Graphgames User Manual

Gian Paolo Jesi

[gpjesi@luiss.it]

Table of contents

Introduction....1
Design and architecture....2
Installation....3
Usage....4
  Changing the ‘admin’ user password....4
  Basic user....4
  Advanced user....5
    Register an advanced user....5
    Users page item....5
    Sessions page item....6
    Games page item....6
    Data file page item....7
Example: setting up a new game configuration....7

Introduction

Graphgames is a novel platform dedicated to behavioral or cognitive games. By playing these kind of prototypical games, we aim to study how human beings learn, adapt and possibly develop new strategies. In addition, we are interested in understanding what are the conditions that foster or limit the cognitive process.

The aim of Graphgames is to lower the burden for non tech-savvy researchers in order to access the IT infrastructure required to deploy experiments, play and collect data for their research goals.

We envision three distinct use cases which correspond to three kind of users:

- basic users: these are the game players, they just have the access to the available games and to their play sessions. They can play from everywhere
- advanced users: they are typically researchers. They can setup experiments by preparing customized instances of platform ready games. They have access to all the data collected by their game instances
- pro users: they are typically researcher with a robust computer science knowledge. These users can implement and deploy new games to the community.

Solitary games and games between multiple users are supported. In addition, the platform features an AI layer that runs automatic players or bots. Bots can be used for test purposes of game models or to test human behavior facing a specific schema or strategy. On the contrary, bots can learn how to play from human players by analyzing the play data (e.g., through machine learning algorithms).

The name 'graphgames' tell us something about the nature of the games that the platform is dedicated to. Essentially, games are represented as graphs. These graphs can be quite large since they are made by all the possible game states. More precisely, the game states correspond to the graph nodes, while the relation that links a node A with a node B is represented by an arc. In general, the relation is presented by a game move.
A graph like representation is a mathematically sound structure that makes analysis convenient from a theoretical and computational perspective. Any game that can be expressed as a graph made of states and moves can be implemented in Graphgames.

Graphgames is designed as a conventional web application. By adopting such a standard way of interaction (through the web) no fancy software have to be installed on each laboratory PC: any recent web browser is sufficient.
Being web enabled, the platform is not limited by the size of a typical behavioral laboratory, but can scale arbitrarily by allowing the access of remote players, even from mobile devices. This feature multiplies tremendously the amount of data that is possible to collect by leaving 24h/24h accessible games over the Internet. Of course, this might not be feasible for any kind of game, but nonetheless it represents a valuable opportunity.

The platform distribution comes equipped with a built-in game implementation: Target The Two (TTT), a well known cards game in the behavioral community.

There are several ways in which our software can be run and exploited. We made our best to make the installation process as simple as possible, trying to accommodate distinct usage profiles, such as personal, group-aware, lab-aware or Internet oriented. The first option targets a single researcher who wants to install the software on its own machine (i.e., laptop) for evaluation and for designing experiments. In this case, a ready to use virtual machine (VM) is provided. The VM runs on any OS and requires VirtualBox[1]. Since many features requires specific knowledge and particular access rights, these are normally disabled; however, the basic game play and data gathering functionalities are always present.
The other usage targets are much more complicated since requires a low level installation and configuration on a Unix OS (e.g., Linux or MacOS) and at least mid computer science skills are required. However, the details about both installation cases are discussed in the following sections of this document together with some consideration about security.

The remainder of this document is organized as follow: we give first a few insights about the design and architecture of our framework, we give simple instruction to install the software and we provide guidelines to understand and use the basic features of Graphgames. Finally, we provide a simple example in order to generate a new game configuration.

1 A free virtualization software available for any operative system; https//www.virtualbox.org .

Design and architecture

This section provides a brief technical overview of the design and technical choices adopted by our framework. It is written for the tech inclined readers, but it can be skip if not interested.

The Graphgames platform is designed as an advanced web application. Basically, a web interface is an obliged choice when the priority is accessibility and ease of interaction and deployment. Web technologies are now a viable route also for very interactive, distributed and computationally intensive applications. However, many web technologies are in a continuous flux of changes (e.g., Javascript syntax and features changes, web-frameworks appear and disappear quickly, etc,…) or has a very low abstraction layer (e.g., HTML, CSS). For these reasons, we decided to limit the direct use of traditional web technologies and tools at the bare minimum and we opted to operate at a higher level which is more suitable when looking for stability and robustness from a software engineering point of view. In addition, a higher level of abstraction allows to better focus on the architecture which is highly desirable when the application goes beyond the basic web-site functionalities.

Essentially, the Graphgames core is developed in Python using a Python-based web framework (Flask), while the client side of the games such as the provided TTT implementation is pure Javascript. The we application graphical layout is provided out of the box from a well know web-frontend framework (i.e., Twitter Bootstrap).

In order to overcome the typical limitation to real-time interaction of the web, Graphgames adopts web-sockets to communicate asynchronously with game clients. Web-sockets greatly simplifies communication: actors exchange text messages (encoded in JSON); the actual protocol between the server and game clients is not defined a priori by the platform, but a game developer is free to design its own. The only requirements is that the game server part has to implement a specific Graphgames interface (API) in order to integrate into the system.

When computationally intensive task or bot behavior comes into play, these cannot be run inside the web app for performance and scalability reasons, but another mechanism is required. Graphgames uses a distributed queue where the heavy tasks are actually run. These mechanism allows running thousands of distinct bots and other tasks without loosing responsiveness on the traditional web interface. The platform uses the (IBM) Redis DB as queue broker and Celery as queue worker.

As any data-driven web application, the relational database part which stores users, moves, sessions and all the details of the application data model is totally abstracted. Any SQL DB can be adopted with Graphgames. Migration of the data model is fully supported[2]. It means that if the data model will change in a future release and the database must be regenerated, all the data previously collected (e.g., users, sessions, … everything on the DB) will not be lost. This feature is quite substantial since data collection in Graphgames is fundamental.

2 Migration works on fully fledged database systems such as: MYSQL, Postgres or

Installation

The installation process is no longer mandatory, since we provide our Graphgames web site: http://graphgames.luiss.it/ .

In the following we provide the instruction for the simplest and most common installation process: the virtual machine installation for personal use. The VM is self contained and all required services are configured and running.

We suppose the user downloaded and installed VirtualBox for his/her machine OS. Please, just follow these steps:

1. download the newest available GraphGames VM image -URL-
2. double click on the downloaded Graphgames image: VirtualBox will open, asking to import the VM. Just accept.
3. Wait for the process import to finish
4. From the VirtualBox window click on the Graphgames VM: wait for the VM booting
5. The VM shows a minimal Linux desktop. Open the browser (Mozilla) by clicking its icon on the left of the top bar
6. In Mozilla (or any other browser of your choice), go to this url: localhost:5000 , now you should see the Graphgames homepage

Usage

This section explains how to use the platform from the point of view of the possible kind of users. They are respectively addressed by a distinct sub-section.

After installation, Graphgames has just a single administrative user: ‘admin’, whose password is ‘adminpw’. You are highly encouraged to change the admin password to a new and very robust one; if the installation is server based and the Graphgames instance is Internet accessible, you absolutely have to[3].

Changing the ‘admin’ user password

Login as ‘admin’ and enter the default password ‘adminpw’. Click in user symbol on top right, then follow the link ‘Change password’. Re-enter the old password (‘adminpw’), then choose and type the new one[4] in both text slots. Remember to choose a robust password and store it in a safe place. Click ‘save’ button and logout from the admin user account.

3 Please, do not underestimate the security aspect of network services especially nowadays.

4 The minimal requirements for a robust password are: no less than 8 characters, at least one digit, at least one capital letter and one non alphabetical character, such as: ?,+*;(/^.

Basic user

The first step when using Graphgames is to login into the system. However, if it is the very first time using the software, the user must register as a valid system user. This is true for any kind of user (e.g., simple, advanced or pro).

Following the Register link (see), the user have to enter a valid email address and a password (the password must be entered twice as a confirmation). The user's email is adopted as a user-name. The password cannot be empty and, of course, choosing a robust password is highly encouraged.

After registration, the user should be already logged in. If it is not, then login manually by following the Login link on the Graphgames homepage.

After login, the login name or user-name appears at the top right corner. The system homepage now shows the available games in the system. The game description is an active link that will directly bring the user to the corresponding game instance.

The page top bar is populated by just two items: the Graphgames home and 'My moves'. The latter item brings to the current user's collection of moves accumulated over time from each played game instance.

Advanced user

As any other user, an advanced user has to first register and then login into the system.

Since advanced users can define new game instances and accessing to the game data (e.g., configuration, files, data, files, moves, etc..) the page top bar is populated with more items corresponding to more features. The extra items available are: 'Users', 'Sessions', 'Games', 'Data files'. Their usage will be discussed in this section.

Register an advanced user

The main practical difference of an advanced user compared to a basic one is that he/she has extra rights or a deeper access in the systems. In fact, users have *roles* and while basic users have just the 'user' role, advanced users has both 'user' and 'admin' roles. However, when registering a new user from the web interface, its role is limited to user for basic security concerns. In order to upgrade it, the administrator user (admin) comes into play.

For a local installation, it is sufficient to login as admin and upgrading the roles for the new user(s). For a more sophisticated installation, if we do not have access to the admin user password, we have to ask to the system administrator.

Now, suppose we are logged as the admin user, the following steps are for upgrading a user's role:

1. select the 'Users' item on the page top bar. A table with all system users will appear
2. find the row corresponding to the user to upgrade and click on the 'Roles' element (e.g., is the one corresponding to the 'Roles' column). A small window will appear over the table allowing to change the list of roles for the selected user

3. click at the right side of the current role (e.g., ‘user’) and a list of possible choices will appear; choose ‘admin’ and confirm by clicking the blue button with the mark
4. the new user is now an advanced one and you can logout from the admin

Users page item

This page presents a table populated with all users and their main details. From this administration page any advanced user can manage system users.

Any record (row) can be edited or deleted by choosing the corresponding button[5] on the left part of the row. An operation, such as delete, can be operated on multiple records be choosing ‘with selected’ tab on top of the table. Each record have to be selected before, by pressing the button on the left of each row.

User can also be created by choosing the ‘Create’ tab on top of the table. A window will appear on top of the table allowing the data entry process.

The table records can be filtered by choosing the ‘Add filter’ tab on top of the table. After clicking a drop-down list will appear showing the available fields for filtering. After choosing a field, a new graphical element will appear on top of the table with which the user can specify a filter over the selected field.

Sessions page item

The sessions page shows a table populated with all the played game sessions. Each session has an id and it is linked to a user (uid) and a type of game (type).

Each record can be deleted or replayed by choosing the corresponding button.

A multiple selection of records can be deleted at once or downloaded by choosing ‘With selected’ tab on top of the table.

Downloading one or more sessions means downloading all moves played during the selected sessions (e.g., the exact database content of the ‘moves’ table). However, each game can set a specific policy with which to download the data; in this manner, the data can be filtered before being exported in order to better fit with other specific tools (e.g. statistical software package).

The basic format adopted is CSV (i.e., Comma Separated Values).

In the particular case of the TTT game, an additional format – Nemik (which is clear text based) - is also provided. In fact, it depends of the actual implementation of each game.

In order to actually download sessions, the user has to select the rows of interest by clicking the tick button of the left of each row and then select the tab ‘with selected’ on top of the table followed by the preferred download option available.

The same procedure is adopted when deleting sessions (e.g., by choosing ‘With selected → Delete’).

5 Respectively, the pencil or the trash basket buttons.

The actual file downloaded is a zip archive file. The actual data are inside it and the file must be decompressed. The choice of using a zip archive is to address the chance to have the data split over multiple files (e.g., as in the case of Nemik format).

The sessions table can be filtered according to session id, user and game type by clicking the ‘Add filter’ tab on top of the table. This filtering mechanism is for visualization only: it does not address the data download process.

Games page item

The games page shows the available games in the system. More precisely, we consider games as game configurations. From a specific game, such as the TTT, an advanced user can setup custom configurations by setting parameters and particular configuration files trying to achieve a particular goal.

The creation of a new game can be started by choosing the ‘Create’ tab on top of the table. A new window will prompt for game parameters and a description. The former is a record written in JSON syntax that describes the parameters of the games, while the latter is a text description that will be shown on the system homepage.

The actual parameters required to setup a configuration is game dependent. Each game defines its set of configurable parameters and it is up to game developers to provide the required documentation. However, there are also system parameters which are always mandatory such as:

- “html_file”: the game page (html) location relative to the application root (e.g., “html_file”: “admin/games/ttt-page.html”
- “replay”: a boolean triggering the replay feature on or off, if available

Essentially, the bare minimum JSON string for the parameters is the following:

```
{“html_file”: “admin/games/ttt-page.html”, “replay”: false}
```

Each record (game configuration) can be edited, deleted and played by choosing the corresponding button on the left part of each corresponding row.

Data file page item

The data file page shows the content of the system file repository. Any kind of file can be renamed or deleted as well as downloaded or uploaded. Folders can also be created at any deep.

Security note: all files belongs to the user which which the Graphgames application is running on the local or server machine. For brevity, this means that any advanced user has access to any other advanced user’s file. This is a security hazard since user’s responsibility and carefulness is the only measure to prevent mistakes especially when the users base is large.

Example: setting up a new game configuration

In this example, we are going to setup a new game configuration based on TTT. First, we need to login as an advanced user: we suppose our login is already setup and we already have the ‘amin’ role.

After login, click the ‘Games’ tab on the page top bar. In order to prepare a new game configuration click the ‘Create’ tab in top of the table. A new window labeled “Create new record” will pop up over the underlying page.

The windows asks for the “params” and “info” fields. Let’s start from the latter in this case. Enter “Example TTT” in the info field. Then, click on the “params” fields and add the very minimum. Remember that these fields requires a JSON syntax:

```
{“html_file”: “admin/games/ttt-page.html”,
“replay”: false, “enable_multiplayer”: false,
“enable_bot”: false}
```

In other words, we specify where is the html page where the TTT game is hosted (i.e., more technically, the html file which links the Javascript code implementing the client side of the game) and we switch off the “replay” feature. This feature allows a user to visualize a replay of any game session: each move is played one at a time recreating the original time gap between moves. Finally we disable the multi-player feature as well as any artificial intelligence (AI) robot.

Then we need to add game specific parameters. In the particular case of TTT we add the following:

- “shoe_file”: is the file holding the cards layout and player turn for each hand of the TTT game. A clever making of this file is the key for interesting experiments. Out of the box, our TTT implementation provides two distinct shoe files: a small 3 hands file (games422-small.txt) and a full size file with 40 hands (games422.txt). These files are located in the “data” application folder which is accessible through the “Data files” tab located on top of the page bar.
- “opponent_covered”: a boolean (“true” or “false”) selecting if the opponent player card must be flipped (covered) when the turn changes.
- “covered”: this parameters requires a record (object) structure, which is essentially a set of key-value pairs. Essentially, it states the cards locations that must be covered from turn to turn. “CK”, “C”, “NK”, “N”, “U”, “T” are the keys corresponding to the TTT card locations[6] and their value is a boolean.

Essentially, the “params” string should become as follows:

```
{“html_file”: “admin/games/ttt-page.html”,
```

6 The keys corresponds to: Color keeper, Color, Number Keeper, Number, Up and Target.

```
“replay”: false,
“enable_multiplayer”: false
“enable_bot”: false
“opponent_covered”: true,
“shoe_file”: “games422.txt”,
“covered”: {“CK”: false, “C”: true, “NK”: false, “N”: true,
“U”: false, “T”: false}
}
```


After filling the window fields, press the blue “Save” button. The new configuration is ready and has been appended to the games table. It can be checked easily by pressing the start button in its corresponding row. The new configuration is now visible as a new game by any (registered) user from the system homepage.