**PowerGrid documentation**

**An overview of the PowerGrid project, how it works, and what the differences are with other Runescape bot clients.**

****

Table of Contents

Chapter 1 – Introduction 3

Chapter 2 – Overview of PowerGrid functions 4

2.1 – Artificial Intelligence as a way of playing Runescape automatically 4

2.2 – Communication with the Runescape client 4

Chapter 3 – Structural overview 5

3.1 – Structural diagram of PowerGrid 5

3.2 – Summary of each of the modules in PowerGrid 6

3.3 – The PowerGrid execution cycle 6

# Chapter 1 – Introduction

Many bot clients already exist for the MMORPG Runescape. Each with their own benefits and downsides. However, almost all bot clients have one thing in common: they offer little to no functionality by themselves, and only provide that which is required to perform basic operations on the bot. A large downside of this is, that this results in bot clients not being able to operate in a dynamic environment, and such clients often also do not provide tools to recognise certain object types, leading to manually checking and comparing id values over and over again.

A solution to the abovementioned problem would be, to provide the missing tools as an extension to an existing bot client, but such an approach also has a price in the form of a very high memory footprint, since much data needs to be cached locally too. This is mainly because almost all clients are closed-source, and getting the necessary data often requires an impractical detour through a variety of method calls.

So the best solution would be to create a bot client from scratch, focussed on solving the above problems and to optimize user experience in this way. However, there is another problem a large number of bot client cope with: It is only possible to play the game in fixed resolutions, because of the way the widget system in these bot clients work. Since most bot clients essentially operate on the same core, these bot clients all deal with the same problems, and as such an entirely new and revolutionary approach must be taken in order to avoid the same problems and issues.

In order to optimize speed and to make it easier to remain undetected as a botting client, it would be a good design choice to implement the project in a programming language other than Java. As such, a programming language was chosen that resembles Java in many ways, but is a lot faster and more dynamic. This chosen programming language was C++.

Putting all the above together, we present you PowerGrid. A revolutionary, open-source bot client that intends to make life easier for everyone by providing functionality that automatically classifies and stores the data from the runescape world in native (C++) objects. Because of this caching behavior, it suddenly becomes possible to plan routes across the entire world, or find the nearest object matching certain criteria even if such an object is far away.

The final goal of PowerGrid is to provide users with a tool that can play Runescape completely by itself, automatically deciding on the tasks to perform based on changes in the environment. PowerGrid will even be able to perform abstract tasks like leveling a certain skill to a certain level, or making a certain amount of money. PowerGrid should then automatically decide on the concrete tasks (what methods of money-making to use, or what method to use to train the requested skill) by effiency.

Please do note, however, that the abovementioned behavior is merely an indication, and exact functionality may change over time.

# Chapter 2 – Overview of PowerGrid functions

This chapter describes the various PowerGrid functions and how these contribute to each other to make PowerGrid work the way it does. PowerGrid basically consists of two main components: the Artficial Intelligence and the communication between the Runescape client and PowerGrid. A description of the Artificial Intelligence module is provided in paragraph 2.1, whereas the communitication with the Runescape client is described in paragraph 2.2.

## 2.1 – Artificial Intelligence as a way of playing Runescape automatically

PowerGrid has a sophisticated Artificial Intelligence (AI) module, which it uses to decide on actions from the Runescape environment. Because PowerGrid uses such an AI module, it is possible for PowerGrid to operate even in unknown environments and behave dynamically based on the state of the environment and the goals set by the user.

The goals the user can provide to the AI are high-level descriptions of tasks, such as to reach a certain location, or to acquire a certain amount of experience for a skill. The AI combines this information with the environment to decide on an action to perform.

For example, reaching a certain location requires the AI to execute a pathfinding algorithm and then follow the path this algorithm produced. As the AI navigates this path, it should still monitor the environment because some things are prone to change. Imagine that the path leads through a door but that door is closed. The AI should then automatically know that the door should be opened before continuing its path.

For gaining experience in various skills the amount of elements to monitor is even bigger, and the approach varies greatly between different skills. Training a skill such as crafting, for example, requires a completely different approach then training a skill such as woodcutting. Therefore an AI is required that can somehow decide on the action based on its goals.

## 2.2 – Communication with the Runescape client

The AI by itself cannot do much good if there is no connection to the Runescape environment itself. As such, a system is needed that ensures a correctly set up connection to the Runescape client. Since Runescape runs on Java, we need to communicate with the Java Virtual Machine (that runs the Runescape client) first. Java provides a system for this called JNI (Java Native Interface). However, the functionality provided by it is not sufficient, and JNI only understands very low-level instructions. As such, Powergrid contains a system that mediates between instructions and information the AI can understand, and the low-level instructions that the JNI can understand. This bridges the gap between C++, the language PowerGrid is written in, and Java, the language of the Runescape client.

# Chapter 3 – Structural overview

In this chapter, an overview is given of the basic structural components of the project and the relations between these components. First, in paragraph 2.1, a diagram is provided that shows the structure of PowerGrid while in action. This is followed by a summary of each module shown in the diagram (paragraph 2.2). Lastly, a description is given about PowerGrid’s execution cycle in paragraph 2.3

## 3.1 – Structural diagram of PowerGrid

The below diagram illustrates the structural diagram of PowerGrid. The diagram shows the different modules in PowerGrid and how these modules interact with each other.

The modules are split up in two groups: The Java Virtual Machine and the PowerGrid client. The Java Virtual Machine contains the running Runescape client as well as access to the JNI functions that PowerGrid uses. PowerGrid itself contains various modules to monitor, process and inject actions and information from and to the Java Virtual Machine.



## 3.2 – Summary of each of the modules in PowerGrid

**JNI module**

Handles basic interaction with the JVM. Can read data from, and write data to the running Java Virtual machine through JNI.

**Monitor module**

Handles incoming raw data from the JNI module and parses it as recognised (native) objects.

**Caching module**

**Stores and manages the data from the Runescape environment for quick access.**

**GUI module**

Handles parsing and configuring the AI module with the information provided by the user.

**AI module**

Decides on an action based on parameters from the GUI module and information from the Monitor module. This is the core of the bot itself that makes the decisions.

**Injection module**

Translates an action from the AI into a (set of) events that can be injected into the JVM. These event objects are then given to the JNI module that will pass them to the JVM.

**JVM core**

The core of the Java Virtual Machine. Manages the basic functions of the running Java Environment. It provides a safe wrapper around the JNI functionality provided by the JVM.

**RS client**

The running Runescape client. Interaction with this environment can only be done using reflection functionality provided by the JVM core, which in turn can be accessed through JNI.

## 3.3 – The PowerGrid execution cycle

PowerGrid has four modules (JNI, Monitor, AI and Injection modules) that pass information to each other in a cycle. This is the contiuous cycle that PowerGrid runs in. An overview of how this cycle works is given below.

First, information is fetched through JNI. This information is parsed by the monitor (and then stored in the caches). Then, the AI module retrieves this information and uses it to decide on an action. The AI may also offer directives to the monitor on what to monitor, what to remember, and what to delete. This improves the hit ratio of cached data. When the AI decided on an action, It is sent to the injector module, which converts (breaks up) the action into separate JNI calls and sends these commands to the JNI module that will execute them on the Java Virtual Machine.

The GUI module provides end users with the possibility to modify how the AI works. When a setting is changed in the GUI, this change is propagated to the AI module, which will change its behavior accordingly.