**PowerGrid documentation**

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

****

Table of Contents

Introduction 3

Structural overview 4

# 

# 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 needed 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 works. 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.

# Structural overview

In this chapter, an overview is given of the basic structural components of the project and the relations between these components. The below diagram illustrates this, followed by a sumary about each module

JNI Objects

Java Objects

RS client

Events

Parsed Data

Actions

Parameters

Raw Data

GUI module

Monitor module

AI module

User Input

PowerGrid

Injection module

JNI module

JVM core

JVM

**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.

**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.

**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.

**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.