and, in turn, would inform the environment of the next action it intended to perform. A more recent platform built on top of Gym is OpenAI Universe[10] which is also open-source[11]. The difference, however, is that the observations obtained from Universe environments correspond to the pixels of the current state of the task, and the actions correspond to keyboard or mouse events, in an attempt to mimic a human-like interface. This also allows for a greater number of applications to be incorporated into Universe since this method of input and output is already present in most games.

Moved by the need for benchmarks to compare algorithms, OpenAI Gym also offers a service to compare agent implementations among the community. This is currently done through their GitHub Wiki page which anyone in the community can edit. Users can add information about their solutions, including obtained score, number of episodes before solving the task, a write-up and/or a video. A write-up consists of information and instructions on how to reproduce a given result, including the agent's source code. The authors make two points as to why they think this implementation sharing is important [24]:

- the scoreboards are not meant to be interpreted as a competition, but instead as a way "to stimulate the sharing of code and ideas, and to be a meaningful benchmark for assessing different methods";

- the write-ups are intended to "encourage a peer review process for interpreting results submitted by users", as submitted agents can be tested by the community to verify, for instance, that the solutions are not overfitting to a specific environment.

However, OpenAI Gym did not always use its GitHub Wiki to host scoreboards. Gym used to offer an API to upload agents to Gym's website[12] so the results would be available for the community to see. But as the website has not been maintained, the developers removed the upload functionality from the library[13] and turned their focus to the Wiki. The now removed functionality was interesting because both the agent and its submission could be made in the same Python script. A rough look at the API's old code indicates that a local HTTP client sends POST messages to OpenAI's server with results information (such as episode rewards and lengths) encoded in JSON.

The unmaintained site displays, for each environment, a scoreboard. Each entry of the scoreboard's table has a link to a page which further describes the submission with information such as authorship, agent performance (i.e. number of episodes before solving, etc), and write-up, if one exists. Below, a comments section can be found for people to discuss the submission and post reproductions they may have done with the provided write-up. It is possible to sign in to the website via a GitHub account.

[10] https://blog.openai.com/universe/ (accessed November 10, 2017)
[11] https://github.com/openai/universe (accessed November 10, 2017)
[12] https://gym.openai.com/envs/ (accessed November 10, 2017)
[13] As a result of pull request #750, https://github.com/openai/gym/pull/750 (accessed November 10, 2017)

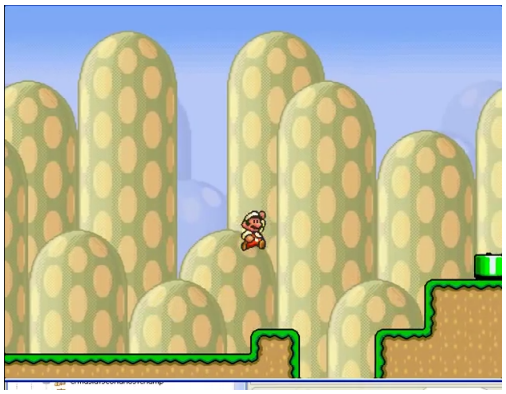


Figure 3.8: A screenshot of the video of a Mario AI 2012 Turing test track submission *https://www.youtube.com/watch?v=ZBz63FZd3Hw (accessed December 30, 2017)*

3.3.4 Mario AI.

A relatively popular video game competition is the Mario AI championship. The championship is based on the Mario AI Benchmark, which is itself based on the Infinite Mario Bros game [25], [26]. Developed and made open-source by Minecraft's creator, Markus Persson, the game itself is a clone of the popular Nintendo game called Super Mario Bros [26]. They are both 2-dimensional platform based single-player games, featuring a main character (Mario), simple gravity physics, enemies, collectibles (e.g. coins), etc. The main difference between Infinite Mario Bros and the Nintendo's original 1985 game is that the former features levels that are procedurally generated. This means that the game has the ability to offer virtually unlimited map content and play time.

Both Infinite Mario Bros and the Mario AI Benchmark (which could be viewed as a modified version of the game and a framework to enable AI research on the game) are written in Java [25]. The authors modified the game so that it could be sped up and also so there was no need to render it (similarly to what was attempted with Geometry Friends). As for the API, agents are developed by implementing an interface which allows an agent to interact with the environment. The agent receives information such as enemy position, terrain, Mario state (power-ups can transform Mario, changing its size and giving it special abilities), etc. The available actions correspond to movements of the Mario character. Originally, six buttons on a Nintendo controller could be pressed, separately or simultaneously. However, the "up" command has no effect in the game, since the "A" button is already used to make the character jump. This means that there are 32 available actions (2^5), which correspond to all the possible combinations of all the other buttons. One other aspect of the game that the authors changed was the level generator. One of the most important configuration parameters in this new level generator is the seed parameter, which allows random levels to be generated if a random number is used, or levels to be exactly recreated if the seed value is known.

The authors also took into consideration accessibility and fairness [25]. For the former, they in-

creased the number of languages that could be used to develop agents with the aid of inter-process communication technologies. Initially, they used a TCP interface for this end, but concluded it was too slow for the purpose of the competition. As a workaround, they developed a new and faster communication library called AmiCo. At the time of their writing, they claim to support Haskell and Python (besides Java). For the latter, that is, in order to improve fairness, they used a Java byte code injector named Punctual Judge which allows competitors to know the amount of byte code instructions in their controllers. This measure is machine independent, in contrast to elapsed time, which may differ depending on computational power.

Several Mario AI competitions were held between 2009 and 2012 at several different events such as CIG, the IEEE Games Innovation Conference, IEEE World Congress on Computational Intelligence, and others [25]. In the first year (2009), the competition was dominated by controllers based on the A* algorithm. After altering the complexity of the challenge, more high level planning solutions took the lead in the following year. The year of 2010 also featured more tracks other than the standard Gameplay track, such as a Learning track, Turing Test track and Level Generation track.

For the year of 2013 the competition changed, having its graphics and sounds altered to those of the open-source game SuperTux, in order to avoid legal issues with Nintendo [26]. It also changed its name to The Platformer AI Competition[14]. However, information on the Mario AI competition is still accessible on their website[15].

Other than to avoid legal issues, the authors also claim that the changes "signify a reinvigoration of the competition taking into account seveal [sic] of the lessons we've learned while running the Mario AI Championship" [25]. These lessons they speak of may refer to a reflection they make on four different objectives they laid out for the organization and rules of the competition:

- They wanted participation to be easy, which they believe was accomplished by simple framework design, availability of simple sample controllers, existence of a simple web page, open-source code, etc.
- They envisioned transparency, which they claim was enabled by forcing all the submissions to be open-source and by showing them on the website.
- They wanted the winner's victory to be unquestionable. However, after two controllers had the same score in 2009, additional performance measures were taken into consideration to better distinguish between similar quality controllers.
- They intended for different quality controllers to have highly distinct scores between each other. They feel that adding complexity to the challenge enabled them to achieve this goal.

[14] https://sites.google.com/site/platformersai/ (accessed 12 November, 2017)
[15] http://www.marioai.org/ (accessed 12 November, 2017)

Turing test track 2012
The winner is: **Vinay Sudha Ethiraj**

Name	Affiliation	Score
T. Schumann and M. Gallagher	University of Queensland	14.28
V. Ethiraj, L. Satish, and S. Johansson	Blekinge Institute of Technology	27.32
L. Heether and R. G. Reynolds	Wayne State University	4.51

Figure 3.9: Part of the 2012 Mario AI results (Turing track). *http://www.marioai.org/turing-test-track/results (accessed December 4, 2017)*

The website is not maintained at this point, as many results are missing, and many links dead. Thus, some conclusions in this analysis are based on what the authors claim in [25], instead of what is currently on the website. Some results can still be seen, however, like the ones in Fig. 3.9. The results are unordered, but the winner's name is shown just under the table's title. In this case, each submission name links to a YouTube video showing the corresponding run.

Table 3.3 presents a summary of the practices which the Mario AI competition followed.

Mario AI Competition			
Transparency	Rules	✔	-
	Scoring	✔	Explained in the rules section
Beginner Friendly	Instructions	✔	Getting started section
	Samples	✔	-
	Documentation	✔	Google site[16]
Accessibility	Cross-platform	✔	JVM
	Multi-language	✔	Java, Python
Enables local testing		✔	-
Open-source	Game(s)	✔	SVN and GitHub repository (now unavailable)
	Entries	✔	-
Submission process	Sign-up required	✖	-
	Automatic controller packaging	✖	-
	Upload through website	✖	Submission via e-mail
Community communication		✔	Google group
Results display	Past results	✔	-
	Controller's description	✔	-

Table 3.3: Summary of the good practices followed (or not) by the Mario AI competition

3.4 Mooshak - A Web Platform for Programming Contents

This section considers the problem at hand in a more general way, by assuming that it is a subset of a more broader problem: general programming contests. In other words, we recognize that artificial intelligence competitions - and, therefore, their platforms - share some properties with programming contests.

A contest is an event in which people compete for supremacy in a certain activity. The similarities between programming contests and AI competitions include a challenge equal for all contestants, digital submissions in the form of code, online participation, etc.

Following this train of thought, we take a look at Mooshak [27]. It is an online platform to manage programming contests such as the ACM International Collegiate Programming Contest. In the paper, the platform's interface is described as having four main views:

- the contestants view, which enables the submission of solutions (source code), the viewing of scores of all submissions as well as the problem description, among others;
- the judges view, which allows the intervention of judges during the contest, including answering questions and giving feedback to contestants, and more;
- the administration view for contest creators and managers to define the parameters of a specific contest, such as the problem description, the start and end dates, team data, etc;
- the public view, that is, what everyone online can have access to and see in order to track the progress of contests.

Succinctly, a contest would be created using the administration view, where the problem would be defined, teams assigned, dates established and other contest parameters introduced. The contestants would then participate by submitting their solutions using the contestants view, which would also show them their score. If necessary, judges can intervene in the process using the judges view. Lastly, anyone can spectate the contest through the public view.

3.4.1 Design Overview

According to the authors, Mooshak runs on a Linux based operating system [27]. The core functionality is implemented as an Apache HTTP server and external programs written in Tcl.

The architecture is quite modular, as the whole network can be viewed as a group of several basic components which the authors call nodes. This means that several servers can be in use in the application layer. The data is synchronized and made consistent using the rsync utility, which enables

replication across different servers. This feature is important for system backups and load balancing, and necessary due to the high number of potential contests and participants using the system.

Mooshak does not rely on RDBMS (relational database management systems) for data storage purposes. Instead, the developers decided to store objects (entities that contain data) persistently by saving them directly on the file system in the form directories and files.

3.4.2 Automated Judging

One of the most important features of Mooshak is the automated judge [27]. It allows the classification of participant's submissions. The relevant steps related to the automatic judging are:

1. setting the compilation and execution command lines for the allowed programming languages through the administration view;

2. setting the inputs and expected outputs of the execution of submissions (also done in the administration view);

3. submission by a participant of a single source file with an extension indicating one of the permitted languages;

4. static analysis consisting of data integrity checks and compilation

5. dynamic analysis consisting of one or more executions of the program using the inputs set in step 2 as standard input and comparing the output to the expected outputs

The verifications performed during static analysis include checking if the source code is written in one of the permitted languages, or if the file submitted is not too long, etc.

The verification performed during dynamic analysis marks a submission as "Accepted" if the outputs obtained during the execution of the submitted program, given the inputs defined by the administrator, matches the expected output, also defined by the administrator. Other states other than "Accepted" are permitted, especially when taking into consideration security reasons, such as "time-limit exceeded" and "output too long".

Other security measures implemented include using a user with limited privileges to execute the submissions. Other limits imposed include sixty seconds timeout for compilation, system memory caps and inability to spawn child processes. The security concern arises from recognizing that user submitted programs can be very dangerous and harmful to the system.

Strenghts	Limitations
Highly modular and expandable	Only evaluates submissions according to predefined inputs and outputs
Supports multiple contests and of big dimensions	Only allows a single source file to be submitted as a solution
Multiple HTML based interfaces to modify and participate in contests	Limited support for complex multi-step compilation and execution commands
Automated evaluation/classification of submissions	-
Informative scoreboards and submission states	-
Supports import/export of contest data	-

Table 3.4: Some strengths and limitations of the Mooshak system

3.4.3 Strengths and Limitations

To recapitulate, Mooshak is a system designed with general programming contests in mind. Contests which can be multiple and of great dimensions. It has been in development and it recently appears to have changed to use Java[17].

Some of the strengths and limitations of the Mooshak system as described in [27] are present in Table 3.4. One of the main limitations is in the evaluation process where the submitted programs are executed with predefined inputs and their outputs matched against predefined expected outputs. This methodology would not work in most of the AI competitions because usually what is being evaluated is not a specific output of a program, but how the programs handles itself in a certain environment.

Mooshak has been used not only in international programming contests, but also in educational settings, such as for evaluation of projects for programming classes.

[17] https://mooshak2.dcc.fc.up.pt/ (accessed June 19, 2018)

4

Analysis of the Current State of the GFGAI Competition

Contents

The Geometry Friends Game AI competition (GFGAI) is the subject matter of this work. The game and the current competition platform were described extensively in Chapter 2. While details regarding, for instance, the submission and evaluation processes were already given in that section, a more in-depth analysis is required in light of the good practices discussed in Section 3.2.

This chapter focuses mostly on the weaknesses and limitations of the current system in Section 4.1, but it finishes by rounding up the main areas of the system that need to be improved or even redone from the ground up in Section 4.2.

4.1 Weaknesses and limitations of the current system

The competition website[1] provides a good amount of information oriented towards new competitors, such as instructions, documentation, rules and scoring. Therefore, this is not a subject of immediate concern.

However, the website contains plenty of outdated or missing information. One example is related to the description of the submitted controllers. Submissions require that a 2-4 pages report with the description of the developed controller be zipped along with the source code. However, the website only displays technical reports dated as late as 2015. Moreover, some of the displayed technical reports are currently down, i.e. the links that are meant to lead a user to the report are dead.

As for the submissions themselves, competitors are given the option to ask the organizers not to publish the source code. This does not follow the guidelines mentioned in Section 3.2 which state that everything should be open-source. The source code of some of the controllers submitted along the history of the GFGAI competition are not available for download.

Another example of outdated information is in the displaying of results. For instance, the results of the current year of 2017 were only displayed on the website with several months of delay. A website section named Results, which is supposed to display the latest results, still displays the results of 2016. This suggests that, if all these processes were to be automated, there would be negligible delays in the presentation of results and submission information. This is important because, as Togelius remarks, new competitors can be attracted by results and implementations of the previous editions [16], so it is paramount to keep these up to date. Not only would delays no longer be a problem, but also organizational costs would be reduced.

One other case of outdated information can actually be misleading. On the submissions page[2], it is said (at the time of access) that submissions are open, while it is also said that submissions end August 8. While issues like these may seem small and can easily be rectified by an organizer, the platform could also be in charge of updating this type of information.

[1] http://gaips.inesc-id.pt/geometryfriends/
[2] http://gaips.inesc-id.pt/geometryfriends/?page_id=124 (accessed December 7, 2017)

KIT Agent

Level	Diamonds	Time (Limit) – sec.	Score
1	2 (2)	8.9 (120)	385
2	3 (3)	19.8 (180)	478
3	3 (3)	16.6 (180)	481
4	3 (3)	9 (150)	488
5	2 (2)	26.7 (240)	378
6	3 (3)	29 (240)	476
7	3 (3)	16.9 (180)	481
8	3 (3)	11 (240)	491
9	1 (1)	7.5 (120)	288
10	2.2 (3)	195.8 (240)	257
TOTAL SCORE			4203

Figure 4.1: KIT Agent results, Circle Track 2017. *From http://gaips.inesc-id.pt/geometryfriends/?page_id=1411 (accessed December 7, 2017)*

The current results tables contain a good amount of information regarding the evaluation of an agent (see Fig. 4.1). However, information about a given submission is very decentralized, since a submission's results, source code and description are in different sections of the website, that is if they are available at all. This is in contrast with tables such as the ones used by the GVGAI competition (Section 3.3.1).

Table 4.1 shows a summary regarding the GFGAI competition which is similar to the ones made to the competitions analysed in Section 3.3.

Another very important factor mentioned in Section 3.2 is the lack of continuity of a competition, which can be a result of a high amount of effort required to run an edition of a competition. In the case of the GFGAI competition, a lot of organizational effort is required. For instance, not only can it can take hours to evaluate just a few agents, but also a human being needs to be part of the process. So it is not just a question of time, but also of manpower.

Besides this evaluation phase, manpower is also required to receive the submissions sent via email by the participants, as well as to manually process and store data to later display on the competition website.

To summarize, the main flaws of the current system come down to:

- 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.

Geometry Friends Game AI Competition			
Transparency	Rules	✔	'The Competition' website section
	Scoring	✔	
Beginner Friendly	Instructions	✔	Image based step-by-step guide
	Samples	✔	-
	Documentation	✔	Knowledge base
Accessibility	Cross-platform	✔	-
	Multi-language	✖	C# only
Enables local testing		✔	-
Open-source	Game(s)	✖	-
	Entries	✖	Limited and outdated
Submission process	Sign-up required	✖	-
	Automatic controller packaging	✖	-
	Upload through website	✖	Submission via e-mail
Community communication		✔	Forum
Results display	Past results	✔	-
	Controller's description	✖	Limited and outdated

Table 4.1: Summary of the good practices followed (or not) by the Geometry Friends Game AI competition

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

In the previous section we looked at how the current system is flawed. Not flawed in the sense that it does not work, but in the sense that it is not only far from optimal, but also inefficient to the point where it could compromise the longevity of the competition. This section will briefly summarize the main areas that need improving, and thus the main focus of the design of the new platform.

4.2.1 User Centered Requirements

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.;

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);

These use cases and scenarios enable us to construct a summary of the most important requirements that the new platform should meet. This is is shown in Table 4.2. Figure 4.2 shows a use case diagram to better understand which type of user/actor is involved in which activity.

The next section expands a little bit on the requirements related to automating the submission handling processes.

Main User Centered Requirements of the New Platform	
Organizers	Organizers should be able to create new GF competition editions online
	Competition specific parameters (e.g. dates, bonuses, etc.) should be configurable
	Organizers should have the ability to produce baseline submissions and corresponding results for each competition instance to guide the participants
	The Geometry Friends game version used in the future competitions may be changed
	All processes spanning from a participant's submission to the display of the its results should be done automatically without human intervention (see Section 4.2.2)
Participants	Participants should be able to access all competitions open for submissions at any given moment, online
	Competition specific details should be made transparent to the participant
	The GF framework should be available for download on the website, including sample agents and the competition levels
	Participants should be able to upload their submissions (and technical report) online
	The results of a submission's automatic evaluation should be made transparent for the participant
	The participant may submit multiple times for a single competition, but only the last submissions should count for the participant's final score
	The participants should have the same agent development freedom they have had so far, that is, agents may use third-party and platform specific dependencies
General Public	Anyone may access the details and results of both unfinished and finished GF competitions
	All of the submitted agents' source code and technical reports should be available to everyone (finished competitions only)

Table 4.2: Some of the main requirements the new platform should meet with regards to the three main groups of people which will be interacting with the new platform.

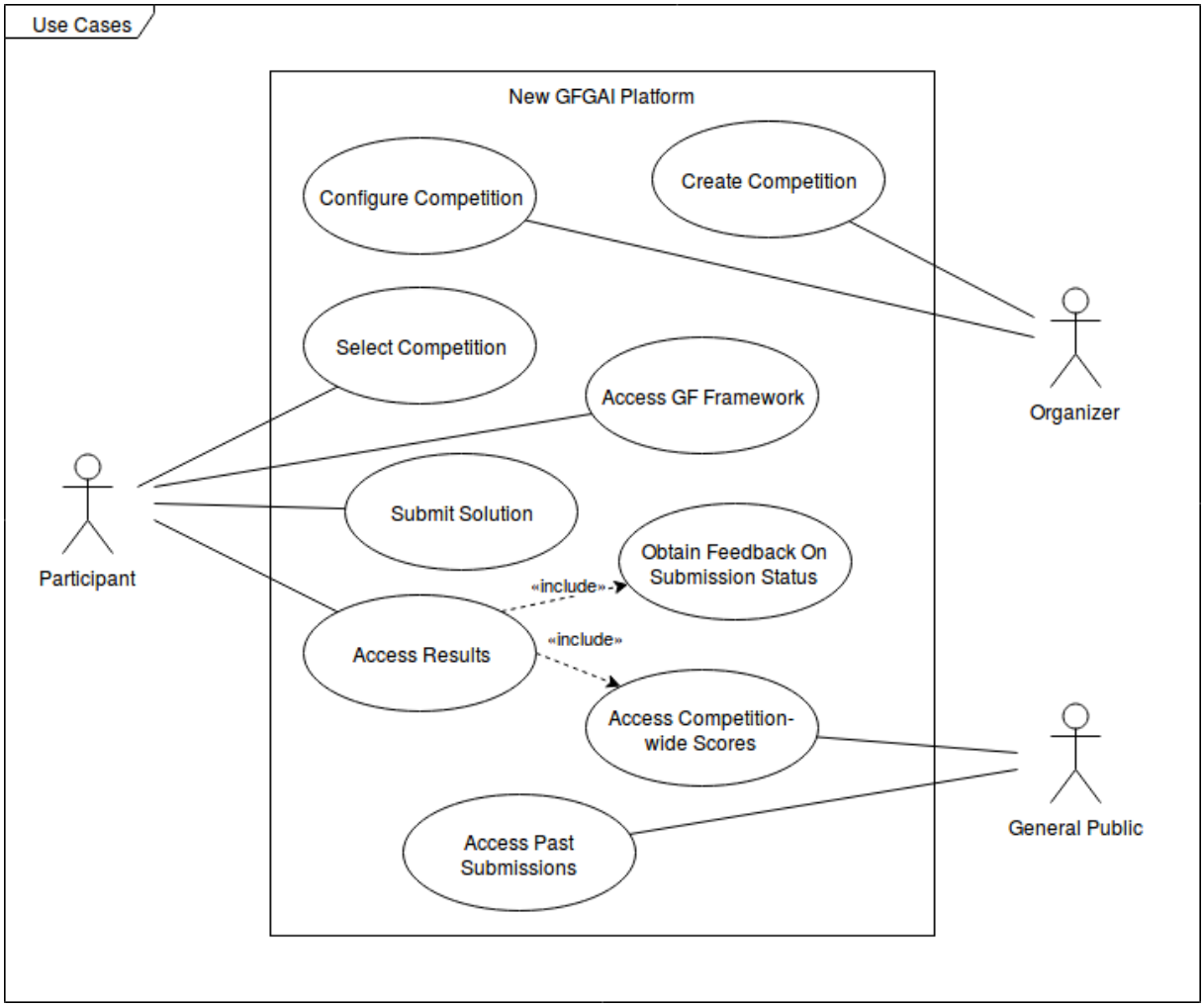


Figure 4.2: Use Case Diagram showing the elementary interactions between the different stakeholders and the new platform.

4.2.2 The Automation Challenge

One of the biggest and most important requirement mentioned above was the automatic processing of submissions in the competition organization section of Table 4.2. Therefore, it is important to provide a deeper understanding of the underlying tasks that occur after an ordinary submission for a competition.

1. a submission zip file containing C# source code and a technical report (PDF) are sent by the participants, which up until now was done via email as described in Section 2.2.2;
2. the submission zip file is extracted, revealing the C# source code;
3. the extracted source code is coupled with the game files and are then compiled, producing a single '.dll' file containing the agent itself;
4. the agent is executed/simulated in the competition levels, often multiple times per level;
5. the raw results are extracted and the actual score is computed according to the level parameters and a specific formula, as described in Section 2.2.3;
6. the results of the participant are displayed on the competition website, including details of the agent's performance on each particular level.

This is a simplified view of the tasks at hand. It gets more complex when taking into account factors such as the ability of a participant to re-submit an agent for the same competition, effectively overriding its previous submission. Another complicating factor lies in the undefined number of participants, which can range from a couple to hundreds. Lastly, we must take into account concerns of security and fairness.

While the individual tasks mentioned above require significant work to automate, one greater challenge is to seamlessly join them together to give the appearance of a single, black-box process.

4.2.3 Quality, Security and other Non-Functional Requirements

The new Geometry Friends platform should also meet several requirements that do not directly impact the user experience. These include requirements typical of most online systems, such as security requirements.

Table 4.3 aggregates some of the main quality, security and other non-functional requirements that the new platform should meet.

As a clarification, we define zombie submission files as files belonging to submissions that were overridden/replaced by newer submissions. This is a consequence of the requirement which allows participants to submit multiple times to the same competition. Since there is no requirement which specifies a need for these zombie submission files, these could be considered a memory leak.

Non-Functional Requirements	
Data Security	The system shall be protected against man-in-the-middle attacks
	Persistent data shall be protected against tampering attacks such as SQL injection
	Data sent to the system shall be properly validated
	Sensitive user information (e.g. passwords) stored in the system must be protected against theft
	Competition management data such as competition configurations and game versions shall not be tampered with by non-organizers.
Submissions	Incoming submissions must have a size limit which will depend on competition configuration
	Zombie submission files shall be removed from the system
	The system shall be safeguarded against malicious submission code
	Each submission shall be evaluated under similar conditions
	The system shall support evaluating a single submission for up to 500 minutes

Table 4.3: Quality, security and other non-functional requirements that the platform should meet.

The requirement which states that each submission shall be evaluated and similar conditions does not take precedence over the requirement which states that submission code may be platform specific (see Section 4.2.1).

The duration of 500 minutes for the estimated ceiling of a submission evaluation assumes a maximum of 5 minutes per level, for 10 levels and 10 simulation per level. Values used are based on previous competition editions.

5

New Platform Overview

Contents

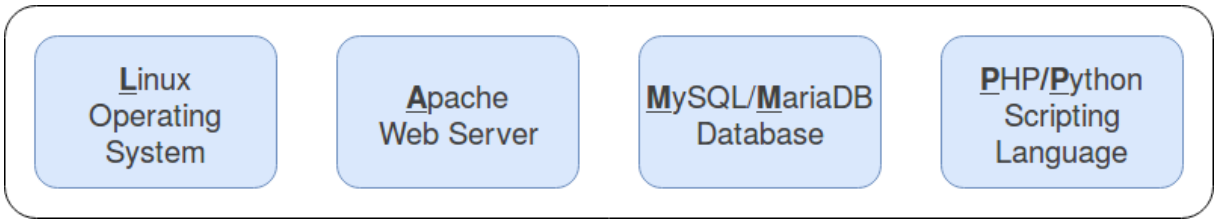


Figure 5.1: The LAMP stack, a popular way to structure web applications

This chapter introduces the new platform that was developed to address the issues of the current Geometry Friends competition platform explored in Chapter 4.

5.1 A Web Application From Scratch

The Geometry Friends competition is meant to be open to the international public, requiring interaction from users anywhere. Similarly the other platforms studied in Chapter 3, the new platform was developed as a web application.

A LAMP stack was chosen as the basic structure of the new platform. This is a well established concept consisting of four main components (also see Figure 5.1):

- A Linux based operating system
- An Apache HTTP web-server
- A relational database management system (RDBMS) such as MySQL or MariaDB
- A scripting language for back-end purposes, such as PHP, Perl or Python

The RDBMS used was actually MariaDB and the language was Python. MariaDB is a community-developed fork of the open-source MySQL, so they are mostly compatible and interchangeable. However, the choice for Python was mostly due to existing personal proficiency with the language.

5.1.1 Why from scratch?

All the studied platforms in Section 3.3 are mostly built specifically for the games and competitions they run. As said in Section 3.4, even Mooshak, a platform designed to support more generic programming competitions, is not flexible enough to accommodate for the needs of the Geometry Friends competitions. For instance, Mooshak only evaluates submissions based on their output matched with expected outputs [27]. Geometry Friends evaluates agents based on more non-discrete outcomes.

Furthermore, the Geometry Friends is a rather unique game and therefore hard to generalize. Factors such as the language it was written in, its specific directory structure and the complexity of the

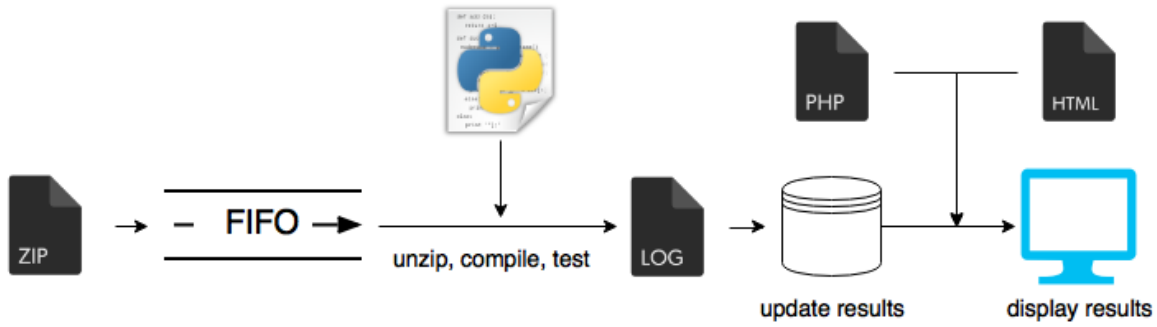


Figure 5.2: A simplified diagram which describes the back-end processes of the gvgai competition (1-player tracks only)

compilation and evaluation result in a set of processes that are not trivial to automate. As such, creating a new platform to manage and automate most of these processes would need to be a highly custom endeavour.

While many characteristics of the game itself and the competition were discussed in Sections 2.1 and 2.2, several other limitations and nuances were brought to light during development which favour the need for a customized platform. These will be detailed throughout the following sections and chapters as each part of the new platform that dealt with those special cases are explained.

5.2 System Overview and Structure

Similarly to the GVGAI back-end (Figure 5.2, repeated here for the reader) mentioned in Section 3.3.1 where there is a clear distinction between the submission handling processes and the website interaction, it made sense for the new GFGAI competition platform to be composed of two distinct components.

This two-component architecture makes it easier to divide different responsibilities and tasks of the platform. A simplified visual representation of the whole system, which is meant to reside in a single machine, can be seen in Figure 5.3. The two main components are:

Website the component which is meant to be interacted with, both by competition participants and administrators and is fully described in Chapter 6

GFHandler the background process which is in charge of handling submissions and more. Chapter 7 describes this component in more detail

The GFHandler component is the component that solves the automation challenge as described in Section 4.2.2. The website is also relevant during the first and last steps of the automation challenge, i.e. helping the participants send submissions and displaying the final results.

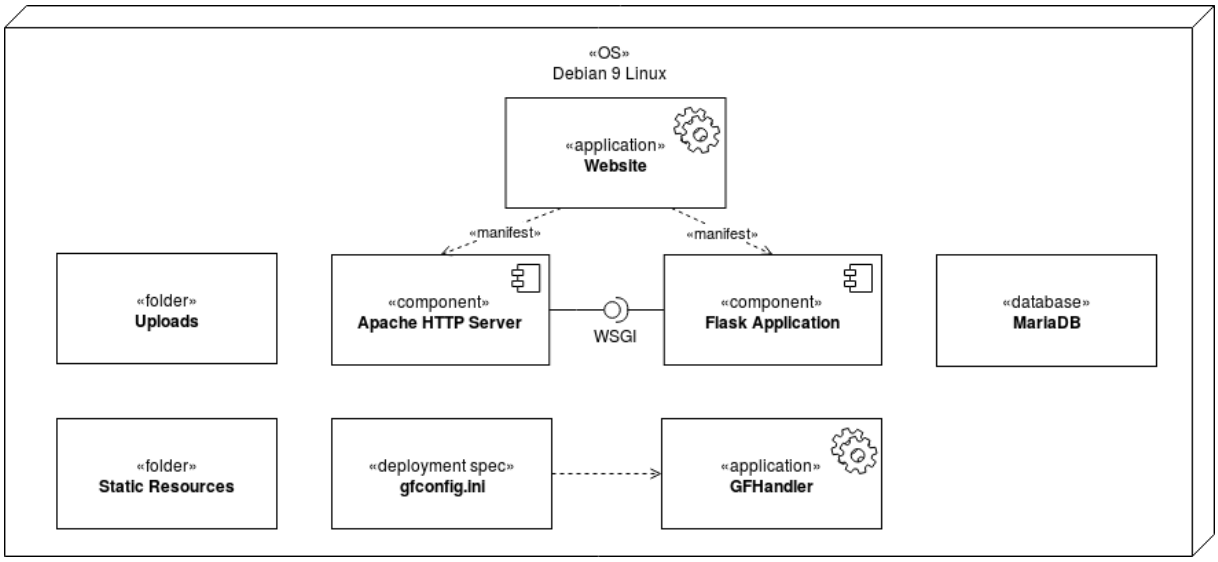


Figure 5.3: Simplified deployment diagram containing some of the most important components of the deployed system.

Although the two components are independent in the sense that they do not communicate with each other directly, they do share information. Both of the described components connect to the same MariaDB database and use the same tables. An overview of the database is given in section 5.3.

The two components share files and directories of the file system. An example of files shared between the two components are the submissions themselves, which the participants are meant to upload using the website component and are then handled automatically by the GFHandler component. Lastly, they also share a single conda[1] environment, as they are both written in Python 3.

More technical details of some of the components, files or directories used in the implementation and that are not covered in this document can be seen in the Appendix B.

5.3 Database Schema

As previously mentioned, the database is a MariaDB database used by both of the components introduced above. Its role is to allow the persistent storage of competition related data (user information, results, etc.) and to act as a common point of data sharing between the two main components.

This section explains a broad and generalized conceptual model of the entities and their relationships, while many details regarding their implementation and how they are used by the system will only be given in the following chapters.

The entities and fields created try to mimic the logic of the current system. Conceptually, the core entities of the system are:

[1] https://conda.io/docs/ (accessed August 9, 2018)

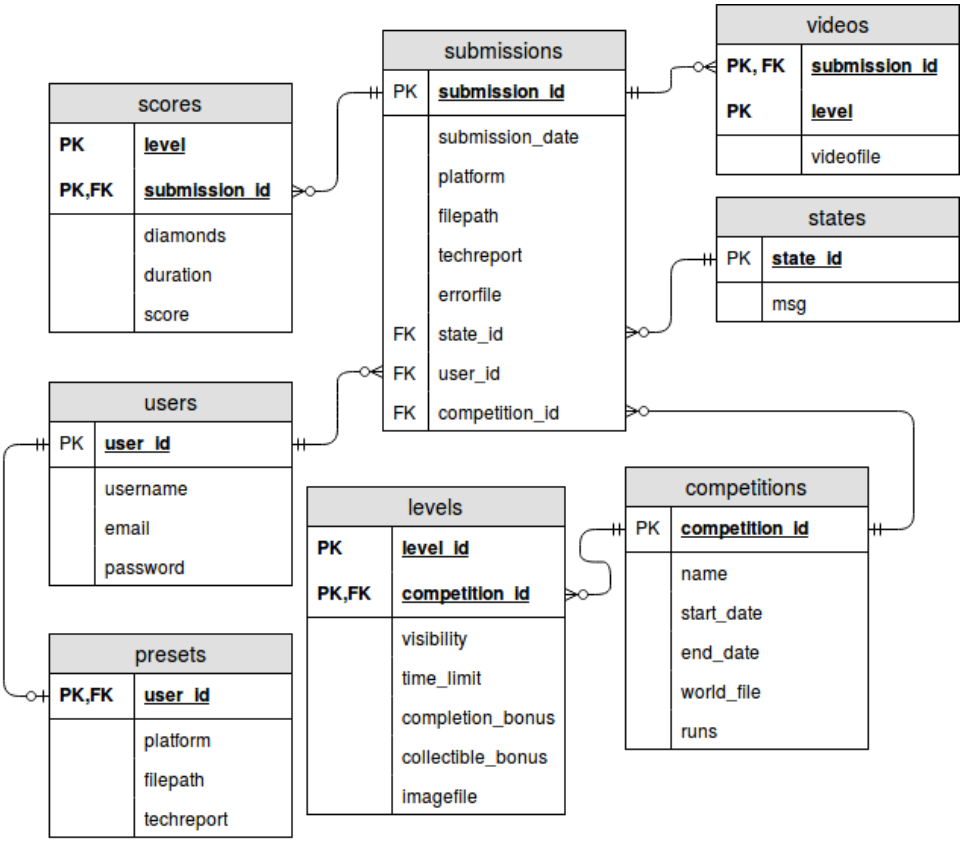


Figure 5.4: Database model in Crow's Foot Notation. Some fields are omitted for the sake of brevity.

User the participants information, not only allowing the personal information to be grouped together, but also to be easily associated with other entities;

Competition the competition information, as introduced by the competition organizers. Here, a competition only refers to a single specific track/category (circle, rectangle or co-op category) so each competition only has one type of levels;

Levels the maps where the agents are evaluated for a certain competition. They are numbered (starting at 1) and may be private or public. Private levels are only made available to the public after a competition ends. A level also has a simulation time limit and other parameters used in the scoring formula;

Submission information regarding a single submission from a certain user to a specific competition (and under this model, a specific category). This is the cornerstone of the system since it is at the centre of many relationships;

Score the performance of a single submission on a single level. This information is what is displayed to the participants on the website.

These core entities, along with a few other entities and the relationships between them all are shown in the database diagram shown in Figure 5.4. The entities that are not explained here will be discussed in the following chapters.

6

The New GFGAI Competition Website

Contents

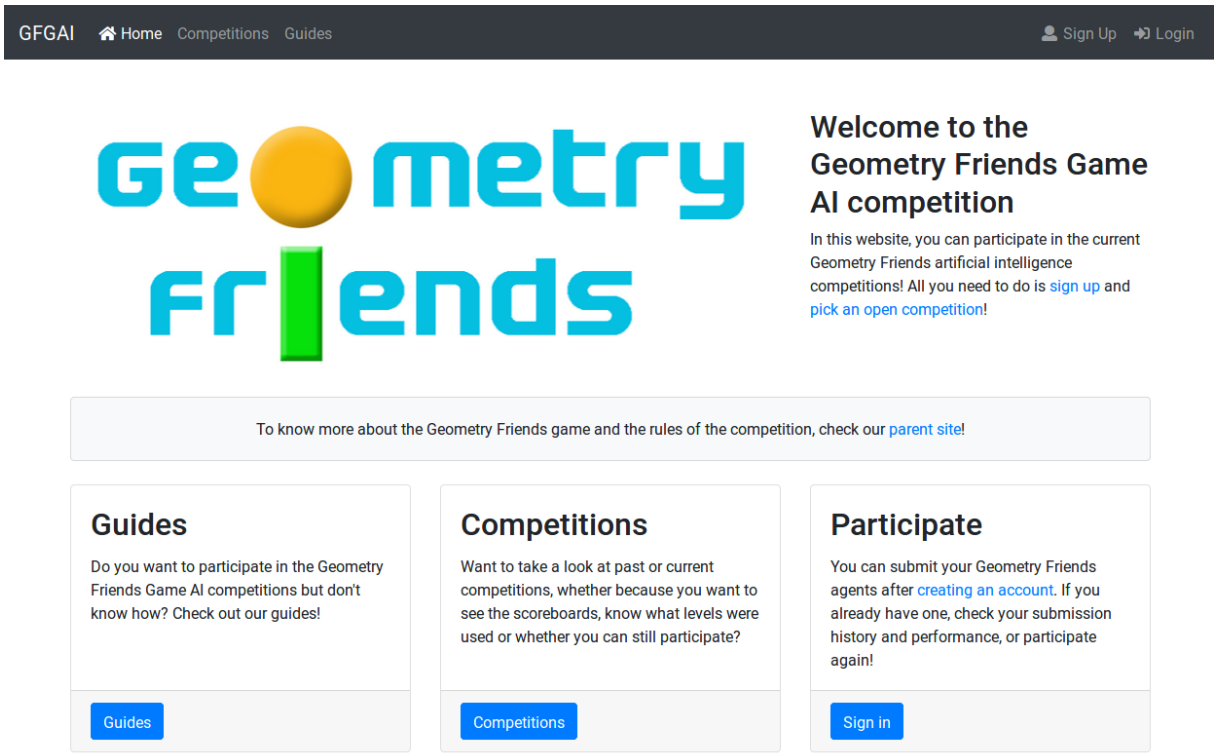


Figure 6.1: Home page of the newly developed website.

The website is one of the two major components mentioned in the previous chapter. It is the part of the new system which interacts with competition participants, organizers/managers and the general public.

We define an instance of a competition, or from now on simply a competition, as having only a single category. Which means that if a Geometry Friends competition with three tracks (circle, rectangle or co-op) needs to be created, one actually needs to create three separate competitions. This is because each set of levels is defined by a single XML file (see Section 2.2.1), and in the model of the new platform, it made sense to associate one XML file to one competition, as it facilitated a lot of processes.

The home page of the new website can be viewed in Figure 6.1. Refer to Appendix A for all the website screenshots that are not included in this chapter.

6.1 Account Based System

In contrast with the current system where the submissions are made via email (as explained in Section 2.2.2), the new platform implements an account based system. While the email system requires participants to send their information (team name, contact details, etc.) along with their submission, an account based system allows these personal details to be defined only once and in a more standardized

Submission Status/State	Meaning
new	The submission has not been picked up by the GFHandler component yet
processing	The submission is being evaluated, meaning its state may change soon. Even without changing state, the score may increase as the GFHandler evaluates the submission on each level of the competition.
handled public/all	Whether the submission has been completely and successfully processed on only the public levels or all public and private levels, depending on whether the competition is still ongoing
* error	The GFHandler may detect and report errors with the submission, such as compile errors, extraction errors, etc.

Table 6.1: The possible submission states that a user may check to know what is happening to their submission.

way.

As such, this personal data can be accessed when there is a need, for example, to contact the participant or to pair a submission with its owner's name in a scoring table.

The information collected from a participant during sign up (see Figure A.1) on the new website is as follows:

- a unique username, which is meant to be the name that will appear on the scoreboards and is the equivalent of the team name;
- a unique email, for contact purposes;
- a password required for log in and that can be reset in case the user forgets it;
- affiliation, which is the participants were asked to send when submitting via email.

To easily access this kind of information, the new platform also features a profile page for each user. All details mentioned above (except for the password) are publicly available. After logging in (using the form in Figure A.2), a user may see, on his/her own profile (see Figure A.3 in Appendix A), a section which includes their submission history, i.e., details on the state and results of their own submissions to any competitions. Table 6.1 gives an overview of these states, which represent the status of a submission as it is handled, in real time, by the GFHandler component (we postpone to Chapter 7 a more detailed discussion of the GFHandler component).

Special administrator user accounts exist to access management interfaces. These special users (only one exists currently) can only be created manually on the database, since they are not predicted to increase in numbers in the future. A specific column on the user's table discriminates between the

different user types. This feature meets the security requirement which states that non-organizers shall not be able to tamper management related data.

Lastly, a simple password reset feature was implemented in case a user wants a different password. An email is sent to the user containing a new randomly generated password.

6.2 Management Interfaces

The management interfaces are pages of the new website designed to meet the requirements of the competition organizers (see Table 4.2), with the exception of the submission handling automation. As such, only the owners of a special administration account may access these pages.

These can be seen as interfaces analogous to the "administration view" of the Mooshak platform (see Section 3.4).

This set of competition management interfaces is one of the most important sections of the new website since, as was previously mentioned, the improvement of the processes related to the organization of the GFGAI competition is a main focus of this work.

6.2.1 Competition Creation

As previously mentioned, we are now considering a competition as a single category/track. As such, a single XML world file containing the description of all the levels of the new competition must be uploaded during competition creation. This cannot be changed afterwards for the sake of consistency throughout the competition's lifespan.

A description may be given to the competition when it is created (see Figure A.4 in Appendix A). This description allows the organizer to reveal more details about the competition, such as associated event, cash prizes, and others.

The dates introduced when creating a competition must represent sometime in the future. Then, a competition may be in one of three states:

Upcoming the competition's start date has not been reached yet. During this stage, the competition is only visible to administrators because its parameters may still be changed in a "Manage Competition" page.

Open the start date has been reached and the competition in now visible to the general public. Parameter editing is limited during this time, as it not possible to edit parameters of the competition levels. This is to maintain consistency of the results obtained for the whole duration of the competition.

Finished the ending date has been surpassed. No competition parameters may be changed during this stage. The final results should be available at this point.

A fourth special stage exists temporarily between the Open and Finished stages, in which the system evaluates all the received submissions for a given competition on its private levels. In place of the competition state, it simply shows "Evaluating Private Levels".

6.2.2 Editing Competitions

The "Manage Competition" page (see Figure 6.2) allows several competition related information to be edited (if permitted in the competition's current state, as mentioned above). They can be general competition parameters, such as:

- the number of runs/simulations per level, defaulting to 10, the same value used by the organizers in the previous competition editions;
- maximum submission size, defaulting to 512 MB, which is a value larger than the size of the largest submission ever received (over 300 MB);
- a time-out period, in minutes, which specifies the minimum amount of time between two successive submissions from the same user and to the same competition. This is useful because users may be otherwise encouraged to repeat submissions unnecessarily since the results are not made available instantly after a submission.

The editable parameters may also be level related, such as:

- the visibility of a level - private or public. The purpose of level visibility is the same as before, that is, private levels are unknown until the competition ends and are used when evaluating the final score of submitted agents. It helps prevent overfitting solutions.
- collectible and completion bonuses, parameters used when applying the scoring formula (see equation 2.1 of Section 2.2.3).

In order for the organizer/administrator to easily identify which levels he/she is editing, the competition management page includes pictures of each level. These images are generated dynamically by the GFHandler, and the implementation of this feature is explained in greater detail in Section 7.4.1. The level previews need to be generated dynamically because all that is uploaded during competition creation is an XML file containing the description of the levels, and so the pictures don't yet exist at this point.

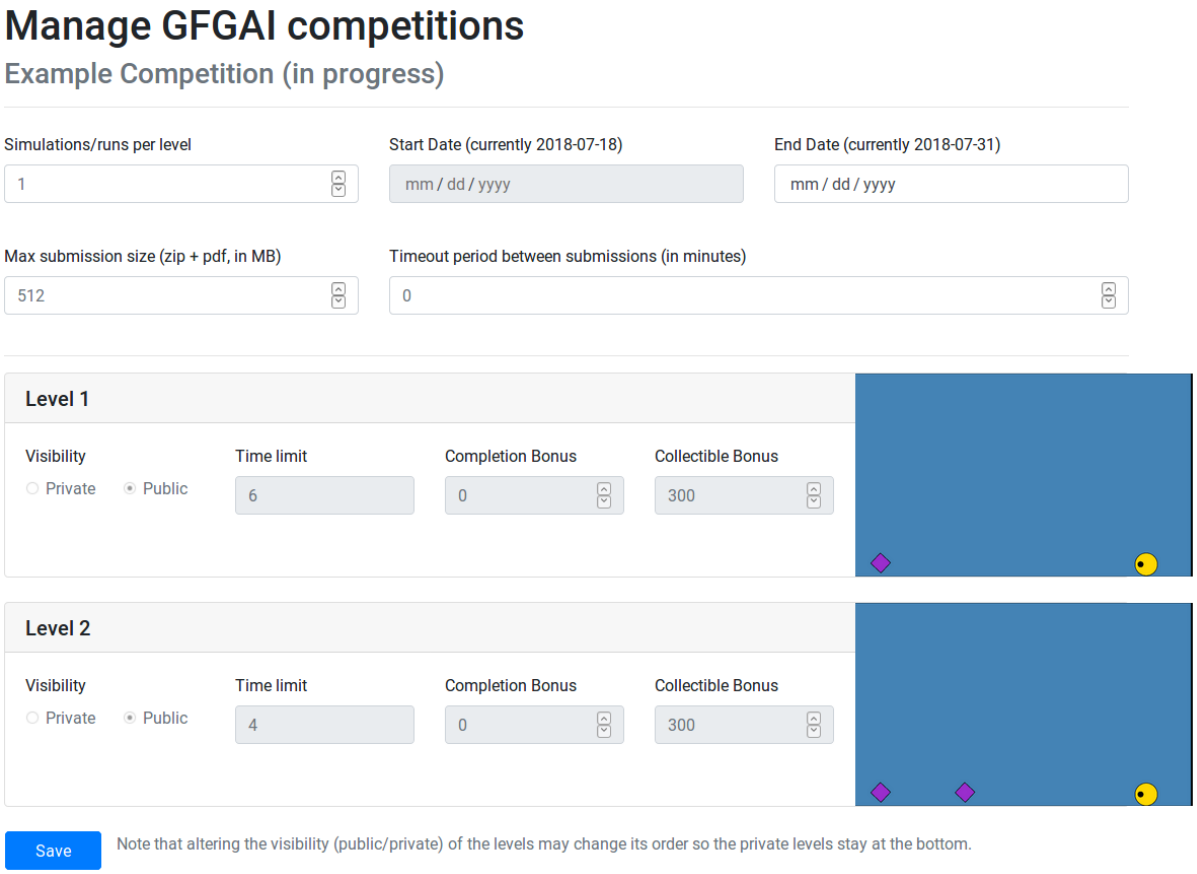


Figure 6.2: Management interface to edit competition parameters. Different form fields can be disabled/enabled depending on the stage of the competition.

6.2.3 Presets

So far, the Geometry Friends Game AI competition has tried to include baselines in some of its editions by sometimes having the organizers re-submit agents submitted in previous editions. This also required the organizers to possess the actual submission code of the submission they were trying to reuse, every time they wanted to reuse it.

The new system includes a feature to introduce baselines: presets. It allows participants to have some sort of reference of a minimum score value to achieve, as well as more examples of possible implementations.

A preset is implemented as a special type of user (see Figure 5.4). It has a name, which is what will appear on the scoreboards, and it may "submit" its solution to different competitions. The "submissions" a preset can perform are always the same, i.e., a preset represents a specific and unchanging solution.

Only the organizers may manage, i.e. create (see Figure A.5) and run (see Figure A.6), presets. In the interest of testing out different competition parameters before a competition opens (i.e. when a competition is in the Upcoming stage and all parameters may be edited), presets may be ran on Upcoming competitions. Because level specific parameters may be changed while presets have been submitted, it makes sense to eliminate these submissions every time the parameters change. This avoids inconsistent results, but forces the organizer to resubmit/rerun the presets in order to obtain results consistent with the new parameters.

Lastly, the participants should be able to easily distinguish between normal submissions and these baseline preset submissions. This is why presets appear differently in the new website's scoreboards, i.e. with a greyed out background and a "(baseline)" string added to its name.

6.2.4 Updating the Game

It is important to note that the new system keeps a copy of the Geometry Friends game to use when automatically handling submissions. However, the game itself may change over time, especially if it ends up becoming open-source as the guidelines discussed in Section 3.2 suggest. This means that the systems needs to have a way to update the current version of the Geometry Friends game it is using, which is a requirement we defined back in Table 4.2.

The management interface allows an organizer (someone with an administrator account) to upload and replace the Geometry Friends game version used by the system (see Figure A.7).

To avoid scenarios where different submissions belonging to the same competition are evaluated using different versions of the game, it is only possible to update the game when there are no open competitions.

A competition organizer also has access to a version history. This is a list containing timestamps of

when new versions were introduced into the system. It helps to keep track of which game versions were active when certain older competitions were open. The old versions themselves are not stored inside the server.

6.3 Participating in Competitions

To participate in a Geometry Friends Game AI competition using the new platform, a user may pick an "Open" (see Section 6.2.1 for details on competition states) competition which they may be interested in. This is done in a page which contains a list of competitions ordered from "Upcoming" (visible to administrators only), through "Open" and finally to "Finished". Thus, a non-admin user will see "Open" competitions first, ordered by the ending date.

Each competition in the list provides several options (see Figure A.8) related to participation. The name of the competition may be clicked to open a page containing more details on the competition (see Figure A.9, such as the parameters set by the organizer/administrator.

A participant may then download the competition package, a ZIP file containing:

- the Geometry Friends game;
- a sample agent to help get started;
- the XML world file containing the levels of the specific competition;
- two packaging scripts (explained in Section 6.5.3).

An option of just downloading the XML file is available in case the participant already has the other files (e.g. from having participating in a previous competition).

This ZIP is constructed dynamically on request to avoid keeping several different and redundant copies of the game on the server, as the game occupies a significant amount of disk space (more than 70 MB).

If the user is logged in, they may submit their solution (ZIP) and technical report (PDF) through a simple form (see Figure 6.3). These files are the ones that have been requested up until now, but this way users do not have to email the organizers every time they want to submit. They also do not need to repeat their personal/team details in the submission, because we now have the profile page for that.

On the submission form, participants have to make it explicit whether they developed their solutions to work under Windows or under Mono (the .NET framework available in operating systems such as Linux and MacOS). The reason for this is explained in Section 7.2.

Participants may then track the status of their submission in real time under their profile (see Section 6.1 and Table 6.1). They may also look at the global scores and compare against other participants under the scoreboards.

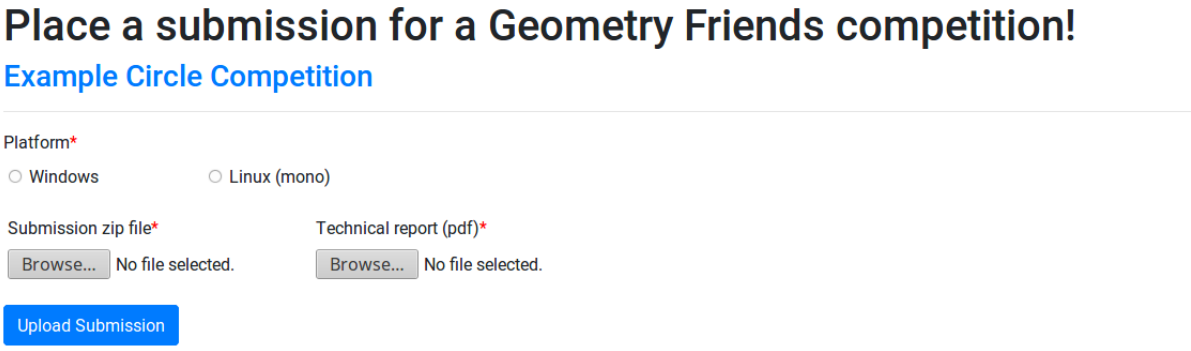


Figure 6.3: The submission form.

6.3.1 Scoreboards

The participants (and the general public) may see the results of each competition on a scoreboard, as shown in Figure 6.4. There are several interesting aspects about this one, when compared to the old one (see Figure 4.1):

- the total score stands out on each row, as it is the main evaluation metric for each submission;
- the level specific scores are hidden by default and may be expanded by clicking on the button on the far left of each row (Figure 6.4 shows that the submission by "Ricardo" is expanded, while the one by "Ranhoso" is not). This enables all the entries to be packed together which facilitates comparison of scores;
- a lot more information is accessible from each row of the new scoreboard, such as the participant's profile (accessible by clicking on the username), their submission's source code and their technical report.

This is a great step in increasing the accessibility of information of the competition. To prevent cheating, both the submission's source code and technical reports are only downloadable after a competition ends.

Additionally, one may watch videos of the submissions performing on the different levels (using the buttons in the "Video" column of the expanded sections) after having requested them (using the buttons in the "Video Request"). This feature is better explained in Section 7.4.2, but videos allow anyone to better understand how a specific solution or implementation works on the specific levels of a certain competition (see Figure A.11).

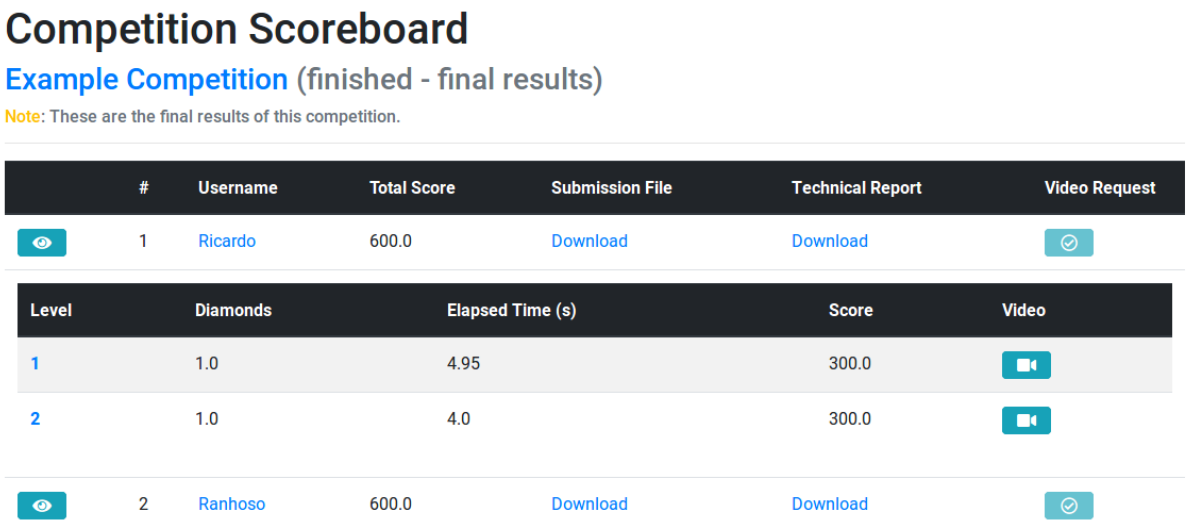


Figure 6.4: Two entries on the scoreboard on the new website. The first entry is expanded to reveal level specific results.

6.4 Guides

Currently, the new platform's website has not merged with the original competition website. As such, much of the previous knowledge base was not repeated to the new website. Instead, the new website includes several links to the old website, when needed.

However, given that the way to participate has changed, new guides needed to be created. To facilitate a quick learning process, some of the new guides are image based, similarly to the ones from the Ms. Pac-Man Versus Ghost Team website.

The four guides are:

- an image based guide which shows how to participate, including the implementation of a simple agent and submission;
- a tutorial which also shows how to participate but in a step by step video;
- a guide detailing how agents are run in the GFHandler;
- an image based management guide, available only for users logged in as administrators, that teaches how to create and manage competitions, presets, etc.

6.5 Implementation and Challenges

As previously mentioned, the website was written in Python, mostly due to personal proficiency. A technology named Web Server Gateway Interface (WSGI) allows the existence of web frameworks

written in Python.

Since this was a project of significant dimensions, the website itself was designed to be quite minimalistic, so the front-end development would not take too much time relative to other important components of the new platform. Having this in mind, a minimalistic and simple Python web framework was chosen: Flask[1].

As for the front-end framework, one of the most popular ones was chosen: Bootstrap[2]. No specific reason was responsible for this choice, other than it being widely used and having good documentation.

A special "uploads" folder inside the website directory is the destination for several files, including participant's submissions, XML world files uploaded by competition managers and presets. These are given unique names or placed in folders with unique names, to avoid collisions. This folder is also used by the GFHandler component, for instance, when it needs to fetch a submission in order to evaluate it.

The technical report/manual (see Appendix B) contains details about versions used, and more.

6.5.1 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. However, the system must be secured against possible attackers.

All the security requirements (see Section 4.2.3) not mentioned so far in this chapter were met in the following way:

- the HTTPS protocol is used by the server, which improves the security of data transmitted between client and server by preventing
- correct usage of the Flask-MySQLdb's [3] package prevent the system from interpreting and executing SQL queries attackers may write on text fields (SQL injection); eavesdropping or communication tampering (man-in-the-middle attacks).
- both client and server side form validation were implemented, ensuring that the data received remains consistent with what is expected by the system;
- passwords are hashed before being stored in the database using PBKDF2 with SHA-256, which enhances protection against password theft.

[1] http://flask.pocoo.org/ (accessed August 7, 2018)
[2] https://getbootstrap.com/ (accessed August 7, 2018)
[3] https://flask-mysqldb.readthedocs.io/en/latest/

6.5.2 Reordering the Levels

Inside the XML world files that the competition managers upload for each competition are descriptions of different Geometry Friends levels which the game can interpret. An important fact for the explanation that follows is that the levels are numbered, in ascending order, starting from 1. The new system is built on the assumption that this particularity of the world file is always true.

The new management interface allows these levels to be configured as public and private, after the file has been uploaded. Before a competition ends, participants should only have access to the public levels, which they should be able to obtain through the website, by downloading either the whole competition package or only the world file (XML) itself.

This means that the XML file that the participants download may only include the descriptions of the public levels. Therefore, when a participant requests the XML file, the file would need to be modified so the XML elements pertaining to the private levels were eliminated.

If the public and private levels were to be intercalated, the numbers of the levels present in the modified XML files would be gapped. Although not very intuitive for the participants, this would not represent such a great problem. However, due to a limitation of the game, it crashes when using an XML file containing gapped level identifiers. While in the future the developers responsible for the game may change this, an alternative solution was implemented.

As soon as the visibility (public/private) of the levels is changed by a competition manager, the corresponding XML file in the server is modified with the aid of the `xml.etree.ElementTree`[4] so the levels are reordered, placing the public levels before the private levels. Then, a copy of the XML file is made, but containing only the public levels. This copy is what is sent to the participants in case they request the XML world file. Because of the reordering, the levels sent are ordered, without gaps, and starting at 1.

A MariaDB procedure was also created on the database to reorder the levels in the corresponding table, to maintain consistency with the XML file. This is why if a competition manager changes the visibility of the levels of a competition (which can only happen before a competition starts), the level order may change.

6.5.3 Packaging Scripts

Participants may compress and submit their whole solution folder if they want. This includes unwanted files such as the game itself, because the GFHandler will use its own copy of the game. Using a local copy of the game prevents cheating, since participants could, in theory, submit an altered version of the game.

[4] https://docs.python.org/3/library/xml.etree.elementtree.html (accessed August 7, 2018)

The size of each submission may vary, as some implementations are more complex than others. However, excluding the game and possible external dependencies, the source code of a submission occupies between 30 to 350 KB. The game itself occupies over 70 MB.

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. 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 Bash script uses the "zip" command line tool usually included in most distributions. As for the Windows script, a PowerShell script was initially developed which attempted to use the "Compress-Archive" utility, but it proved extremely difficult, if not impossible, to achieve the desired functionality. In the end, the Batch script was written to use the 7-Zip[5] external program. Participants are informed in the guides and in the script itself that 7-Zip must be installed and present in the PATH environment variable for the script to work.

[5] https://www.7-zip.org/ (accessed August 7, 2018)

7

GFHandler - The Submission Handling Program

Contents

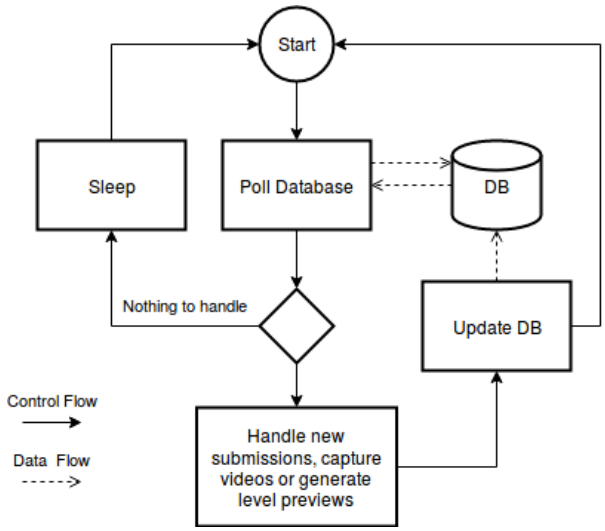


Figure 7.1: A flowchart containing the core elements and procedures of the main polling loop of the GFHandler program.

The GFHandler is the most complex program of the new Geometry Friends competition platform. Is is meant to act behind the curtain, working autonomously mainly to handle submissions (hence its name). It is also capable of other actions, such as generating level previews and submission videos.

By handling submissions we mean to automatically evaluate submissions from the moment they are uploaded to the moment when the final results appear on the website. This is includes a series of steps that previously required human intervention, and therefore, organizational costs.

7.1 The Main Loop of the GFHandler Program

A new submission has two important characteristics for the GFHandler: a zip file with the submission's contents (C# source code containing an agent's implementation) and a new entry on the database's "submissions" table. All this data is the results of interactions and uploads through the new website.

As a program which is in charge of handling the submissions, the GFHandler must always be expecting new submissions to arrive at any given time. As such, the GFHandler works by constantly polling the database, at regular intervals, for new and unprocessed submissions.

Each submission entry in the database possesses all information required by the GFHandler program (see Figure 5.4), such as the path where the uploaded submission was stored to, the current status of the submission (e.g. if it has been processed already or not), and more.

A MariaDB view was created to facilitate the process of fetching new submission information. Then,

the GFHandler only has to query this view, and does so using mysqlclient[1], a fork of MySQLdb[2] which allows for Python 3 support.

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 7.1 shows a simplified diagram of the main loop of the program.

7.2 Simulation Environments

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

- a standardization of the mentioned processes, which 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[3] is sent.

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. Docker[4] is an example of container technology. However,

[1] https://pypi.org/project/mysqlclient/ (accessed August 1, 2018)
[2] https://github.com/farcepest/MySQLdb1 (accessed August 1, 2018)
[3] Malicious submissions are submissions which contain code that may harm the system it is executed in.
[4] https://www.docker.com/ (accessed August 1, 2018)

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, which allows us meet multiple non-functional requirements (see Section 4.2.3. 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[5] is an open-source virtualization product that possesses all the necessary features (e.g. snapshots), and thus, it was used in this work (version 5.2.12). The problem now was deciding what virtual machines to use and how to have them do what is required for the submission handling system.

7.2.1 Choosing the Operating Systems

The majority of Geometry Friends agents are written either using Microsoft .NET Framework (under Windows) or the Mono[6] 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. Lightweight systems boot and work faster by using less resources, which also means that these resources (e.g. RAM, CPU) are free to be used by the submission handling processes.

The Linux distribution chosen was Ubuntu Server 18.04, a recent operating system based on the popular Ubuntu distribution, but even lighter since graphics related packages are not provided by default. A clean installation only uses between 50 and 60 Megabytes of RAM.

Finding a lightweight Windows operating system is not easy. Microsoft Nano Server[7] is an installation option for Windows Server 2016 which is lightweight and meant for headless systems. It was created as a way to compete with the popular, widespread lightweight Linux based server distributions.

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

However, Nano Server 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.

7.2.2 Setting up the Virtual Machines

The machines were created to use two CPU cores and 4 GB of RAM, but this can be easily reconfigured since they are virtual machines. Both machines have more than 5 GB of disk available for the participant's submission alone, which is an order of magnitude greater than the size of the largest submission ever received (between 300-400 MB).

A user called "gf" was created on both virtual machines (Ubuntu Server 18.04 and Windows Server 2016). The users were created without administrator privileges (at least without entering a specific password) for security reasons. Privileged access is not require to run the Geometry Friends game.

While Ubuntu features a package manager (Apt[8]) which facilitates the installation of the required software, Windows does not. However, there are third party package managers for Windows. We used Chocolatey[9].

As for Ubuntu Server, the most relevant packages that were installed are: `unzip`, `openssh-server` (see Section 7.2.3), `mono-complete`, `xvfb` (see Section 7.6.1). MSBuild[10], Microsoft's build tools which enable the building of submitted agents through a command line interface, is installed along with mono. At the time of setup, the most up to date stable versions were Mono 5.10.1.47 and MSBuild 15.6.0.0.

PowerShell possesses a command to extract archives, so a package analogous to `unzip` was not required. OpenSSH server and `visualstudio2017buildtools` were installed using Chocolatey. MSBuild was included in the latter package. However, its bin directory had to be placed on the PATH environment variable manually since Chocolatey did not appear to take care of this automatically. The most up to date MSBuild version installed at the time using the method above was MSBuild 15.6.85.37198, meaning that this version belongs to the same major release as the one installed on the Ubuntu Server machine.

Regional settings were also changed in Windows, especially the float separator settings. When using the Portuguese float separator (comma) instead of the more international dot, the CSV files were not being read correctly.

Lastly, an empty directory named "submission" was created under the home directory of the "gf" on both VM's. With everything ready to receive and handle submissions, a VirtualBox snapshot of each machine was generated. This way, restoring to the snapshot ensures the VM returns to this clean state.

[8] https://wiki.debian.org/Apt (accessed August 2, 2018)
[9] https://chocolatey.org/ (accessed August 2, 2018)
[10] https://msdn.microsoft.com/en-us/library/dd393574.aspx (accessed July 26, 2018)

7.2.3 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. The VirtualBox machines, which were configured to use Network Address Translation (NAT), needed to have their SSH port (port 22) forwarded so it could be accessed by the host system.

Because each machine has a user named “gf” (see Section 7.2.2), and because SSH keys were setup on both machines, the SSH command to access and run commands are almost the same. The only exception is the path used, which on the Windows machine includes the drive letter.

7.3 Handling the Submissions

Now that we have seen how the submission environments are setup, this section describes how the submissions are handled.

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 (see section 4.2.1 of the manual in appendix B for a complete list of possible states).

After detecting the existence of an unprocessed submission (by looking at its status), 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. Because there is no practical way for the virtual machine to notify the host system that it is completely booted up, the GFHandler sleeps for a predefined amount of time, selected to ensure that system can safely boot.

After a while, the GFHandler connects to the VMs via SSH, which allows the GFHandler to issue commands inside the virtual machines. 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.

To process the submissions, the GFHandler:

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 and any XML existing world files) 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, 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 2.1 by parsing the results from a game generated CSV file;
7. powers off the VM.

see Figure 7.2 for a simplified sequence diagram which gives an overview of these steps

The virtual machines do not possess a copy of the Geometry Friends game (step 4). This 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 5, a DLL of the submitted agents is generated with a randomized and unique name. This ensures that when we execute the game, the agents simulated are the ones we just compiled, instead of pre-compiled agents which may have been included in the submission files.

After steps 3 and 4, there will only be one XML file containing the only levels we need in the submission folder (index 0 on the levels folder).

The Geometry Friends command line interface (CLI) can receive a number of runs to be performed during the simulation. However, if the game crashes on any run, the simulation would stop. So, on step 6, in order to allow the simulation to evaluate an agent on the remaining number of runs in the event it crashes on a run, the GFHandler calls the Geometry Friends CLI command R times, specifying only one run per time.

Given all this and the Geometry Friends command shown in Section 2.2.1, we can run the submitted agent with the following variation (for each level, and for each run):

```
GeometryFriends.exe -st 0 <level> -a Agents/<compiled_agent>.dll
```

Also 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.

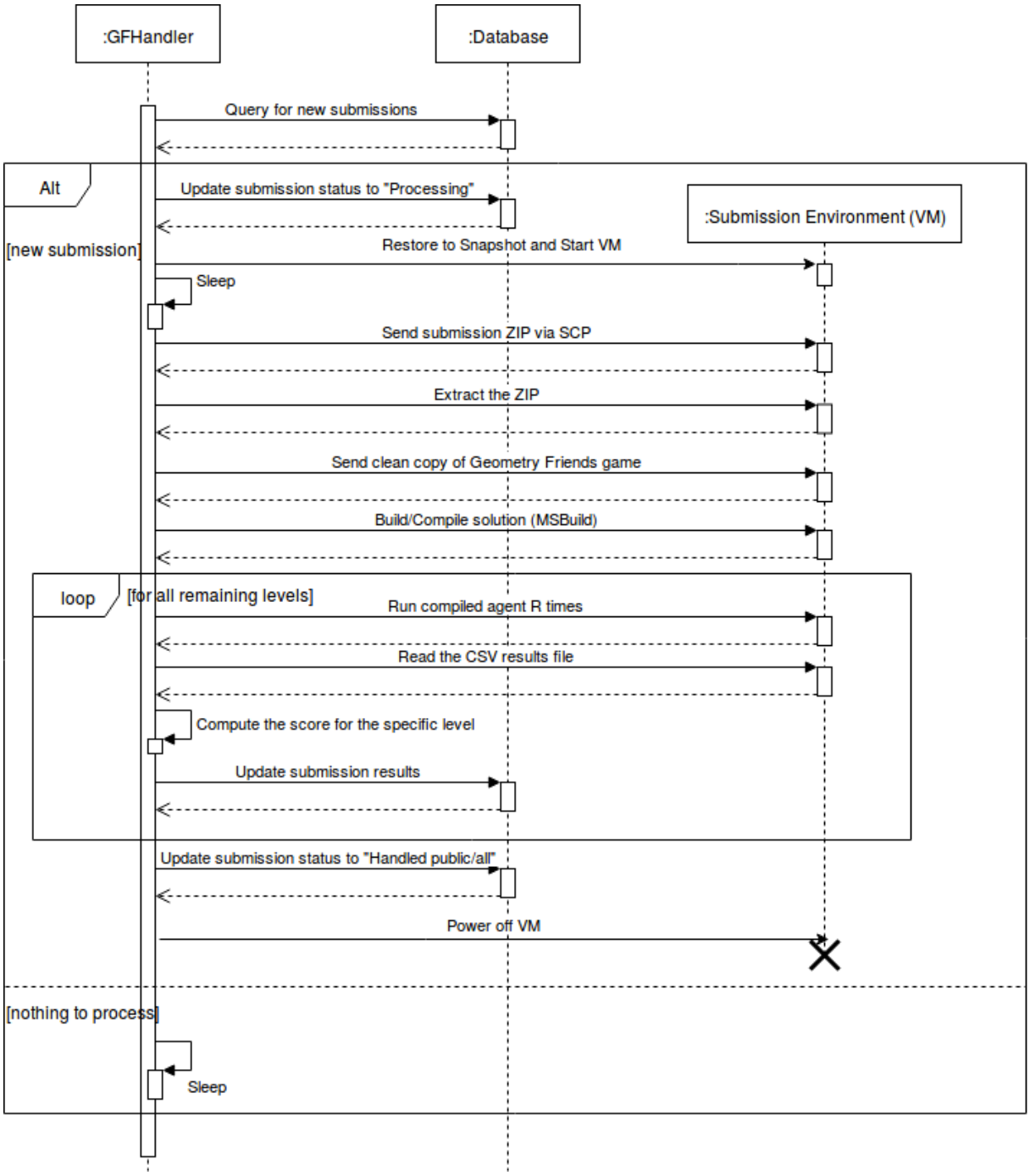


Figure 7.2: Sequence diagram containing the main steps of a single iteration of the submission handling process.

7.3.1 Main Data Structures and Classes of the GFHandler

Now that we have taken a rough look into how the GFHandler program interacts with the different components of the system in order to handle submissions, we can dive a little deeper into its actual implementation.

The GFHandler is written in Python 3. The GFHandler itself is a single Python package, containing a sub-package inside called "envs" which contains all the classes that represent the submission environments.

Figure 7.3 shows a class diagram of the GFHandler program, containing the most relevant classes and the relationships between them. Classes such as the `VideoMaker`, `VideoSubmission`, `Screenshooter` and `NewCompetition` provide functionality for features which we will discuss in Section 7.4. So for now, we are going to focus on all the other classes.

Having in mind a normal submission handling process, the main idea is that a `GFHandler` object, which is the main object in charge of dictating the control flow of the entire program, has a `Submission` object and a `SubmissionEnv` (submission environment) object. A submission is evaluated inside a submission environment.

The actions executed inside a submission environment correspond to external commands (not in Python), such as compiling a C# solution. To facilitate this, a `SubmissionEnv` extends a class called `CommandDispatcher` which provides a very simple wrapper for the `run` method of the `subprocess`[11] module. Most methods inside a submission environment use the wrapped `run` method, which was named `run_cmd`. This wrapper specifies an option to the original `run` method which guarantees that injection cannot occur on these external commands.

Just like the `subprocess run` method, our `run_cmd` returns a `CompletedProcess` instance, an object which contains information about the external command executed, such as standard output and standard error streams, and exit codes. Listing 7.1 shows how one of these external calls can appear simple in the GFHandler's code.

Listing 7.1: Method in charge of running the external command which compiles a solution

```
def compile(self, timeout=None):
    """Compiles the solution using MSBuild in the root of the submission
    directory.
    """

    cmd = ["msbuild", "/p:AssemblyName=" + self.assembly_name]
    return self.run_cmd(cmd, timeout, cwd=self.submission_dir)
```

[11] https://docs.python.org/3/library/subprocess.html (accessed July 30, 2018)

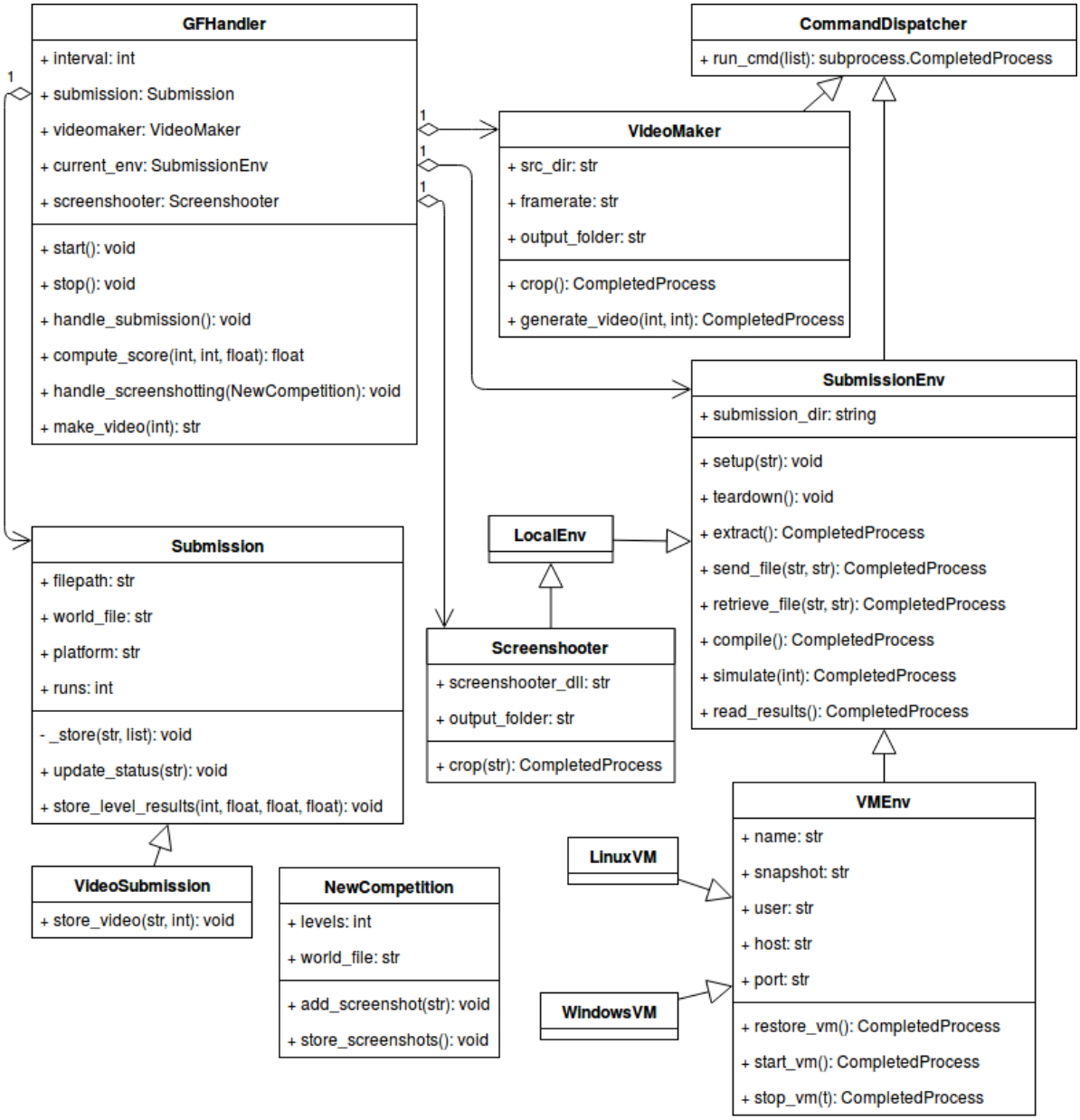


Figure 7.3: Class diagram containing some of the most relevant classes, attributes and methods of the GFHandler python module.

The run_cmd method is overridden several times in the different subclasses shown in the diagram. For instance, in the VMEnv class, the class in charge of handling submissions inside the virtual machines, the run_cmd method is overridden to make sure every external command is executed through SSH to the virtual machine. Under the WindowsVM environment, the run_cmd method is overridden to ensure that all external commands are executed in Windows PowerShell.

Other methods were also overridden when more shell specific commands were needed. For instance, the extract method is overridden in the WindowsVM environment to call PowerShell's Expand-Archive command.

The VMEnv also has methods to start, stop and restore a virtual machine to a given snapshot. In contrast to the VMEnv, the LocalEnv allows submissions to be handled on the host system. While the LocalEnv ended up being useful for the creation of another environment whose use will be described later, it can also be used in a testing scenario to avoid the usage of the heavier virtual machine environments. However, the LocalEnv environment is vulnerable to malicious submissions as it does not benefit from the isolation of virtual machines, and cannot run submissions which depend on Windows.

In order to avoid hardcoded static parameters, an external configuration file is used to store the configuration of the GFHandler program. This includes, but is not limited to, the polling interval of the main loop (in seconds), paths of important files and directories, virtual machine names, ports and snapshots. All the configurable parameters are thoroughly described in section 4.1 of the technical report/manual in Appendix B.

7.3.2 Error Handling and Logging

As is normal practice in complex programs, the GFHandler keeps a log file of what it is doing. In fact, there are two log files: the file which the GFHandler is logging to until it reaches 2 MB, and a backup log file of the previous 2 MB of logs. After reaching the 2 MB of logging, the current log file replaces the backup log file and an empty log file is created. This means that there will be 4 MB of logging, at most, at any given time.

The subprocess Python module used by the GFHandler captures both the standard output and standard error streams, providing more details of a possible error on a failed command. The following is an excerpt of the GFHandler log file detecting a critical error when trying to send a clean copy of the game to a virtual machine submission environment (the necessary folder was purposely removed for this example):

```
[10/07/2018 16:18:30] [INFO]: Sending a fresh copy of the game to the environment
[10/07/2018 16:18:30] [INFO]: Sending /home/ricardo/Documents/thesis/gfclean/GeometryFriendsGame/
to virtual machine path gf@localhost:C:/Users/gf/submission using port 2224
[10/07/2018 16:18:31] [ERROR]: Encountered a server error while running the command: ['scp', '-P', '2224', '-r',
'/home/ricardo/Documents/thesis/gfclean/GeometryFriendsGame/', 'gf@localhost:C:/Users/gf/submission']
```

```
stdout + stderr:
/home/ricardo/Documents/thesis/gfclean/GeometryFriendsGame: No such file or directory
```

The GFHandler program is expecting errors at any stage of its execution. 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.

After encountering an error and informing the database of what happened to the submission being handled, the GFHandler performs clean-up procedures (e.g. shutting down the VM) and advances to the next submission. The only exception occurs when encountering runtime errors during game simulation, where the GFHandler simply skips to the next run of the 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 simulation 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.

In the event the GFHandler needs to be restarted, it can detect which submission it was handling before, and how many levels it already simulated it on, all by looking at the database. This allows it to resume operations without having to repeat many actions. This is another perk of updating the database in real time, in the middle of submission handling processes.

7.4 Capturing Level Previews and Submission Videos

Up until now, only the submission handling processes were detailed, which are the most important activities of the GFHandler. However, when not busy computing scores of new submissions, the GFHandler may do additional things. This was already hinted in Figure 7.1.

7.4.1 Level Previews

Some pages of the new website show pictures of the different levels of a given competition, as can be seen in some of the screenshots in Appendix A. However, only a simple XML file containing the description of the levels is given to the system when competitions are created. This means that the pictures need to be generated dynamically.

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.

7.4.2 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 7.3
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[12]
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.

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

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. An additional command line parameter is now available on the Geometry Friends CLI, named `--capture`, which accomplishes step 2 above.

7.5 A Step Into Open-sourcing the Geometry Friends Game

As Section 7.4.2 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 3.2 mentioned, open-sourcing is considered a good practice to run game-based AI competitions.

7.6 Additional Implementation Details and Challenges

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

7.6.1 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 CLI 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 it also encountered the same errors as the Ubuntu Server.

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. This is because the simulations stop automatically once they are completed.

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 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[13].

7.6.2 What the GFHandler Does When Simulation Hangs Indefinitely

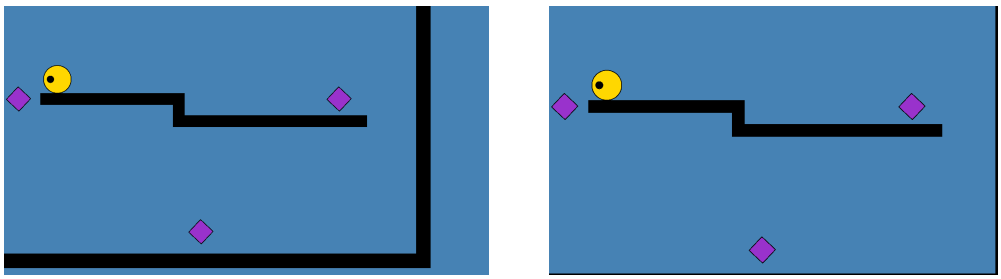
The majority of the errors the GFHandler is capable of detecting during the handling of a submission are detectable because commands which fail return an error code, which the Python module the GFHandler uses to run external commands, subprocess, can detect.

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. To test this, an agent was developed that slept indefinitely during its setup.

The solution was to use timeouts. Under the Ubuntu Server VM, commands were run in Bash via SSH. Hence, the "timeout"[14] 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

[13] https://www.x.org/archive/X11R7.6/doc/man/man1/Xvfb.1.xhtml (accessed July 30, 2018)
[14] http://man7.org/linux/man-pages/man1/timeout.1.html (accessed July 30, 2018)



(a) Before cropping **(b)** After cropping

Figure 7.4: Cropping a screenshot taken inside the Ubuntu Server 18.04 VM. The original shows areas outside of the game boundaries to the right and bottom.

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.

7.6.3 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.

7.6.4 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 [15] image manipulation tool suite are used for that end.

The result of this crop procedure can be seen in Figure 7.4.

7.6.5 Handling Overridden Submissions and Automatic Clean-up of Outdated Submission Files

Section Section 4.2 introduced the requirement which states the system should be able to receive multiple submissions from a single user, for a single competition. According to this same requirement, only the latest submission counts for the official score.

The new website marks submissions with an "overridden" state if new ones are placed by the same owner to the same competition. In order to save space on disk, submission files belonging to overridden submissions are deleted (this is a requirement mentioned in Section 4.2.3).

Given that submissions take a significant amount of time to be processed, it is possible that new submissions from a single user may appear while the user's previous submission is still being handled by the GFHandler. Deleting submission files that are currently in use results in a critical error by the GFHandler, which should never happen.

This issue was tackled in both fronts, website and GFHandler, like this:

- when a new submission is received on the website, mark previous submissions from the same user and for the same competition as "overridden". If the previous submission does not have the "processing" status, then its files (ZIP and PDF) are eliminated;
- at several stages during the submission handling of a given submission, the GFHandler checks if the submission's state has changed to "overridden". If this happens, an exception is raised causing the GFHandler to halt, delete the overridden submission's files and go back to the main loop.

7.6.6 How GFHandler Runs on the Background

Given the polling architecture chosen for the GFHandler program, it is required that the program always be running. Because the machine may be restarted for maintenance or other reasons (e.g. unplanned crashes), it makes sense to have the operating system control the execution of the GFHandler so it can restart the GFHandler at boot. This prevents the need for manual intervention.

[15] https://www.imagemagick.org/script/index.php (accessed July 30, 2018)

It is important to recall that the new platform was designed to be deployed (and was eventually deployed[16]) on a Debian based operating system. It was found that the feature which best met the desired requirements was `systemd`[17] service manager. A `systemd` service is often a program (or sets of programs) which run in the background and whose execution is controlled by `systemd`. Popular examples of programs which are executed as services include databases (MySQL, MariaDB, etc.), the Apache HTTP web server, and many others.

Listing 7.2: GFHandler service unit file.

```
[Unit]
Description=Geometry Friends Submission Handler service
After=mysql.service

[Service]
Type=simple
WorkingDirectory=/home/gf/gfhandler
ExecStart=/home/gf/miniconda3/envs/gfenv/bin/python /home/gf/gfhandler/start_gfhandler.py
User=gf
Group=gf

[Install]
WantedBy=default.target
```

In order to transform our GFHandler program into a `systemd` service, a `systemd` unit file was written. This file specifies a configuration for the service, such as when to boot (in this case, after the database service), the path of the file to execute, the working directory, and more. The full contents of the `gfhandler.service` unit file are show in Listing 7.2.

[16] https://geometryfriends.gaips.inesc-id.pt/
[17] https://www.freedesktop.org/wiki/Software/systemd/ (accessed August 2, 2018)

8

Evaluation

Contents

In this chapter we describe the evaluation process used to assess the usability of the new system, which consisted of tests with real users.

After analysing the results of the tests, Section 8.4 discusses the results in the different scenarios in light of the requirements introduced in Section 4.2.

8.1 The Test Scenarios

Three test scenarios were devised to evaluate the usability of the platform. In all the scenarios, a single person is asked to interact with the new website in order to complete a series of steps and answer a few questions. All questions and steps are present in a Google Forms[1], which is made available to the tester at the beginning of each scenario. The three forms, one for each scenario, are provided in Appendix C.

The activities on each of the three scenarios are based on the goals which we wanted users to be able to accomplish when using the new platform, as specified in Section 4.2. No scenario was devised for the general public as the low amount of goals for this group did not justify the creation of a specific test.

The Google Forms questionnaires follow the same format where each form is split into three sections:

1. the first asks details about the person (age, gender, field of study) and also two important questions to assess the past experience of the person with activities related to what will be tested, namely:
   (a) whether the person has experience in artificial intelligence competitions or simply programming contests, which would be the case of the usage of platform like Mooshak;
   (b) whether the person has knowledge of artificial intelligence in games.
2. the second section contains the steps the person is meant to follow to complete the test, with a few questions along the way. This is the only section which varies greatly from scenario to scenario.
3. the third and last section contains a standard System Usability Scale[2] (SUS).

The actions executed on step 2 make permanent changes on the platform, which were not reset in between tests. This attempts to simulate the natural flow of events the platform would undergo if it were open to the public, that is, the number of competition instances grow with the tests, as do submissions and users. All files required by the testers were available on the website or were provided (e.g. via a flash-drive) during testing.

[1] https://www.google.com/forms/about/ (accessed August 3, 2018)
[2] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (accessed August 3, 2018)

Immediately after each scenario, each tester is asked to provide feedback on the activities they just executed, including difficulties, opinions and suggestions. This is useful to establish the modifications that the platform still needs to undergo.

One scenario evaluates the competition organization features of the new platform, while the other two test participant related features.

8.1.1 Competition Management Scenario - CM

The CM scenario has testers enact the role of a competition manager/organizer. Given that one of the main objectives of this new platform is to facilitate the organizer's job, this scenario can be considered the most relevant of the three.

The results of these tests will allow conclusions to be made on the usability of the various competition management interfaces of the new website.

Testers following this scenario are supposed to log in with a special administrator account and be able to create and configure a simple Geometry Friends competition. Afterwards, they are asked to create and run two submission presets (on the created competition), one for each platform (Windows and Linux) and track their results, which are computed in real time by the system. The Linux preset always results in a compilation error (where an error log is made available by the platform), while the other succeeds with a score greater than zero. Testers are meant to understand this outcome and answer a free-answer question. Lastly, they need to generate a video of the submission that did not fail.

8.1.2 Thorough User/Participant Scenario - TU

The TU scenario has testers enact the role of participants. Testers are told to create an account an participate in a specific competition by downloading the competition package (contains a copy of the game, the world file of the competition, a sample agent and packaging scripts) and to develop a simple agent in VisualStudio or MonoDevelop. Then, after building their solution and testing it locally, they must ZIP it using the right packaging script and upload it through the new website. Lastly, they are required to report on the results of their submission.

The first section of the TU questionnaire features an additional question to determine the past experience of the tester with VisualStudio or MonoDevelop.

This scenario is meant to test not only the usability of the participant interfaces of the new website, but also the usefulness of the help provided to the participants in terms of navigating the Geometry Friends framework to easily develop, package and upload a simple submission.

8.1.3 Simple User/Participant Scenario - SU

The SU scenario is similar to the TU test, but simpler. Instead of developing a simple agent, a couple submission ready ZIP files are provided, which testers are meant to submit to the same competition.

A single but very important question was added to this test which asks users about what happened to their first submission after they have submitted their second. It is expected for testers to be able to respond that only their latest submission will count for their final score.

As such, this scenario will mostly just test the usability of the participant's interfaces of the new website.

8.2 Pre-Tests

Pre-tests were made in order to prepare for the final tests. All three scenarios were tested on a classroom on university campus which contains several desktop computers. The computer used inside this classroom was always the same, a Windows 10 computer with VisualStudio 2017 installed and 7-Zip present on its PATH environment variable. From this point forward, this computer, under this setting, will be referred to as the LAB.

Because there is only one scenario where the tester is asked to enact the role of a competition organizer and the scenario has little in common with any of the other two, all testers were asked to participate in two scenarios: one CM scenario and one of the participant's scenarios (SU or TU). The order was randomized to ensure participants did not always start in the same scenario.

All pre-tests were executed on the LAB, on 09/07/2018, by three masters students in Information Systems and Computer Engineering. Three CM tests, 2 TU tests and 1 SU test. A total of 3 people participated. While the sample is small for any statistical inference, the provided feedback led to a few changes to the website, which are shown in Table 8.1.

Given that past experience on either VisualStudio or MonoDevelop seemed to make a significant impact on the test, the question related to this was added to the TU questionnaire from this point forward.

8.3 Final Tests

A total of 18 people tested the system. Every person did a CM test and one of the other two participant's tests, SU or TU (as with the pre-tests, the order was randomized to ensure testers did not always start in the same scenario). The system usability results can be seen in Table 8.2, but the results to each specific scenario will be approached in the following sections.

Feedback/Problems	Website Changes
A tester tried to use the Linux packaging script under Windows on a TU test	The video tutorial was changed to specify what script to use under Windows
When refreshing the page to see the dynamically generated level previews on the competition editing page, two testers lost unsaved changes on the HTML form (CM test)	A JavaScript alert was created which alerts people when leaving the competition edit page with unsaved changes
A Bad Request error was encountered when one tester tried to run presets with no competition selected (CM test)	The bug was found and patched
Some testers had trouble realizing submissions take time to evaluate	Success headers and tooltips were added across the website to alert people of this fact
Some testers did not notice or did not care about the guides	More warnings and references to the guides were placed around the website

Table 8.1: Website changes according to pre-tests feedback

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 8.2: SUS results for all the test scenarios.

According to results of 500 SUS evaluations[3] [4], the average SUS score is 68 and achieving over 80.3 is considered a very good score.

The tests were conducted during summer time, between July 20 and September 14.

8.3.1 CM - Results

Testers were asked to bring and use their personal computers, if possible, to test the developed platform under different systems and browsers. 6 out of the 18 CM testers used their own computers, while the others used the LAB (the LAB is the same as explained in Section 8.2). Across all testers, Ubuntu, Windows 10 and macOS were used, as well as Google Chrome, Mozilla Firefox, Microsoft Edge and Safari.

Seventeen out of 18 testers were college students (10 of which were IT students and 8 out of these 10 claimed they had knowledge of AI in games or that they had participated in programming contests with platforms such as Mooshak or AI competitions) between 18 and 25 years old. The other was one researcher of the field of study in question (AI for games), who was familiar with the Geometry Friends game and competition. Two testers were women.

All testers completed the required steps successfully, although intervention was sometimes required

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

on the competition configuration step because some testers did not understand they had to refresh the page to see the level previews appear.

The CM test had only one free-answer question, which asked testers to report on the results of the presets that they ran during the test (see Section 8.1.1). Only one person did not eventually realise that the status of the submissions change in real time and answered that one of the presets was processing. While this still means the person was able to find the status of their submission, this was still not the expected answer.

All the others got both the status and the scores correctly. 15 out of the 18 testers included the error log in their answer, which indicates that it was easy to find.

The researcher who also tested the system complimented the idea and modern design, and provided some constructive feedback which led to improvements to the new website (see Section 8.3.4).

In one of the tests, a generated video showed up as completely black in the browser of a tester's own personal computer (Windows 10 + Google Chrome). Later, when trying to diagnose the problem, not only was discovered that the GFHandler reported that the video was generated correctly, but also that the video actually played on several browsers (both Firefox and Google Chrome). Nothing could be concluded, but given the evidence it does not seem that either the website or the GFHandler were at fault.

In the only test which uses Safari, it was discovered that the HTML input type "date" field (used when configuring a competition's start and ending dates) was not working. Website w3schools confirms that this HTML element is indeed not supported in Safari.[5]. As this only affects competition management, and until a workaround is implemented in a future version of the website, a simple warning was introduced in the competition management page to discourage the use of the Safari web browser.

As for the usability of the competition management interfaces, a SUS score of 76.6(6) was obtained (see Table 8.2). According to the explanation above, this means that the competition management interfaces achieved an above average usability score, close to being considered very good.

8.3.2 TU - Results

Nine people played the role of participants on the TU test: 8 students (6 in IT fields) and one researcher (the same as in the CM test). While only 6 people claimed to have used VisualStudio or MonoDevelop before, everyone was able to complete the steps.

The major difficulty for most of the participants was to import the solution correctly in VisualStudio. This was not explained on the Google Forms and so many people imported it incorrectly, at first. When realizing they were stuck, they were finally forced to find the guides and video tutorial on the website. Afterwards, all of them were able to create a simple agent which successfully ran locally.

[5] https://www.w3schools.com/tags/att_input_type_date.asp (accessed September 23, 2018)

Everyone submitted their developed agent and correctly interpreted the outcome (status and score) of their submission. The researcher was more critic during this test, suggesting several changes to the video and guides. He also suggested that the two evaluation phases (public levels and then private levels when the competition ends) should be more evident. Changes were made to the website according to this feedback and more, as detailed in Section 8.3.4.

The TU test achieved a usability score of 78.3(3), as shown in Table 8.2. This was quite similar to the result obtained for the CM test and, as explained earlier, this a score well above average.

8.3.3 SU - Results

The simplest of the test scenarios, 9 people participated in the SU test, all of them students, 4 of them in IT. Two of the students were female. Everyone was able to place two submissions to the platform, as per the scenario, but only one person did not understand that the status of their submissions would change over time and was thus unable to provide the expected answers.

All testers were able to realize that their first submission was voided/cancelled/overridden after they submitted a second time to the same competition.

Out of the 8 testers that were able to see that their first submission resulted in a compile error, 7 were able to find the error log and include the specific error in their answer.

The SUS usability score of the SU test was 84.4(4), as shown in Table 8.2. This number is not only considered a very good score, but it was also significantly larger than the CM or TU SUS scores, possibly due to the simplicity of this test scenario which required less steps.

The difference between this SUS score and the one obtained in the TU test may suggest that the participant interaction with the new website may be more usable than the interaction with the Geometry Friends framework, a type of interaction only present in the TU test.

8.3.4 Feedback and Improvements to the System

Similarly to the pre-tests (Section 8.2), the feedback from the testers allowed for several website improvements to be made. The feedback was mostly in the form of complaints, constructive criticism or simply tips. Most of the tips came from the IT students, as they had more informed opinions. As an example, there were several complaints about dead end pages and redirection issues, which were fixed by the addition of more buttons and hyperlinks. Some of the most significant improvements are shown in Table 8.3.

One important thing to note is that, just as with the pre-tests, no changes were required on the GFHandler component.

Feedback/Problems	Website Changes
Some testers did not understand that they had to refresh the page to see the previews of the levels (CM test)	The guide was updated to address this concern more clearly. A tooltip was added to the level preview placeholder images suggesting a refresh to the page
People complained that although the website did state where certain information could be found, the website could redirect them automatically to more appropriate pages	Page redirects were changed, such as redirection to the scoreboard after running presets or redirection to the profile after submitting
On the CM test, people complained that there was a lot of separation between the management interfaces and the rest of the platform, making the management pages hard to find	Administrator account now see a different home page that has links to management tools. A button to create competitions was also placed, for admins only, on the public competition list
Testers complained that the competition details page did not provide any actions, becoming a dead end	Several buttons were added to the competition details page, more specifically the buttons that were already in the competition list
Many people did not realise they could mouse over several elements of some pages to understand more about what they meant, because the tooltip had a delay	The website now uses a tooltip based on jQuery UI (https://jqueryui.com/ [accessed August 13, 2018]) which shows up instantly and is more readable
Some users were confused that a button to sign in was still present on the home page after signing in	The sign in button on the home page was replaced with a button that leads to the user's profile page
A user suggested that the website should further clarify the differences between the two evaluation phases (public and private levels)	Both the submission guide and the video tutorial were updated to make more evident the existence of these two phases, as well as how the final score includes contributions from both phases. On the competition details page, the private levels are now separated from the public levels. On the scoreboard, a column in the level specific scores table now allows people to differentiate between the public and private levels

Table 8.3: Some of the website changes according to the feedback of the final tests

Implemented User Centered Requirements		IMP	UT
Organizers	Organizers should be able to create new GF competition editions online	✔	✔
	Competition specific parameters (e.g. dates, bonuses, etc.) should be configurable	✔	✔
	Organizers should have the ability to produce baseline submissions and corresponding results for each competition instance to guide the participants	✔	✔
	The Geometry Friends game version used in the future competitions may be changed	✔	✖
	All processes spanning from a participant's submission to the display of the its results should be done automatically without human intervention	✔	✔
Participants	Participants should be able to access all competitions open for submissions at any given moment, online	✔	✔
	Competition specific details should be made transparent to the participant	✔	✔
	The GF framework should be available for download on the website, including sample agents and the competition levels	✔	✔
	Participants should be able to upload their submissions (and technical report) online	✔	✔
	The results of a submission's automatic evaluation should be made transparent for the participant	✔	✔
	The participant may submit multiple times for a single competition, but only the last submissions should count for the participant's final score	✔	✔
	The participants should have the same agent development freedom they have had so far, that is, agents may use third-party and platform specific dependencies	✔	✖
General Public	Anyone may access the details and results of both unfinished and finished GF competitions	✔	✖
	All of the submitted agents' source code and technical reports should be available to everyone (finished competitions only)	✔	✖

Table 8.4: Table showing which user centered requirements were successfully implemented (IMP) and which were user tested (UT).

8.4 Successfully Implemented Requirements

The tests above were meant to directly address and test many of the requirements listed in Section 4.2. Table 8.4 shows which user centered requirements were implemented and which were successfully tested by real users. As for the non-functional requirements, Table 8.5 shows that they were all implemented.

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.

Implemented Non-functional Requirements		
Data Security	The system shall be protected against man-in-the-middle attacks	✔
	Persistent data shall be protected against tampering attacks such as SQL injection	✔
	Data sent to the system shall be properly validated	✔
	Sensitive user information (e.g. passwords) stored in the system must be protected against theft	✔
	Competition management data such as competition configurations and game versions shall not be tampered with by non-organizers.	✔
Submissions	Incoming submissions must have a size limit which will depend on competition configuration	✔
	Zombie submission files shall be removed from the system	✔
	The system shall be safeguarded against malicious submission code	✔
	Each submission shall be evaluated under similar conditions	✔
	The system shall support evaluating a single submission for up to 500 minutes	✔

Table 8.5: Table showing which non-functional requirements were successfully implemented.

Other additional features outside the scope of the initial requirements were implemented and tested during the development phase of this project. Some of the most significant include:

- the submission videos, which were actually tested on the CM scenario;
- a password reset feature;
- the automatic sending of critical error logs via email;
- two submission packaging scripts which are shipped with the framework package;
- recuperation of submission handling processes in the event of a system reboot.

9

Conclusion

Contents

A new and functional platform for the Geometry Friends Game AI competition was developed successfully during this work. 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 many of them tested repeatedly on the aforementioned tests with people, as was stated in Chapter 8.

Taking into consideration the initial goals of this project:

- the new platform allows the organizers to be free from any submission handling/evaluation procedure;
- the new website allows:
  - organizers to manage several aspects of the GFGAI competition;
  - participants to compete in the competition by offering the GFGAI framework and and allowing submission uploads;
  - feedback to be automatically given to participants regarding their submissions;
  - knowledge to be made available automatically and immediately after competition restrictions allow them to (e.g. past submissions can only be made public on finished competitions);
- the usability score obtained when testing the system using the System Usability Scale was above average.

As such, it is possible to conclude that all objectives were met successfully, including the main goal of designing and implementing a new platform to support the Geometry Friends Game AI competition.

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 (see Appendix B).

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 certain 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.

9.1 Future Work

A lot more can be added to the developed competition platform, as we will see in Section 9.1.1. Section 9.1.2 approaches all game based AI competition platform in a more general way.

9.1.1 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, that is, it would allow the web page to be better prepared to increasing amounts of competitions.

While the guide states that all times are in Lisbon time, this may not be very "internationally 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 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.

A number of other small improvements could be made, such as:

- ability to edit things after creating them. This would include user information (with the exception of the password, which may be reset), competition descriptions to allow organizers to better convey information about a specific competition, etc;
- ability to generate the level previews in a separate thread of the GFHandler, to prevent it from potentially taking a very long time to generate;
- asynchronous requests on the level previews in the competition management page, to prevent the organizers from having to refresh the page to see the new level previews.

- creating a service inside each VM that is responsible for communicating to a listening socket on the GFHandler to inform it that it has been booted up. This would remove the current sleep and make the system more robust.

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, more specifically the modification of the code base to allow for cross-platform development. However, before making its code base public, the Geometry Friends game could use, for instance, some external documentation for developers and perhaps extensive refactoring.

One of the limitations of the Geometry Friends game that would have a significant impact if fixed is the simulation speed up feature. This would lower the time between a submission and the feedback provided to the participant.

9.1.2 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 [1] 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 related tables (e.g. users, scores, etc.) and scripts to install everything.

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

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.

Bibliography

[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.

[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 191-198. 2017

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

[11] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Woo- Seo, R. Simmons, S. Singh, J. Snider, A. Stentz, W. Whittaker, J. Ziglar, H. Bae, B. Litkouhi, J. Nickolaou, V. Sadekar, S. Zeng, J. Struble, M. Taylor, and M. Darms: "Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge". Technical Report, Carnegie Mellon University. 2007

[12] J. Schmidhuber: "Deep Learning in Neural Networks: An Overview", Neural Networks 61: 85–117. 2015

[13] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa: "Robocup: The robot world cup initiative", in Proceedings of the first International Conference on Autonomous Agents, ACM, pp. 340–347. 1997

[14] J. Bennet and S. Lanning: "The Netflix Prize". KDD Cup and Workshop, 2007. www.netflixprize.com

[15] 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

[16] 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

[17] 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

[18] T. Schaul: "A Video Game Description Language for Model-based or Interactive Learning", in Proceedings of the IEEE Conference on Computational Intelligence in Games, 193–200. 2013. Niagara Falls: IEEE Press.

[19] M. Genesereth, N. Love: "General Game Playing: Overview of the AAAI Competition", in AI Magazine, vol. 26, pp. 62-72. 2005

[20] D. Perez, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C. Lim, T. Thompson: "The 2014 General Video Game Playing Competition", in IEEE Transactions on Computational Intelligence and AI in Games, vol. 8, pp. 229-243. 2015

[21] 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

[22] R. D. Gaina, D. Perez-Liebana, S. M. Lucas: “General Video Game for 2 Players: Framework and Competition”, in Proceedings of the IEEE Computer Science and Electronic Engineering Conference (CEEC). 2016

[23] 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

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

[25] 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

[26] 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

[27] J. P. Leal, F. Silva: “Managing Programming Contests with Mooshak”, Universidade do Porto. 2001

A

Pages of the New Platform's Website

This appendix contains screenshots of the most important pages and forms of the new Geometry Friends competition website. One of the most important forms, the competition configuration page, is present in the document and is not repeated here.

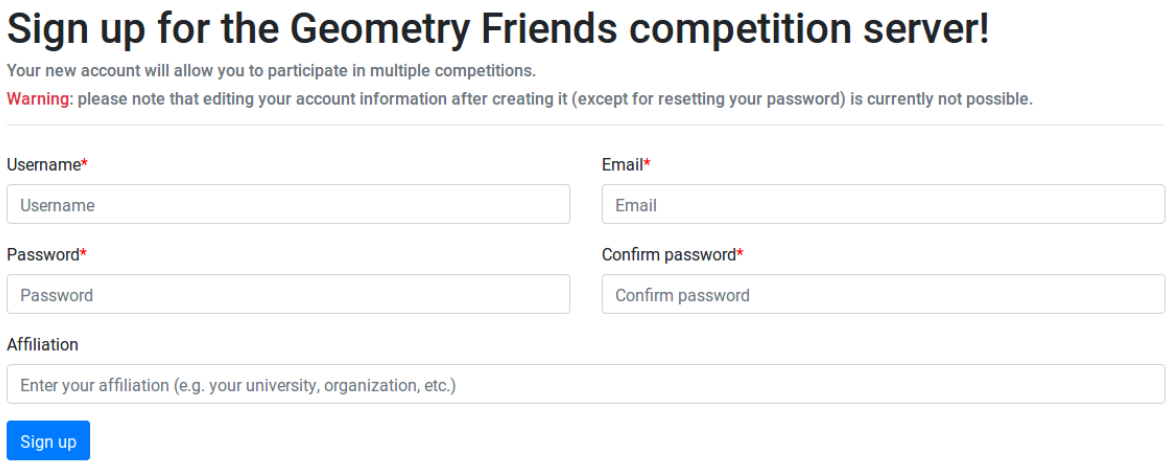


Figure A.1: The sign up form of the new GFGAI competition platform's website.

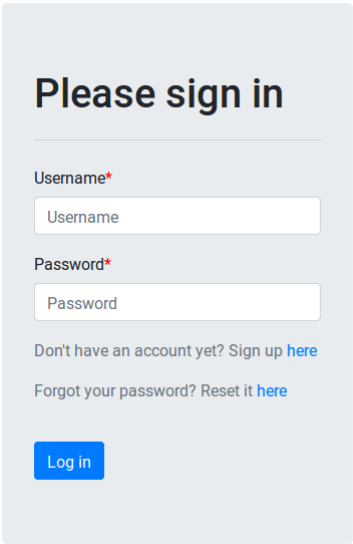


Figure A.2: The login form of the new GFGAI competition platform's website.

Profile page

Ricardo

Profile

Joined:	2018-07-18, 12:13:11
Email:	ricardo.gomes.costa@tecnico.ulisboa.pt
Affiliation:	IST

My submissions

Remember that submissions take time to evaluate, so make sure to **hit refresh** to see if their status changed!

Not sure about how to interpret your submission's status message? Mouse over your submissions or read the guide!

Date	Competition	Status	Total Score
2018-09-21 12:41:57	Example Circle Competition	compilation error (errors)	0.0
2018-09-21 12:30:30	Example Circle Competition	overridden	0.0
2018-07-18 14:02:21	Example Competition	handled all	600.0
2018-07-18 14:00:51	Example Competition	overridden	0.0

Figure A.3: A profile page of a user who has submitted multiple times. Notice how the latest submission, which failed with a compilation error, possesses a hyperlink which enables the participant to download an error log to see what went wrong.

Create a new GFGAI competition

Warning: it is currently impossible to modify the competition name, description or world file after it is created. It is only possible to edit competition parameters of competitions which haven't started yet (before their start date). To know more, read the guide.

Competition name*

Enter the name of the new competition

Start Date*

mm / dd / yyyy

End Date*

mm / dd / yyyy

Competition description (maximum of 512 characters):*

Write a small description of the competition

Competition world file*

Browse... No file selected.

Create competition

Figure A.4: The form which allows managers to quickly create a new Geometry Friends Game AI competition.

Create a new submission preset

Warning: Presets can't be changed after they're created. They represent baselines. To know more, read the guide.

Preset name*

Preset name (e.g. RRTAgent2017)

Platform

○ Windows ○ Linux (mono)

Preset zip file* Browse... No file selected.

Technical report (pdf)* Browse... No file selected.

Create preset

Figure A.5: Form which allows managers to create a preset. Notice the similarities between this form and the participant's submission form (Figure 6.3).

Run submission presets

Set baselines for specific competitions. Running a preset is similar to submitting a predefined agent.

Select a competition

Select a competition to submit the presets to

Presets to be submitted (use the CTRL key to select multiple presets)

ExampleAgent
ExampleAgent (Windows)
MSBuild Error (Linux)
fdfd

Run presets

Figure A.6: Managers can command multiple presets to be evaluated on specific competitions.

Version History

This shows all previous game versions. The most recent one is the active game version.

8: 2018-09-21 14:45:51 (current)
7: 2018-09-21 14:44:48
6: 2018-09-21 14:44:36
5: 2018-09-21 14:43:23
4: 2018-09-21 14:11:27
3: 2018-09-21 13:51:45

Upload Form

New Geometry Friends game (zip)*

Browse... No file selected.

Update Game (it may take a bit)

Figure A.7: Notice how easy it is for managers to update the Geometry Friends game version used for the automatic evaluation of the submissions. There is also a version history to keep track of previous versions.

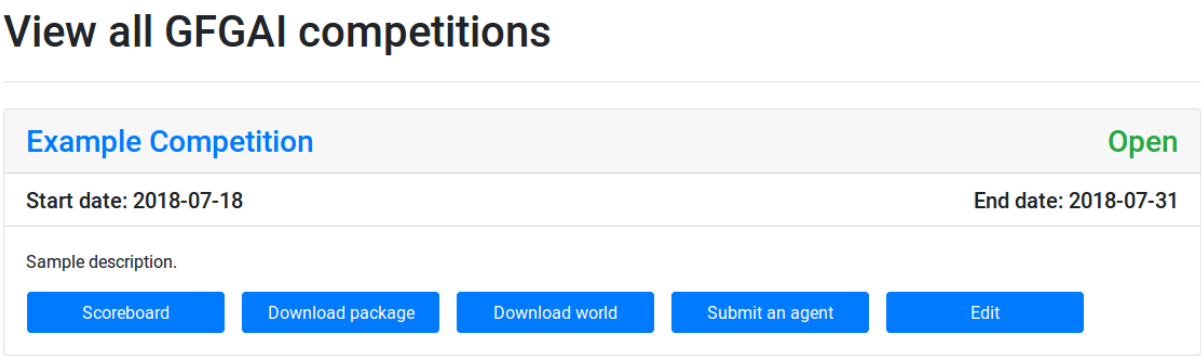


Figure A.8: A single competition in the competition list. The ”Edit” button is only visible to administrators. Clicking on the competition name leads to a page containing details specific to that competition.

Competition details

Pick Another Competition

Example Competition

Description: Sample description.

Available actions:

Scoreboard | Download package | Download world | Edit

General competition details — Finished

Name:	Example Competition
Start date:	2018-07-18
End date:	2018-07-31
Simulations/runs per level:	1
Maximum submission size (zip + pdf):	512 MB
Timeout period between submissions:	0 minutes

Public Levels:

Level 1

Time limit (seconds):	6
Completion bonus:	0
Collectible bonus:	300

Private Levels:

Level 2

Time limit (seconds):	4
Completion bonus:	0
Collectible bonus:	300

Figure A.9: A public page showing details of a specific competition.

Place a submission for a Geometry Friends competition!

Example Circle Competition

Platform*

Windows Linux (mono)

Submission zip file* Technical report (pdf)*

Browse... No file selected. Browse... No file selected.

Upload Submission

Your submissions to this competition

Remember that submissions take time to evaluate, so make sure to **hit refresh** to see if their status changed!

Not sure about how to interpret your submission's status message? Mouse over your submissions or read the guide!

You can also see all your submissions and their current status in your profile.

You don't have any submissions for this competition yet. Unsure of what to do? Check out our guides!

Figure A.10: The submission page, containing both the form and a personal submission history.

Watch submission video

User: Ricardo

Competition: Example Competition

Level: 1

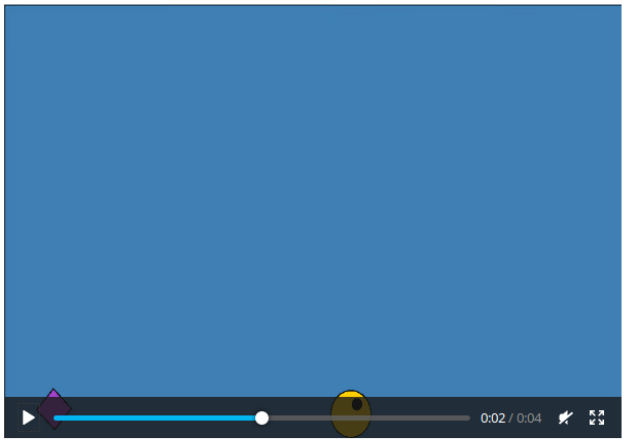
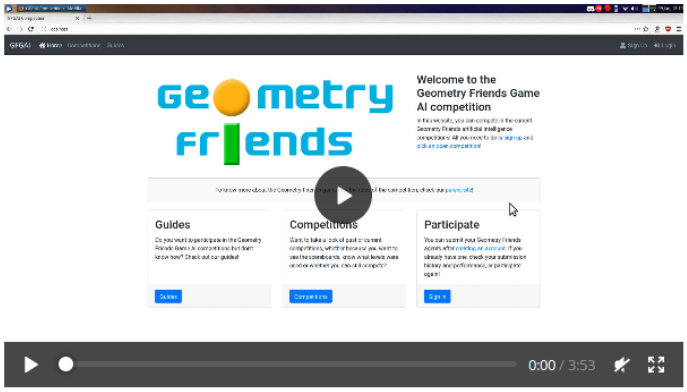


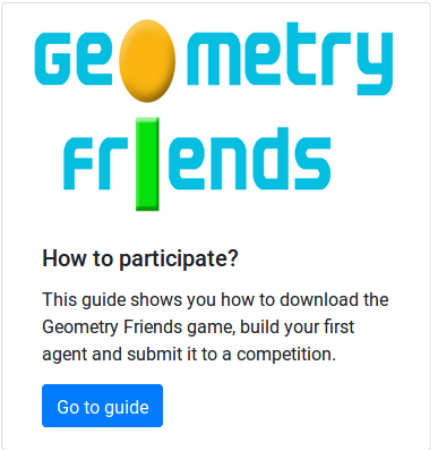
Figure A.11: A video generated automatically, on request, by the GFHandler. After it is generated, it is embedded on the website, as shown in this picture.

Video Tutorial

Have a submission ready in just a few minutes! It is recommended to watch in fullscreen.



Pick a guide!



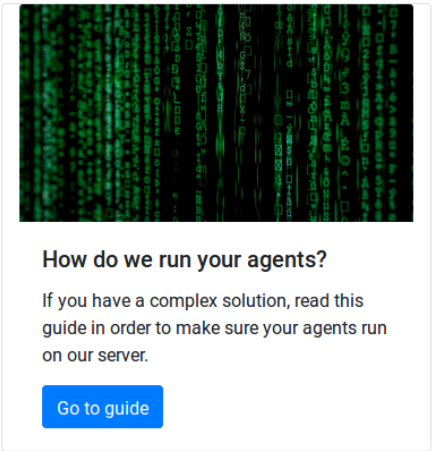


Figure A.12: The guides page, showing two available guides and a video tutorial.

Guide: How to compete on a Geometry Friends Game AI competition?

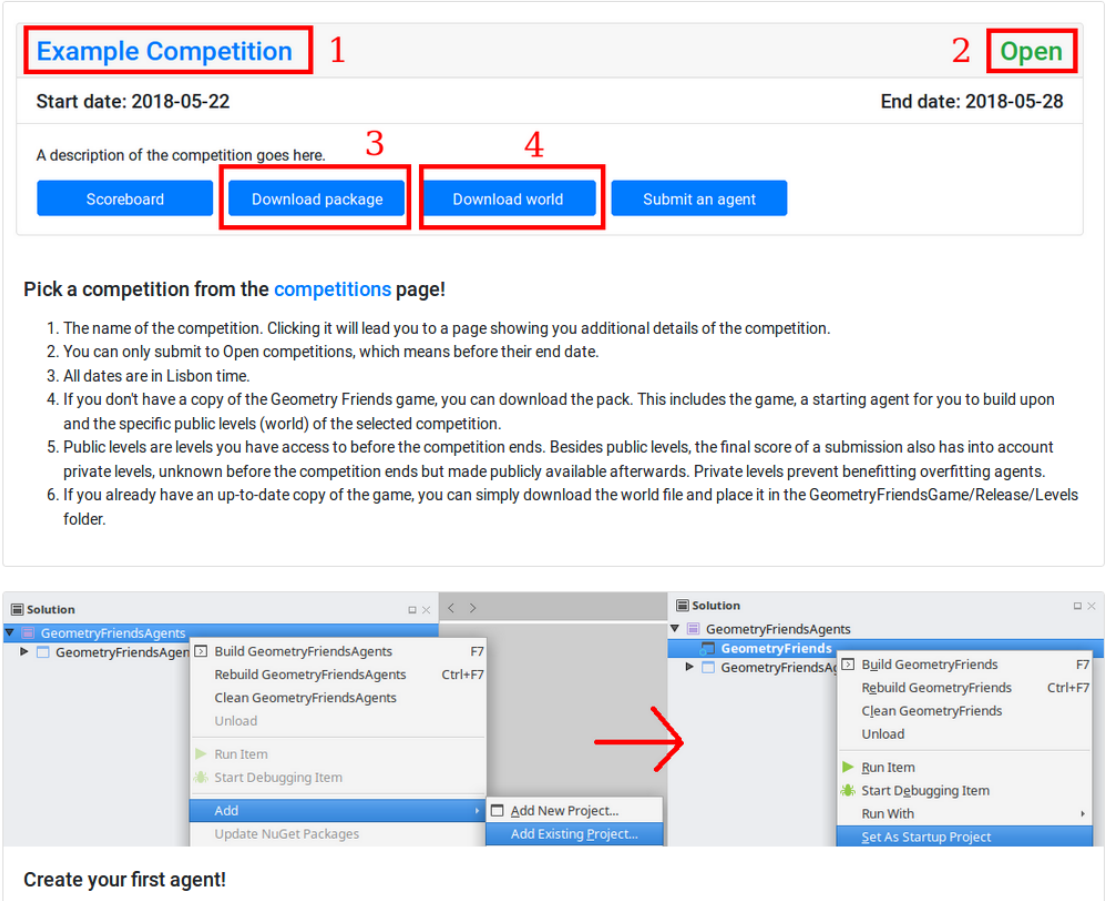


Figure A.13: A small part of an image based guide teaching participants how to place submissions using the new platform.

Guide: How do we run your submitted agents?

What is this guide about and who is it for?

This guide describes how we run and evaluate the agents you submit to the Geometry Friends Game AI competitions.

If you are interested in knowing how to compete, please check out our guide on how to compete.

This guide is for those who:

- got an error (e.g. extraction error, compile error...) on their submission and would like to understand or even replicate the server environment in order to figure out what could be wrong
- want to know what kind of fairness measures are being taken to ensure equal treatment for each submission
- are curious about what's happening 'under the hood'

What machines are being used to run the agents?

Participants are allowed to build their agents with the aid of third party libraries. However, these may be platform specific, which is why we have two different virtual machines available:

- Ubuntu Server 18.04. Uses Mono 5.10.1.47 and MSBuild 15.6.0.0
- Windows Server 2016. Uses MSBuild 15.6.85.37198

This is why you need to select the platform when submitting your agent. If you don't use platform specific libraries, then it shouldn't matter which platform you choose.

Moreover, they are both configured to use the following resources:

- 2 cores
- 4 GB of RAM

Is the system fair?

The virtual machine based system offers great benefits in terms of fairness, such as:

Figure A.14: A small part of a guide which explains how agents are evaluated under the new system.

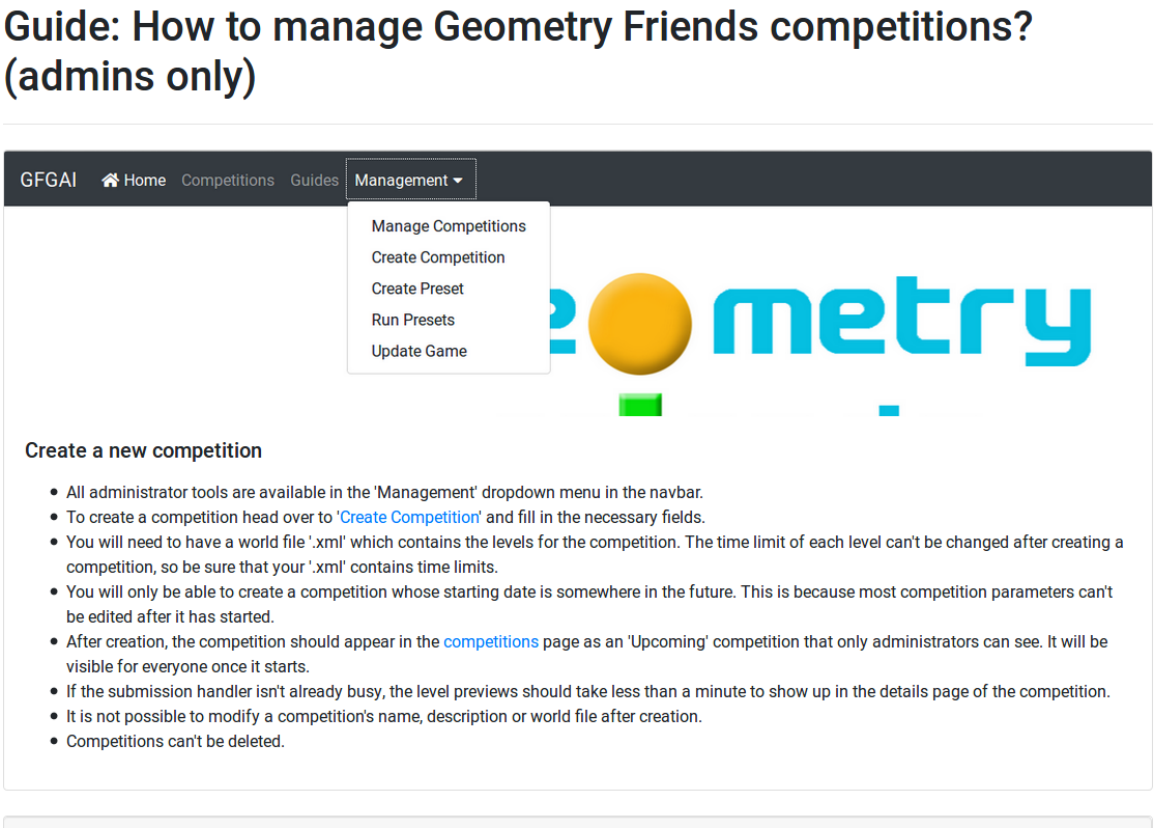


Figure A.15: A small part of an image based guide only available to administrator users and that teaches the basics of managing competitions and presets.

B

Technical Report / System Manual

This appendix contains the PDF document which dives a little bit deeper into some technical details of the developed system, especially how it can be modified and updated in the future.

Some technical details overlap between the system manual and this document, which means that considerable information is repeated, albeit with more detail.This makes sense, given that this document did give considerable insight into the implementation details of the new system.

Geometry Friends Game AI Competition Platform - Technical Report

Ricardo Costa

November 15, 2018

Contents

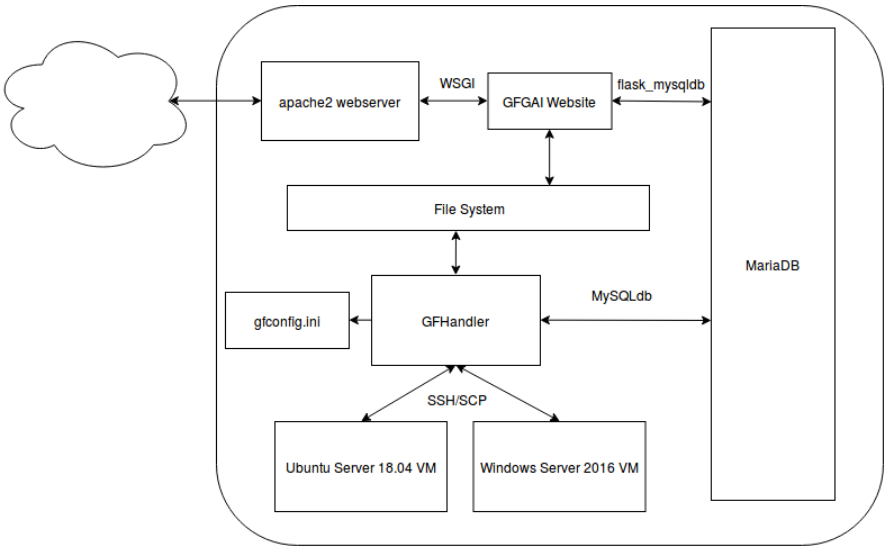


Figure 1: Platform overview

1 Introduction

The Geometry Friends Game AI (GFGAI) competition platform is a system designed and implemented to improve upon the processes involved in managing and running GFGAI competitions. The system was developed mainly to reduce operational costs and to increase the attractiveness of the competition.

One of the main aspects of the design lie upon heavy automation of old processes, e.g. handling submissions and updating results. Thus, it is expected that the system may be operated with very little human interaction.

This document describes the system and how it can be operated correctly. Depending on the goal of the reader, he may wish to address specific sections of this document.

Readers who would solely like to manage GFGAI competitions can skip to Section 6. If the goal is to modify the system, the reader may address Section 3.2 to know more about modifying the website, or Sections 4.1 and 4.4 to configure or modify the submission handling system, respectively.

However, even when the goal is to modify the system, a general understanding may be necessary. In this case, reading the whole document in order is advised.

2 Overview

A simplified visual representation of the system can be seen in Figure 1. It resides in a single machine and is divided into two different independent com-

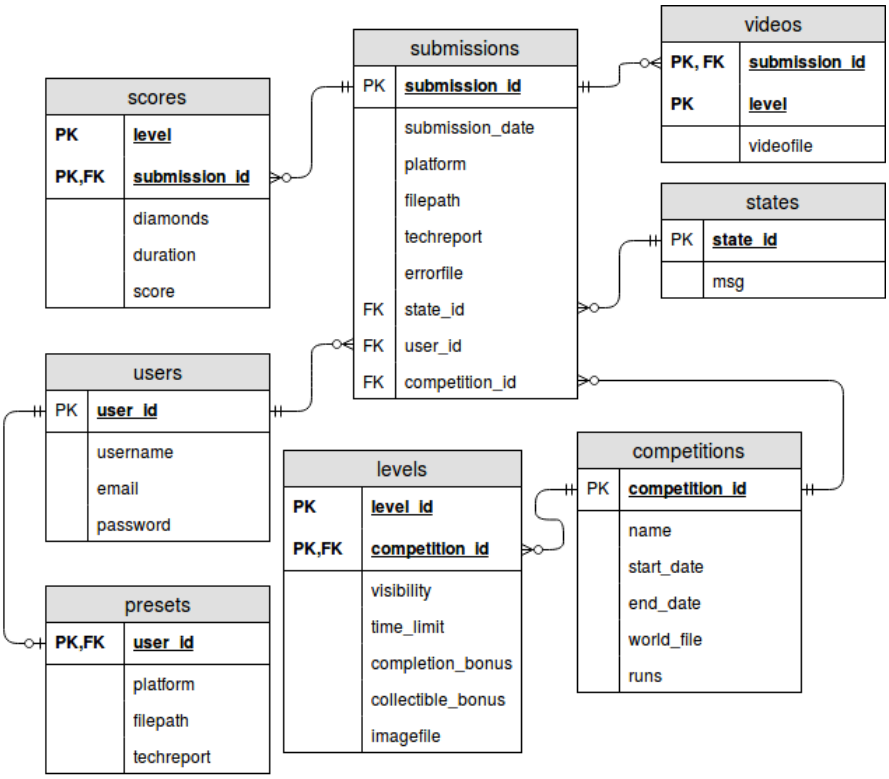


Figure 2: Database model in Crow's Foot Notation (simplified)

ponents:

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

GFHandler the background process which is in charge of handling submissions, generating level previews, etc.

Although they are independent in the sense that they do not communicate with each other directly, they do share information. Both of the described components connect to the same MariaDB database and use the same tables. They also share some directories in the file system, as well as a conda environment.

2.1 Database

MariaDB is used as a database server. While being one of the most popular ones, it is a fork of the also popular database server called MySQL. MariaDB is supposed to be mostly compatible with MySQL, meaning that most commands

can be used interchangeably. MariaDB's CLI can be accessed using the 'mysql' command. This also means that a migration between the two shouldn't be too difficult.

A user named 'gf' was created in the database server and granted all privileges on a database called 'gfdb', which stores all the competition related information.

Figure 2 shows an overview of the database schema. For brevity, some attributes are omitted.

As previously mentioned, this database is accessed both by the website and the gfhandler components. For instance, when a user places a submission for a competition using the website, this information gets recorded in the database. Shortly after, the gfhandler program uses the information to handle the submission and update the database with the results of the submission. These interactions are detailed in Sections 3 and 4.

An SQL script named 'reset.sql' is located under the 'dbscripts' directory. This script contains all the instructions used to set up the 'gfdb' database, such as tables, views, stored procedures, etc. Calling this script (e.g. by using the 'source' command inside the mysql CLI) effectively resets the database. Use with caution because all the submissions, users (except for the admin user) and presets will be gone.

While this script is useful while testing the system, it should not be needed in the future. However, it exists as a way to inspect and learn about all the available tables, views, procedures and functions the database has to offer.

Lastly, it is important to note that eliminating the entries from the database (e.g. submissions), does not remove the actual files (e.g. submission zip, technical reports, etc.) from the system. These need to be removed manually, as described in Section 5.

2.2 Production System and Directory Structure

This subsections presents some important details of the machine chosen to run the system, as well as the directory structure of the deployed system, that is, how the files required to run the system are organized in the file system. Their functionality will be explained throughout the following sections, so use this subsection as a simple and quick reference for the existing files.

The chosen machine has 8 logical cores and 8GB of RAM. It runs the Debian 9.4 'Stretch' operating system. A user named 'gf' was created for this purpose.

Figure 3 is a simplified directory structure containing the most relevant directories and files of the implemented system (many are still omitted for brevity). The majority of the files reside in the home directory of the 'gf' user, meaning they can be accessed and modified without requiring the super user's credentials.

```
/
├── etc/apache2/
│   ├── sites-available/
│   │   └── gfwebsite.conf
│   └── apache2.conf
├── etc/systemd/system/
│   └── gfhandler.service
└── home/gf/
    ├── gfwebsite/
    │   ├── gfwebsite/
    │   │   ├── static/
    │   │   │   ├── img/
    │   │   │   │   └── screenshots/
    │   │   │   └── videos/
    │   │   ├── templates/
    │   │   ├── datastructures/
    │   │   ├── guides/
    │   │   ├── manage/
    │   │   ├── config.py
    │   │   └── views.py
    │   ├── uploads/
    │   │   ├── presets/
    │   │   ├── worlds/
    │   │   │   └── public/
    │   └── gfwebsite.wsgi
    ├── gfhandler/
    │   ├── gfhandler/
    │   │   ├── envs/
    │   │   ├── core.py
    │   │   └── db.py
    │   ├── gfconfig.ini
    │   ├── report_error.sh
    │   └── start_gfhandler.py
    ├── gfclean/
    ├── screenshooter/
    │   └── Screenshooter.dll
    ├── videomaker_tmp/
    │   └── Capture/
    ├── virtualbox_vms
    │   ├── Ubuntu Server 18.04/
    │   └── WS2016/
    ├── dbscripts
    │   └── reset.sql
    └── send_email.sh
```

Figure 3: Directory structure of the deployed system (simplified)

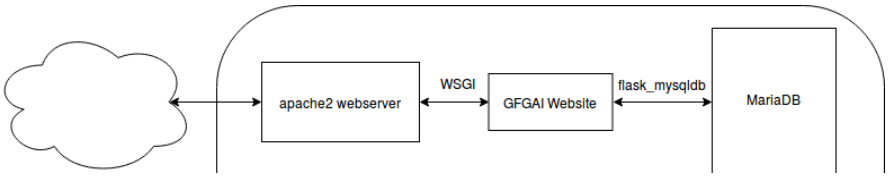


Figure 4: Website overview

3 Website

The website is the front-end of the competition platform. It allows users to participate in competitions or manage them. Section 6 covers the latter.

This section describes how the website is implemented and how it could be changed in the future.

3.1 Architecture

Figure 4 represents the website component of the system. Flask[1] was chosen as the web framework to help build the website. It is written in Python and allows the GFGAI website to use Python (in this case Python 3) as the scripting language (instead of the traditional PHP). Web Server Gateway Interface (WSGI) is the technology which allows requests to be forwarded from web servers to web applications written in Python.

3.1.1 GFGAI Website - web application

The website lies in the 'gfwebsite' directory. The front-end of the website uses:

- Bootstrap (CSS + JS) 4.1.3[2]
- JQuery 3.3.1
- JQuery UI 1.12.1
- Popper.js 1.12.9
- Font Awesome 5.0.10[3]

Very little additional CSS (present in the static/css folder) was added. No additional JavaScript was added.

The entire website is contained in the 'gfwebsite' module, which defines a Flask 'app' object. A 'gfwebsite.wsgi' script file, which acts as an entry point to the web application and imports the mentioned object.

[1] `http://flask.pocoo.org/`
[2] `https://getbootstrap.com/`
[3] `https://fontawesome.com/`

The configuration (e.g. folder paths, database credentials) is stored an a 'config.py' file in the website module.

Miniconda[4] was used to create the Python environment. The website uses a Python 3 conda environment called 'gfenv' which contains, among others:

- Python 3.6.2
- pip
- mod-wsgi 4.6.4
- Flask 1.0.2
- Flask-MySQLdb 0.2.0
- mysqlclient 1.3.12
- Jinja2 2.10
- Werkzeug 0.14.1

For a password reset feature, a script which sends an email to a specific user with a new password is configured in 'config.py'. At the moment, this script is located directly under the home directory and is named 'send_email.sh'. It uses 'exim' and a no-reply address. This script is not to be confused with GFHandler's 'report_error.sh' script, which also sends an email, but it is highly customized for error reporting. More on this is explained in Section 4.3

3.1.2 Apache2 - web server

The web server in use for the website is an Apache HTTP Server 2.X. Two files were altered which can be found in /etc/apache2:

sites-available/gfwebsite.conf	specifies the site-packages directory of the conda environment, the '.wsgi' file location and other website and WSGI related configurations
apache2.conf	Only two lines were added at the end, specifying the conda environment and the mod-wsgi library path

3.2 Modifying the website

A basic understanding of how Flask works may be required to change or even add pages to the website. After understanding how simple routing works, a page can be added in the views.py file inside the module, and the respective HTML file in the 'templates' folder.

[4] `https://conda.io/miniconda.html`

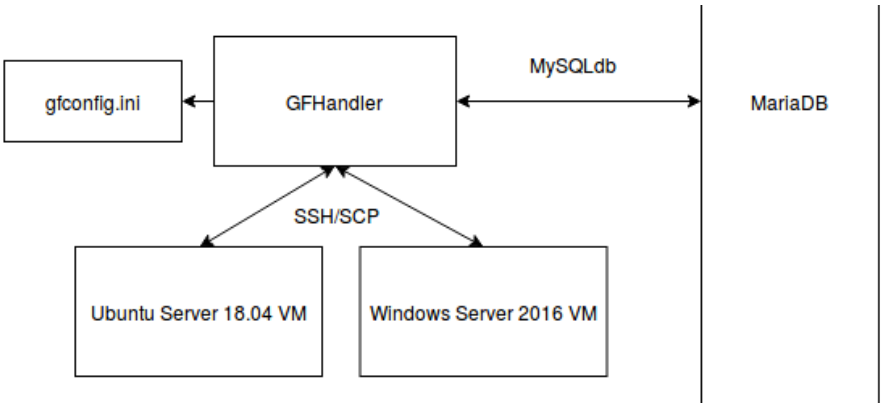


Figure 5: GFHandler overview

Changing an existing HTML file is even simpler. Just locate the desired file in the 'templates' folder (note that more template folders can also be found in the 'guides' and 'manage' sub-modules, which are called blueprints in Flask) and edit it.

After performing any change to the website, it must be reloaded to take effect. This can often be done by updating the 'gfwebsite.wsgi' file. Even if there is nothing new to add to it, a simple

```
touch gfwebsite.wsgi
```

should suffice. If not, then it is possible to manually restart the apache2 service by using:

```
sudo systemctl restart apache2
```

4 GFHandler

The 'GFHandler' is the background process in charge of several things related to the GFGAI competition, such as submission handling, level preview generation and video capturing. It is represented in Figure 5.

It is implemented as a systemd service, similar to the apache2 and mysql services. A systemd unit file named gfhandler.service is placed in the directory '/etc/systemd/system/'.

The service is enabled. As such, the GFHandler starts at boot, meaning that no intervention should be required in order to start this in the event of a system crash and/or reboot.

The GFHandler program is also written in python, and for simplicity it uses the same Anaconda environment as the website (see Section 3.1.1). However,

the only third-party packages of interest in this case is MySQLdb[5], to connect to the MariaDB database.

If separating the two components (website and gfhandler) is necessary, a separate conda environment can be created containing only the mentioned dependency.

The program consists of a python module called 'gfhandler', located in the 'gfhandler' directory and script called 'start_gfhandler.py', located in the same directory. This script is the entry point of the program, and is what is called by the 'gfhandler.service' systemd service.

As can be seen in Figure 5, the GFHandler makes use of two different Virtual Machines (VM). These are used to simulate the game and agents in isolated environments and in different platforms, as some submissions are platform **dependent**. The virtual machines used are:

- Ubuntu Server 18.04 LTS
- Windows Server 2016

The GFHandler is the most complex component of the whole system.

4.1 Configuring the GFHandler

The GFHandler's entry point, the 'start_gfhandler.py' script, assumes the existence of a configuration file called 'gfconfig.ini' in the same directory.[6] Contrary to what happens in the website, where the configuration is in a python file, here we use an external simple text file which allows for the configuration to change without having to change the actual code.

The 'gfconfig.ini' configuration file is split into several sections. First, there's the 'handler' section, which specifies general parameters of the GFHandler, such as:

polling interval the delay between queries to the database before going to sleep

vm boolean indicating whether we are using VM's for the simulations or we're running them locally. This should always be 'true' in a production environment because locally we only have Linux

fresh_copy the path of a clean copy of the game

speed simulation speed (currently advised to stick with speed=1)

email_script path of the script that enables sending any errors and the 'gfhandler.log' file to the specified email inside the script (usually someone in charge of maintaining the system). This script should be placed in the same directory as the log.

[5] `https://mysqlclient.readthedocs.io/`
[6] `https://docs.python.org/3/library/configparser.html`

Then there's the 'db' section, which should specify the same database connection parameters as the website.

The VM configurations are done in the structurally identical sections 'linux-vm' and 'windows-vm'. The VM's are managed using VirtualBox[7] version 5.2.12. The options are:

name the name of the virtual machine

snapshot the name of a VirtualBox snapshot of this machine where the machine is ready to receive submissions

user the username of the non-admin account of the VM's OS

host name/location of the host to connect with the VM through SSH. Since the VM's are in the same system as the GFHandler, this can be left as 'localhost'

port the port where the VM is listening to for SSH connections

submission_dir an empty directory **inside** the VM where the participant's submission will go to for later processing

The 'screenshooter' section contains the configuration for the subcomponent of the GFHandler which is in charge of generating the level previews for new competitions. The parameters are as follows:

screenshooter_dir similar to the submission directory of a VM, but this one is local. It should point to the base directory of the clean local copy of the game

screenshooter_dll the agent which captures a screenshot, saves it in the working directory and exits

screenshot_name the name of the captured screenshot

crop_dimensions the dimensions the screenshot is supposed to have

output_folder the website folder where the screenshots will go to. The website is already expecting the current value

Lastly, the 'videomaker' section describes the parameters for the component that deals with capturing videos of finished submissions (this feature uses FFMPEG version 3.2.12):

src_dir a directory, preferably empty, where the captured frames will reside while being processed

crop_dimensions the dimensions the frames are supposed to have

[7] `https://www.virtualbox.org/` (accessed August 1, 2018)

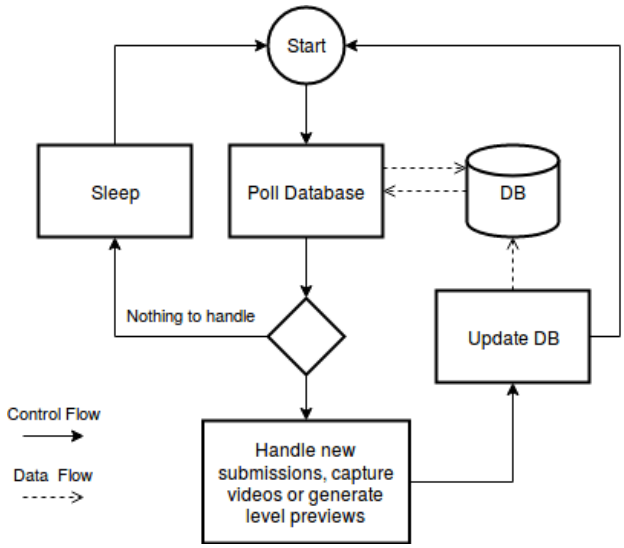


Figure 6: GFHandler simplified flowchart

framerate the '–capture' option of the GF game is set to always capture 24 frames per second, so until that is changed, this value should always be 24

output_folder the website folder where the videos will go to. The website is already expecting the current value

4.2 Understanding the GFHandler main loop and data structures

Whether modification to the GFHandler is required or simply desired, a minimal understanding of how it works is necessary. This section presents a quick introduction to the insides of the GFHandler program.

After being started with the 'start_gfhandler.py' script (which should be done automatically at boot by systemd), the GFHandler enters a polling loop as shown in Figure 6. If there's a low amount of activity on the website, then the GFHandler will likely spend most of its time sleeping.

A python object called GFHandler is in charge of controlling most of the operations. Its 'start' method contains the polling loop, while its 'stop' method gracefully terminates it. One of the most important queries the GFHandler executes when polling the database is to a view called 'next_submission', which returns at most one row with the next submission to handle.

Submissions are processed - processing includes actions such as extraction, compilation, simulation, etc. - inside 'environments', classes which define methods to manipulate submissions. The environments are implemented in the 'envs'

sub-module. Notable environments are the WindowsVM, LinuxVM and Screenshooter environments.

The environments are created once during initialization, since they have a lot of configuration parameters that remain unchanged throughout execution. For parameters that change from submission to submission there's the 'setup' method, which may also contain other setup operations. In contrast, there's also a 'teardown' method to be called after every submission is handled. As such, most of an environment's actions are enclosed between setup and teardown calls.

The file 'db.py' contains most of the methods and data structures which communicate directly with the database, such as obtaining new submissions to process, or new competitions to capture level previews of. An important class is the 'Submission' class which has methods to update the database with new scores, update its status flag - a column in the submission table indicating the status of the submission -, etc.

Figure 7 shows a class diagram containing the most important classes of the GFHandler program, including relationships between them.

Having in mind a normal submission handling process, the main idea is that a `GFHandler` object, which is the main object in charge of dictating the control flow of the entire program, has a `Submission` object and a `SubmissionEnv` (submission environment) object. A submission is evaluated inside a submission environment.

The actions executed inside a submission environment correspond to external commands (not in Python), such as compiling a C# solution. To facilitate this, a `SubmissionEnv` extends a class called `CommandDispatcher` which provides a very simple wrapper for the `run` method of the `subprocess`[8] module. Most methods inside a submission environment use the wrapped `run` method, which was named `run_cmd`. This wrapper specifies an option to the original `run` method which guarantees that injection cannot occur on these external commands.

Just like the `subprocess run` method, our `run_cmd` returns a `CompletedProcess` instance, an object which contains information about the external command executed, such as standard output and standard error streams, and exit codes. Listing 1 shows how one of these external calls can appear simple in the GFHandler's code.

Listing 1: Method in charge of running the external command which compiles a solution

```python
def compile(self, timeout=None):
    """Compiles the solution using MSBuild in the root of the
    submission directory.
    """

    cmd = ["msbuild", "/p:AssemblyName=" + self.assembly_name]
    return self.run_cmd(cmd, timeout, cwd=self.submission_dir)
```

[8] `https://docs.python.org/3/library/subprocess.html` (accessed July 30, 2018)

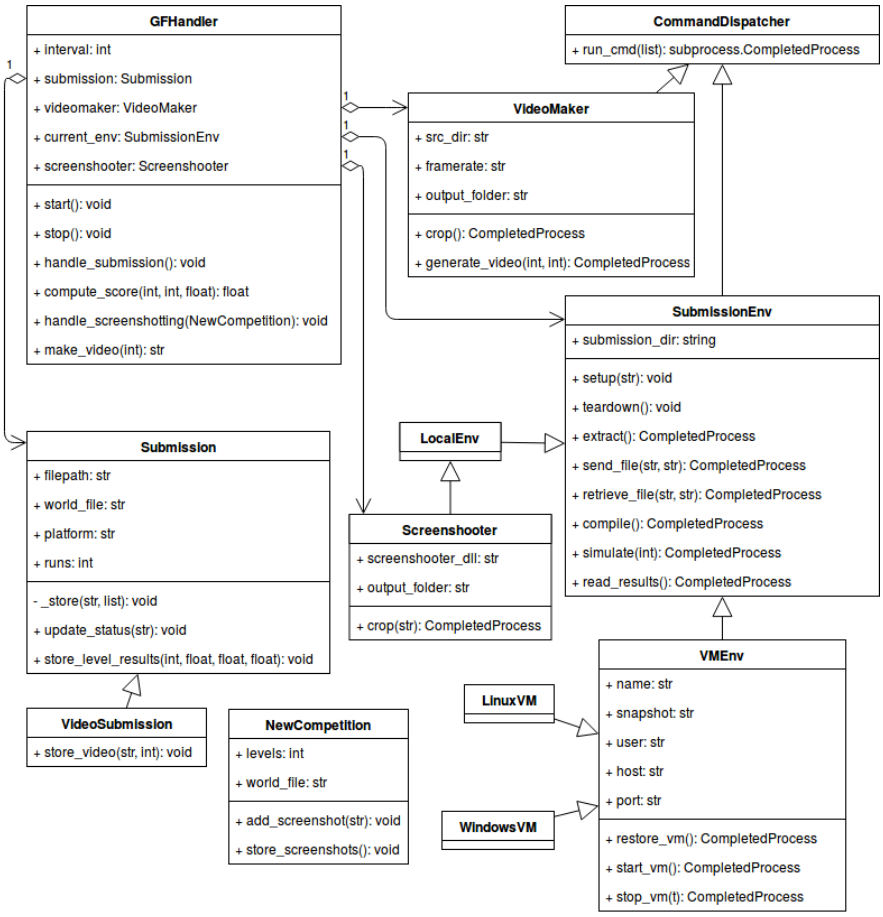


Figure 7: Class diagram containing some of the most relevant classes, attributes and methods of the GFHandler python module.

The `run_cmd` method is overridden several times in the different subclasses shown in the diagram. For instance, in the `VMEnv` class, the class in charge of handling submissions inside the virtual machines, the `run_cmd` method is overridden to make sure every external command is executed through SSH to the virtual machine. Under the `WindowsVM` environment, the `run_cmd` method is overridden to ensure that all external commands are executed in Windows PowerShell.

Other methods were also overridden when more shell specific commands were needed. For instance, the extract method is overridden in the `WindowsVM` environment to call PowerShell's Expand-Archive command.

The `VMEnv` also has methods to start, stop and restore a virtual machine to a given snapshot. In contrast to the `VMEnv`, the `LocalEnv` allows submissions to be handled on the host system. While this ended up being useful for the creation of another environment whose use will be described later, it can also be used in a testing scenario to avoid the usage of the heavier virtual machine environments. However, this environment is vulnerable to malicious submissions, and cannot run submissions which depend on Windows.

4.2.1 Submission status and GFHandler exception handling

Every command (environment action) may fail. The most critical errors would be due to necessary files or folders not being present, such as the ones specified in the GFHandler configuration file. Those files MUST exist.

However, other problems may arise due to bad submissions, which the GFHandler can't predict. Errors can be detected on any action the GFHandler executes. Some, such as the extract, compile or simulate actions are likely to fail. This can happen if, for instance, a participant didn't follow the instructions/guide correctly.

The database's submission table holds a field called 'status', which has the following possible values:

new unprocessed submission, it is 'waiting' to be processed

processing currently being handled by the GFHandler. If the GFHandler is restarted, it will search for a 'processing' submission to resume simulation

handled public the submission has been evaluated on all the competition's public levels and will be processed again once the competition ends

handled all the GFHandler is done with this submission, meaning it evaluated it on all levels, public and private. The submission's score is now final

overridden this submission was replaced by a more recent submission from the same user and for the same competition

extraction error the GFHandler's submission environment caught an error while extracting the submission ZIP file

compilation error the GFHandler's submission environment caught an error while compiling the submission

server error any other error on any of the other commands of the submission environment. These may not be the participant's fault and a system check may be required

It is important to note that any 'handled *' submission may have encountered runtime errors in some (or all) of its runs. This can be caused by a mono exception or by simulation timeout (different than the timeout enforced by the Geometry Friends game). An error log containing information about the error is saved locally and made available for download through the website.

To avoid the GFHandler to hang on failed commands, a global timeout of 360 seconds is enforced as a default for all the commands. This can be a cause of a 'server error'. However, for the actual simulation, the timeout is 330 seconds by default. A simulation may timeout in this way for one of two reasons:

- the levels may not have time limits (problem with XML world file) and the agent is unable to finish before the 330 seconds
- the game may not be responding, in which case it will be unable to exit by itself, and so the GFHandler kills the game process after the 330 seconds

4.3 Logging

The GFHandler is a complex program, and although it was designed to be quite robust, it has many possible points of failure. In the rare event that an error may occur, it may be able to be detected by a "server error" on a submission that was being processed when the error occurred. Other errors that can be more difficult to detect may result in non existent video files, level previews, or some other unexpected outcome.

In any of these situations, the cause may not be a faulty submission, and instead be a serious GFHandler issue that needs to be resolved. Having this in mind, the GFHandler logs key parts of its operation to a filed called 'gfhandler.log' and placed in the 'gfhandler' directory. A server error should be tagged with 'ERROR'.

In case one of these dangerous errors occur, an email should be automatically sent to the address specified under the 'report_error.sh' script. This script, which should not be confused with the 'send_email.sh' script used in the password reset feature, sends the latest log file as an attachment. This way, the error may become apparent without the need to access the machine itself, although most fixes to this type of errors would most likely require intervention inside the machine.

To prevent an excessive accumulation of logging, the file size of the log file is limited to 2MB, with a single backup of 2MB, for a total of 4MB of GFHandler

logging. The backup and file size limiting is all done automatically with the aid of python's 'logging' module, more specifically a RotatingFileHandler handler.

The logging itself should not require maintenance. Having that said, the logging configuration is performed in the 'start_gfhandler.py' script.

4.4 Modifying the GFHandler

This section covers the main parts of the GFHandler that may require modification. Although the system is built to be able to resume the handling of submissions, it is advised to make sure the GFHandler is inactive (i.e. no submissions are being made, etc.) before modifying it.

Ideally, one would only have to modify the configuration file, as described in Section 4.1. But several other motives can exist that require the modification of the actual code.

One of the main reasons one would want to modify GFHandler's code is if the Geometry Friends game folder structure changes, as the GFHandler is highly dependent on the current directory structure of the game. This structure, detailed in Section 6.5, is mostly hardcoded in the 'construct_paths' method of the SubmissionEnv superclass.

Another important aspect of the GFHandler that can be changed is the scoring formula, implemented in the 'compute_score' method of the GFHandler class.

It is also possible to expand the GFHandler's functionality by adding more things to its main loop, in the GFHandler's start method. This requires some thought, as the order in which things happen can matter a great deal.

After modifying the GFHandler code, it is possible to apply the changes by restarting the service with:

```
sudo systemctl restart gfhandler
```

4.4.1 Altering the contents of the Virtual Machines

The whole system is designed to run in headless mode, i.e. without a display attached. Thus, there are two ways to access and modify the contents of a GFHandler VM:

- move all the VirtualBox files related to that specific VM into another computer which has a display, boot the VM, modify it and then move the files back into where they originally were
- start the VM inside the system in headless mode with 'vboxmanage startvm <name> --type headless' and access it with SSH

The SSH credentials for each machine are the same as the ones specified in the GFHandler configuration file. To install or update software:

- in the Ubuntu Server 18.04 VM simply use the 'apt' package manager

- in the Windows Server 2016, use the 'choco' command of the Chocolatey[9] package manager

To power off a VM, simply run 'vboxmanage controlvm <name> poweroff'. It is advised that all modifications are made while the GFHandler isn't busy. To ensure this, you can stop and restart it using the 'systemctl' command, e.g. 'sudo systemctl stop/start gfhandler'.

5 Resetting the Whole System

Resetting the whole system should never be necessary. It was done during development and testing phases. To reset the whole system:

1. make sure the system is off-line to avoid inconsistent results by using systemd (systemctl) to ensure the Apache and GFHandler services are stopped (there should be no need to wait for the GFHandler to become idle before stopping it as it can exit gracefully at any moment);
2. run the "reset.sql" script, which recreates the database without any entries (except for some default entries such as the admin user);
3. delete all the **files**, **recursively**, inside the `gfwebsite/uploads` folder, but **make sure to leave the directory structure intact** as per Figure 3;
4. do the same as in step 3 (delete files only, but keep directories intact) to folders `videos` and `img/screenshots` inside the `gfwebsite/gfwebsite/static/` folder.

Note that this assumes that future development of the website is not placing any static files that are not dynamically generated by the GFHandler component, and perhaps would be desirable to keep even in the event of a full system reset, on any of the mentioned folders.

In the future and if for some reason it becomes necessary to reset the system with some frequency, a bash script could be created to automate these tasks. One was not created because such a scenario seems highly unlikely.

6 Managing GFGAI competitions through the website

Managing the Geometry Friends Game AI competitions doesn't require interacting with the system directly where it is deployed, that is, it is not required to modify code or manually make any change in the server. It is only required to interact with the website's competition management pages.

[9] `https://chocolatey.org/`

But first, a quick introduction on how users are stored. There's a dedicated table in the database for the users. New entries are inserted into this database when new users sign-up or new presets are created (see Section 6.3). A special column in the users table called 'admin' holds a boolean value which indicates whether the user is an administrator or not. Administrator users can only be created directly in the database.

Administrators are the only users that can access the competition management pages on the website, all of them available in a drop-down menu in the navigation bar called 'Management'. Administrators also have access to a special guide in the guides section, which contains pretty much the same information as the following subsections. The guide on the website may even be better formatted and thus more readable.

At the time of writing, only one administrator account exists and it is named 'gfadmin'. If more administrator accounts are required, these will need to be created directly in the database.

6.1 Creating competitions

One can create a competition in the 'Create Competition' option of the 'Management' drop-down menu. The required information consists of:

- the competition name, which must be unique (using dates in the name may be a good idea) and CANNOT be modified after competition creation
- the start and end dates. It is only possible to specify a starting date in the future, and the end date must be further into the future than the start date
- a description, where it is possible to include details such as prizes for the winners, for example, and CANNOT be modified after the competition creation
- a competition world file, an XML file containing ALL levels (public and private) for a competition, along with their time limits, which CANNOT be changed after the competition is created

After creation, the competition should appear in the competitions page as an 'Upcoming' competition that only administrators can see. It will be visible for everyone once it starts.

If the submission handler isn't already busy, the level previews should take less than a minute to show up in the details page of the competition.

Competitions can't be deleted.

6.2 Editing competitions

There are two ways to access the edit page of a competition:

- through the 'Edit' button in a competition card in the competitions page (only visible to admins)
- by selecting the desired competition in the Manage Competitions page

Level specific parameters (bonuses, etc.) can only be changed before the competition starts. Time limits can't be changed as they come from the XML world file.

There's more flexibility for changing non-level parameters, such as number of simulations per level, maximum submission size, etc., as these can be changed while the competition is running.

Dates are also editable, for example, to extend a competition's duration or to delay its start. It is currently impossible to change the dates so a competition is immediately opened, but it is possible to immediately finish one. An normal scenario would be, for example, to create a competition starting the following day, edit it, maybe even run some presets and then wait a day for it to start.

Changing level specific parameters will eliminate the preset submissions for the given competition.

Changing level visibility (public, private) will often change the order of the levels, so that the public levels appear first.

6.3 Creating presets

Logically, presets are users which always submit the same files and always choose the same platforms. They are meant to be used as baselines.

A preset can be created in the 'Create Preset' page (selectable from the 'Management' drop-down menu, similarly to the way a submission would be created. The only difference is that a name has to be given to the preset, which is what will appear in the scoreboards.

Presets are reusable, meaning that once created they can 'participate' in multiple competitions, even in 'Upcoming' competitions (excludes finished competitions).

Presets can't be edited or deleted after creation.

6.4 Running presets

Logically, running a preset is like telling the preset user to submit their ZIP (source code) and PDF (report). This can be done in the 'Run Presets' page accessible from the 'Management' drop-down menu.

After selecting the desired competition, multiple presets can be selected using the CTRL key while clicking on their names in the select area.

As with any submission, it will take time for them to be processed. Their performance can be checked in the scoreboards. Presets will appear differently in the scoreboard, with a greyed out background.

More details can also be accessed in the preset's profile page, as the 'My submissions' section is visible to admins. This can be useful to detect errors with presets, or simply check out their processing status.

For upcoming competitions, changing level specific parameters will remove all preset submissions to avoid inconsistencies. The presets can simple be re-run afterwards.

The preset's submission zip and report will be publicly available even before the competition ends.

6.5 Updating the Geometry Friends game

The game may be updated in the 'Update Game' page accessible from the 'Management' drop-down menu. This feature is useful in case the Geometry Friends game changes in the future.

This is a dangerous thing to do, and the responsibility lies entirely with the administrator, as the server doesn't do any integrity check on what is uploaded.

Updating the Geometry Friends game version removes the old one, which is why:

- the new version must work as expected and meet the requirements listed below
- it is not possible to update the game version when there are open competitions, to avoid inconsistencies in the results
- no presets must be scheduled for simulation when updating the game

The format for a new Geometry Friends game version NEEDS to meet the following requirements:

- it needs to be a zip file containing a starting agent for the participants to be able to understand how they can begin to write their implementations
- alongside the files of the starting agent must be the game itself in a folder called GeometryFriendsGame/Release. Inside this directory is the executable and other libraries required.
- GeometryFriendsGame/Release also needs to contain static game resources in a folder called Content (the sub-folders Agents and Levels inside the Content folder will be REMOVED) and nowhere else
- other folders, such as Agents, Levels and Results, should be removed before uploading the new game version
- lastly, the new game version (the structure showed in Figure 8) must be in the base directory of the zip file, and not inside a directory

A Geometry Friends game folder may have a structure similar to what is shown in Figure 8 (some output is truncated).

The actual upload form should be at the end of the 'Update Game' page, where a ZIP containing the new game version can be uploaded.

```
.
├── GeometryFriendsGame
│   └── Release
│       ├── Content
│       │   ├── Common
│       │   │   ├── geometryfriends-logo.xcf
│       │   │   ├── *.png
│       │   │   └── ...
│       │   ├── Fonts
│       │   │   ├── Arista.ttf
│       │   │   ├── CARTOONIA.TTF
│       │   │   ├── *Font.spritefont
│       │   │   └── ...
│       │   ├── LevelEditor
│       │   │   ├── *.png
│       │   │   └── ...
│       │   ├── Sound
│       │   │   ├── *.wav
│       │   │   └── ...
│       │   └── XBoxLogoWithBorder.png
│       ├── FarseerPhysics.dll
│       ├── FarseerPhysics.pdb
│       ├── GeometryFriends.exe
│       ├── GeometryFriends.pdb
│       ├── OpenTK.dll
│       ├── OpenTK.dll.config
│       ├── OpenTK.pdb
│       ├── OpenTK.xml
│       ├── WiimoteInput.dll
│       └── WiimoteLib.dll
├── GeometryFriendsAgents.csproj
├── GeometryFriendsAgents.sln
├── Properties
│   └── AssemblyInfo.cs
├── CircleAgent.cs
├── RectangleAgent.cs
├── windows_packaging_script.bat
└── linux_and_macos_packaging_script.sh
```

Figure 8: Normal Geometry Friends directory structure (some output is truncated)

6.6 Generating submission videos

The system is capable of generating videos of any **handled** submission that didn't fail (e.g. that didn't get an error such as an extraction error, compile error, etc.). This is done in the scoreboard by clicking on the button (visible to admins only) in the 'Video Request' column.

Once a video is requested, it will take some time for the videos (one for each level of the competition) to be available in the expanded scoreboard of the submission.

If the competition isn't finished yet, only videos corresponding to the public levels will be made, which is why most of the time it will only make sense to request videos at the end of a competition.

Currently, this can only be done once per submission. This means that if videos are requested before the competition ends, there will be no way to get videos for the private levels.

When dealing with presets and competitions which aren't finished yet, it is possible to simply re-run the preset, wait for it to be handled and then request the videos again.

C

Evaluation Questionnaires

This appendix contains the three Google Forms questionnaires used for the three different test scenarios: competition management (CM) scenario, thorough user/participant (TU) scenario, and the simple user/participant (SU) scenario.

When viewed on the browser, each questionnaire has three pages/sections and a lot of colours. When exported to PDF, which is what will be shown here, the questionnaires take a more raw format.

C.1 Competition Management (CM) Questionnaire

Competition Management Test

*Required

Personal Information

All answers are anonymous.
Personal information collected here will be used ONLY for system quality evaluation purposes.
This test may take around 15 minutes to complete.
Thank you for participating!

1. **How old are you?** *
   Mark only one oval.
   - <18
   - 18-25
   - 26-35
   - 36-45
   - >45

2. **What's your gender?** *
   Mark only one oval.
   - Male
   - Female

3. **What's your occupation?** *
   Mark only one oval.
   - Student
   - Teacher
   - Other: _____

4. **What's your course (in case you're a student) or your field of study (in case you're a teacher/researcher)? If neither, you can refrain from answering.**

   _____

5. **Check the boxes that apply to you.**
Tick all that apply.

- [ ] I have participated in artificial intelligence competitions or programming contests before (e.g. Mooshak)
- [ ] I have knowledge of artificial intelligence in games.

Practical Part - Follow these steps

You may take your time reading and investigating the platform before each step.

0. All information required to complete the following steps is present in the website. When in doubt, you may take your time to investigate the website in search for help. You will often be asked to upload files throughout this test. All files you'll need are located in the "cm-test-files" folder in the provided flash drive. You'll also be given an ID number to use whenever the '<id>' tag appears in the questionnaire.

1. Head over to the GFGAI competition website and login as the administrator

The administrator credentials are:
- username: gfadmin
- password: *provided by the test supervisor*

2. Create a competition with the following properties:

- it should be called "Test Competition <id>"
- it should start tomorrow and end a week after tomorrow
- the world file to use is called "WorldFile.xml"
- give the competition any description you want.

3. Edit the newly created competition and make sure that:

- submissions are run once per level
- the level with a single diamond (the purple collectible) is private and the level with two diamonds is public
- the level with a single diamond has a completion bonus of 160 and a collectible bonus of 90
- the level with two diamonds has a completion bonus of 130 and a collectible bonus of 120

4. Create two presets with the following properties:

- one preset named "TestWindowsPreset<id>" which shall run on Windows; the zip to upload is named "WindowsPreset.zip" and the report is named "ExampleReport.pdf"
- one preset named "TestLinuxPreset<id>" which shall run on Linux; the zip to upload is named "LinuxPreset.zip" and the report is named "ExampleReport.pdf"
(Note: presets are reusable baselines that serve as a reference point for participants)

5. Run the two presets just created on the competition and report below on their status.

Write a short answer in your own words and include:
- their scores
- their state
- possible errors and error details

6. **Write below!** *

6. Generate a video of both submissions (if possible)

7. This part is over! If you wish, you can watch the video(s) after the system has made it/them available. Make sure you log-out! Proceed to the next section for a quick questionnaire.

Usability Questionnaire

7. **1. I think that I would like to use this system frequently** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

8. **2. I found the system unnecessarily complex** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. **3. I thought the system was easy to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. **4. I think that I would need the support of a technical person to be able to use this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. **5. I found the various functions in this system were well integrated** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. **6. I thought there was too much inconsistency in this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. **7. I would imagine that most people would learn to use this system very quickly** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

14. **8. I found the system very cumbersome to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. **9. I felt very confident using the system** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

16. **10. I needed to learn a lot of things before I could get going with this system** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

C.2 Thorough Participant (TU) Questionnaire

User Interaction - Thorough Test

*Required

Personal Information

All answers are anonymous.
Personal information collected here will be used ONLY for system quality evaluation purposes.
This test may take around 15 minutes to complete.
Thank you for participating!

1. **How old are you?** *
   Mark only one oval.
   - <18
   - 18-25
   - 26-35
   - 36-45
   - >47

2. **What's your gender?** *
   Mark only one oval.
   - Male
   - Female

3. **What's your occupation?** *
   Mark only one oval.
   - Student
   - Teacher
   - Other: _____

4. **What's your course (in case you're a student) or your field of study (in case you're a teacher/researcher)? If neither, you can refrain from answering.**

   _____

5. **Check the boxes that apply to you.**
Tick all that apply.

- [ ] I have participated in artificial intelligence competitions or programming contests before (e.g. Mooshak)
- [ ] I have knowledge of artificial intelligence in games.
- [ ] I have used either Visual Studio or MonoDevelop before.

Practical Part - Follow these steps

You may take your time reading and investigating the platform before each step.

0. All information required to complete the following steps is present in the website. When in doubt, you may take your time to investigate the website in search for help. You will often be asked to manipulate and upload files throughout this test. Some files will be obtained through interaction with the system, while others will be located under the "tu-test-files" folder in the provided flash drive.

1. Head over to the GFGAI competition website and create a brand new account.

You can use any details you like, but you'll need to remember your username and password.

2. Login with your new account

3. Participate in a competition named "Static Test Competition". Use the helpful information on the website to guide you through all the steps, which are summarized below:

3.1 Download and extract the right competition package

You may use any directory you like on the machine you're using for this test.

3.2 Edit the solution in Visual Studio or MonoDevelop.

Make sure you import your solution correctly
For this test case, simply replace line 148 of the CircleAgent.cs file with the contents:

currentAction = Moves.ROLL_LEFT;

3.3 Build and test your solution locally. Did it pass the level?

Report on the outcome of your test. If your agent didn't pass the level, answer with 'Other' and write down what you think went wrong.

6. **Answer below!** *

Mark only one oval.

- Yes
- Other: _____

3.4 Pack your submission using the appropriate packaging script that came in the competition package

In Windows, double-clicking "windows_packaging_script.bat" should be enough
Anywhere else, the "linux_and_macos_packaging_script.sh" should work

3.5 Submit the generated zip to the competition. For the report PDF, upload the "ExampleReport.pdf" file that should be located under "tu-test-files" in the provided flash drive.

4. Try to follow the status of your submission. What happened to it?

Write a short answer in your own words and include:
- the submission's score
- its state/status
- possible errors and error details

7. **Write below!** *

_____
_____
_____
_____
_____

5. This part is over! You can now log out. Proceed to the next section for a quick questionnaire.

Usability Questionnaire

8. **1. I think that I would like to use this system frequently** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. **2. I found the system unnecessarily complex** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. **3. I thought the system was easy to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. **4. I think that I would need the support of a technical person to be able to use this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. **5. I found the various functions in this system were well integrated** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. **6. I thought there was too much inconsistency in this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

14. **7. I would imagine that most people would learn to use this system very quickly** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. **8. I found the system very cumbersome to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

16. **9. I felt very confident using the system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

17. **10. I needed to learn a lot of things before I could get going with this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

C.3 Simple Participant (SU) Questionnaire

User Interaction - Simplified Test

*Required

Personal Information

All answers are anonymous.
Personal information collected here will be used ONLY for system quality evaluation purposes.
This test may take around 10 minutes to complete.
Thank you for participating!

1. **How old are you?** *
   Mark only one oval.
   - <18
   - 18-25
   - 26-35
   - 36-45
   - >45

2. **What's your gender?** *
   Mark only one oval.
   - Male
   - Female

3. **What's your occupation?** *
   Mark only one oval.
   - Student
   - Teacher
   - Other: _____

4. **What's your course (in case you're a student) or your field of study (in case you're a teacher/researcher)? If neither, you can refrain from answering.**

   _____

5. **Check the boxes that apply to you.**
Tick all that apply.

- [ ] I have participated in artificial intelligence competitions or programming contests before (e.g. Mooshak)
- [ ] I have knowledge of artificial intelligence in games.

Practical Part - Follow these steps

You may take your time reading and investigating the platform before each step.

0. All information required to complete the following steps is present in the website. When in doubt, you may take your time to investigate the website in search for help. You will often be asked to upload files throughout this test. All files you'll need are located in the "su-test-files" folder in the provided flash drive.

1. Head over to the GFGAI competition website and create a brand new account.

You can use any details you like, but you'll need to remember your username and password.

2. Login with your new account

3. Submit an agent to a competition named "Static Test Competition".

Consider that:
- your submitted agent will run on Linux
- the zip file you upload is "LinuxSubmission.zip"
- the report you upload is "ExampleReport.pdf"

4. Try to follow the status of your submission. What happened to it?

Write a short answer in your own words and include:
- the submission's score
- its state/status
- possible errors and error details

6. **Write below!** *

5. Place a new submission to the same "Static Test Competition".

This time, make sure that:
- the agent you're about to submit was developed on Windows
- the zip file you need to upload is "WindowsSubmission.zip"
- the report you need to upload is "ExampleReport.pdf"

6. In a small sentence, describe what happened to your first submission after you placed a new submission.

7. **Write below!** *

7. Once again, try to follow the status of your new submission. Try to describe with your own words what happened to your submission

Write a short answer in your own words and include:
- the submission's score
- its state/status
- possible errors and error details

8. **Write below!** *

7. This part is over! You can now log out. Proceed to the next section for a quick questionnaire.

Usability Questionnaire

9. **1. I think that I would like to use this system frequently** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. **2. I found the system unnecessarily complex** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. **3. I thought the system was easy to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. **4. I think that I would need the support of a technical person to be able to use this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. **5. I found the various functions in this system were well integrated** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

14. **6. I thought there was too much inconsistency in this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. **7. I would imagine that most people would learn to use this system very quickly** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

16. **8. I found the system very cumbersome to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

17. **9. I felt very confident using the system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

18. **10. I needed to learn a lot of things before I could get going with this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree