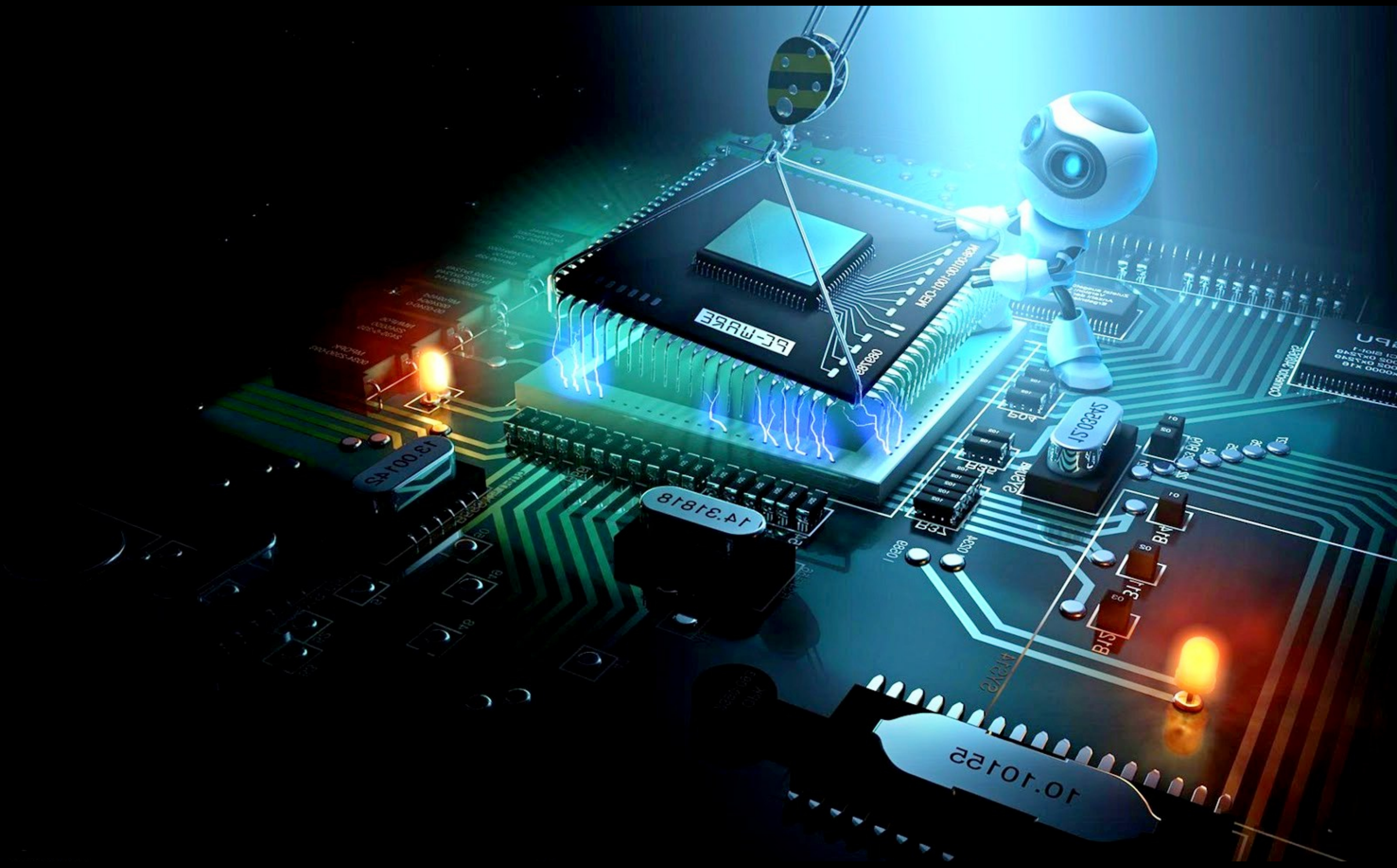
JAROSŁAW MACIEJEWSKI

TECHNICAL ASPECTS OF CREATING A BOT IN THE VIRTUAL WORLD OF SECOND LIFE



2021

Version 1

Original title: Techniczne aspekty tworzenia bota w wirtualnym świecie Second Life

English title: Technical aspects of creating a bot in the virtual world of Second Life

DOI 10.5281/zenodo.4729025

Version 1

Poland, 2021

Table of Contents

1. Introduction

Second Life is a virtual world, accessible via the Internet, where different people meet together. Despite the small number of active users in relation to registered accounts, the project is still alive. As everywhere on the Internet, there are bots - computer programs that have specific tasks.

In this e-book you will find information about Second Life, creating your own bot, interesting projects and solutions. Additional materials can be attached to the e-book, which will make your work easier.

The e-book is currently translated into Polish and English, so I matched some things in both documents:

- the US dollar (USD) is the default currency,
- the numbers are written in English.

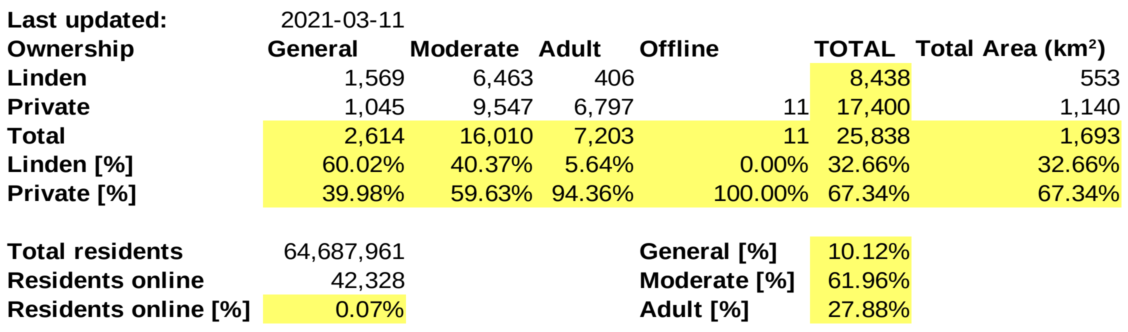
In the future, it is planned to first release loose extensions to this e-book, and then in the next version to combine them with the e-book into one whole.

Skills you need to work with this e-book:

- knowledge of LSL,
- knowledge of PHP,
- basics of getting around Second Life, starting a group, managing a group, etc.

2. Second Life, bots

Second Life is an internet project developed since 2003 by Linden Research Inc., which allows you to create a virtual world together. It may be a bit like The Sims game, but it differs from this game in that we control our character (avatar), who does not have any skills, statistics, or profession. Second Life is built on the principle of a grid of servers, on which each of them runs several regions (virtual lands). As for the virtual lands in Second Life, according to data downloaded from the portal gridsurvey.com from 2021-03-11[01], their total area is approximately 1,700 km^2, most of the lands are under private control and labeled 'Moderate'. While there are about 65 million registered accounts in Second Life, only about 0.1% of them are active.



Last updated:	2021-03-11					
Ownership	**General**	**Moderate**	**Adult**	**Offline**	**TOTAL**	**Total Area (km²)**
Linden	1,569	6,463	406		8,438	553
Private	1,045	9,547	6,797	11	17,400	1,140
Total	2,614	16,010	7,203	11	25,838	1,693
Linden [%]	60.02%	40.37%	5.64%	0.00%	32.66%	32.66%
Private [%]	39.98%	59.63%	94.36%	100.00%	67.34%	67.34%
Total residents	64,687,961			**General [%]**	10.12%	
Residents online	42,328			**Moderate [%]**	61.96%	
Residents online [%]	0.07%			**Adult [%]**	27.88%	

Data on lands and residents from Second Life

To get into Second Life, you first need to create an account on page[02] and download and install the computer program. You can install the official program[03] or any of the supported third-party programs here, such as Firestorm Viewer[04].

Second Life does not have the world climate imposed from above, although most locations resemble those known from real life, you can nevertheless find locations in the atmosphere of the Middle Ages, fantasy, antiquity or sci-fi.

After logging in to Second Life, it is best to start with the configuration of the program itself (keyboard shortcut CTRL + P) to set the program for yourself, and then get to know the program's interface, e.g. how to walk, fly, build, chat voice, etc.

01 Website: http://www.gridsurvey.com/
02 Website: https://join.secondlife.com/?lang=en - US
03 Second Life installer: https://secondlife.com/support/downloads/
04 Firestorm Viewer installer: https://www.firestormviewer.org/

As a virtual character, we can interact with other avatars (e.g. through a voice or chat conversation) and objects.

The program itself contains simple tools for building the world. The basic material is prim (colloquially called 'block'), while more advanced structures can be created and imported from Blender (so-called Mesh). Prim combined with other prims and meshes to create an object. There can be a maximum of 255 such connections with one prim or mesh. To animate a prim, mesh or an object, connect an LSL script to it, which is a combination of languages like C++, C#, Java and which is compiled by mono.

Of course, having a tool for building the world at your disposal, a basic knowledge of the LSL language does not mean that Second Life offers unlimited possibilities. Each region has a limited number of prims stored, usually a maximum of 15,000 or 30,000 prims. You can include the maximum size of one script, which is 64KB, and the maximum number of avatars in the entire region of 100 people.

Bot (other names include, for example: web robot, internet robot, agent, scripted agent, NPC (Non-Player Character), softbot) is a computer program that usually performs tasks faster and easier than human being, does not have a physical body, can have a virtual body, emulates human activity on the Internet, e.g. conversation with another human.

In Second Life, bots can be divided into two types:

- prim-mesh - the bot is created using prims, objects and meshes and scripted only with LSL,
- avatars - the bot is launched as a regular avatar from an external computer, communication takes place via a programming interface that can be mixed with LSL.

Second Life's bot policy is transparent. The bot can basically do everything normal people do in the form of avatars, however it cannot generate too much artificial traffic on the plot, copy stuff without looking at copyright, send more than 5000 messages per day, group messages are counted as 1 message for 1 recipient, buy MainL and region, other things mentioned in the Terms of Service, so basically the bot can do anything according to the principle: 'what is not prohibited is allowed'.

Bots in Second Life deal with various tasks assigned to them, e.g. they can simulate air traffic (ATC - Air Traffic Control), manage a region (manger, security guard), manage a group, work as hostesses, model on the catwalk showing clothes for sale, play as an actor in a mini-game, etc.

3. Comparison of bot building software in SL

There are several projects on the Internet that allow you to run your own bot in Second Life. You should consider how we will run our bot:

- Whether it will run around the clock or for several hours on certain days,
- Will it run on our home equipment or will we rent a server,
- Whether we want to use free or paid software.

Running a bot on your own hardware has the advantage that we have the bot and the computer under our control all the time, at our home. We definitely have to be ready for electricity and Internet charges as well as for the repair or replacement of damaged or worn-out computer components as a result of operation. We can allocate any old computer to the computer, as long as it meets the minimum requirements, you can buy a used SFF computer[05], laptop, netbook.

When deciding to run a bot on a remote server, eg VPS, we have remote access to the server and the bot, we must be prepared for a recurring fee for renting the server.

With VPS, we have to look for a good offer from a company that provides a VPS server rental, so I do not want to analyze offers here, because they may be out of date very quickly. Just enter the phrase 'vps linux'[06] or 'vps windows'[07] in the search engine depending on which system we are looking for VPS.

Below is a list of software to run your own bot in Second Life:

- **SmartBots** - the bot is hosted only on the company's server, the cost of renting the bot is: \$0.32/week for the basic version (with the possibility of extending it with additional components) and \$1.92/week for the full version. SmartBot allows you to invite visitors to a group, send an announcement at a specific time to multiple groups, monitor a group chat via a web browser, add spammers to the blacklist, create checkpoints through which the bot will pass (useful e.g. during trips), etc. Communication with bot is done via: LSL, HTTP. SmartBots works with a fairly extensive list of third party software, eg for lease (e.g. CasperLet, HippoRent), sales (eg CasperVend), advertising (eg IntelliAd). Web address: https://www.mysmartbots.com/.

05 SFF - Small Form Factory
06 Google Search with the phrase 'vps linux' - https://www.google.com/search?q=vps+linux
07 Google Search with the phrase 'vps windows' - https://www.google.com/search?q=vps+windows

- **Corrade** - the bot's software should be downloaded from the developer's website. The bot can be hosted on your own computer or on a VPS, communication with the bot is via LSL, HTTP, MQTT, WebSockets, TCP. Works with third-party rental and sales software, such as CasperLet, CaspetVend. Web address: https://corrade.grimore.org/.
- **QubicBot** - software that works only on Windows. The full version costs $12, you can also host it on the manufacturer's server, then we will pay a few cents for a week of use. The program allows you to run an unlimited number of bots, send group and private messages, integrate with SmartBots and PikkuBot (to exp and your possibilities). Web address: http://qubicbot.com/.

As you can see, we have several suggestions, it's up to us which software we like.

Corrade - compatible examples will be used later in the book.

4. Corrade

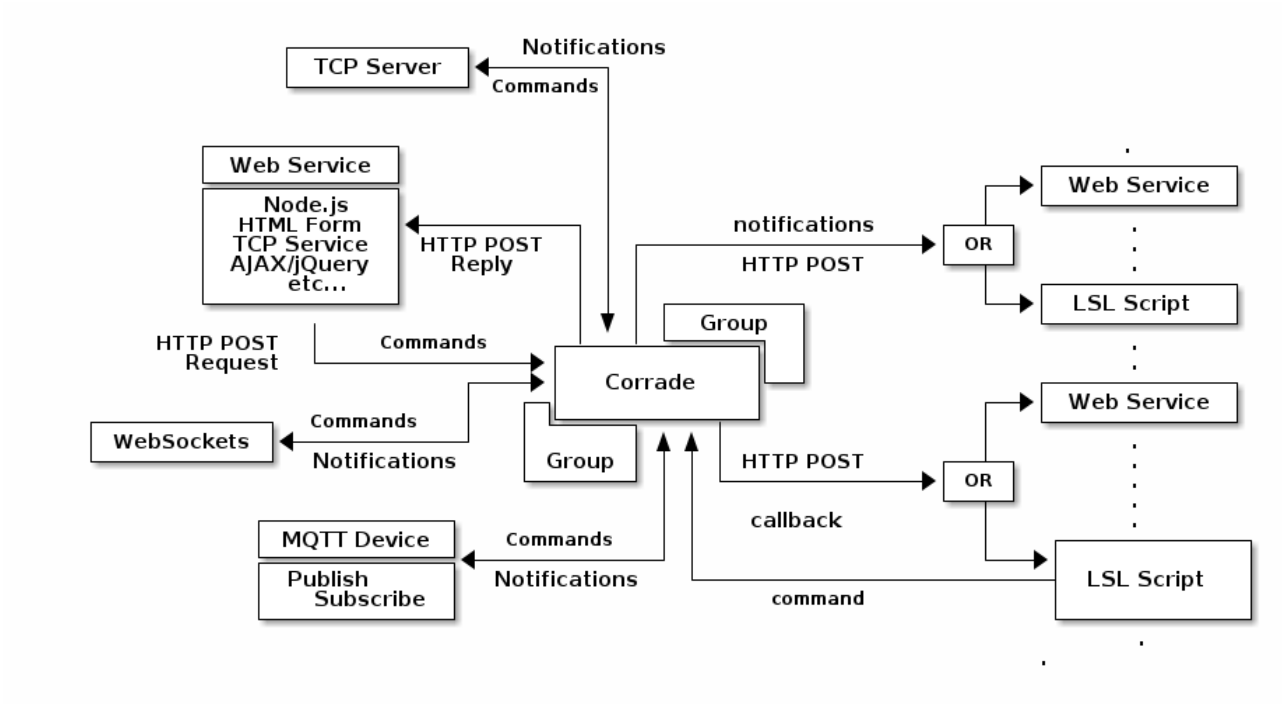
4.1. Introduction

Picture 4.1 - Corrade logo

Corrade:

- multi-functional, cross - platform software for running your own bot in the virtual world of Second Life,
- is the bridge between the user and Second Life,
- supports systems: Linux ARM, Linux 64-bit, Mac OSX 64-bit, Windows 7 and newer 64-bit,
- control by internal (LSL) and/or external scripts (any programming language),
- integration with external services through built - in servers:
  - HTTP,
  - WebSockets,
  - MQTT,
  - TCP.
- bot configuration takes place in a web browser via the following address: http://127.0.0.1:54377/,
  - advanced configuration by editing the Configuration.xml and Nucleus.xml files
- requirements:
  - one of the previously mentioned operating systems installed,

  - .NET Core 3.1+,
  - min. 2 GB of memory (4 GB recommended),
  - stable and fast internet connection.
- Website WWW: https://corrade.grimore.org/



Picture 4.2 - The way of information flow in Corrade

Corrade connects to Second Life as a normal avatar where, like any avatar, it belongs to some existing group in this world. In this group, he has assigned roles (one or more) with appropriate permissions.

By default, commands can be sent via an LSL script, which can also receive responses from the bot (callback), or this response can be redirected to an external service, such as a script on an external server.

The same goes for notifications that receive real-time responses from the bot and can be processed in an LSL script or by an external service.

Commands or notifications can also be sent by: TCP server, WebSockets server, MQTT server or script written in any programming language (eg PHP, Perl, Python, Ruby, etc.), where the response is sent in the same way.

Sending a command to Corrade should be bound with a group and a password as authorization data.

On the project's website you will find quite extensive API for the bot, and most of the commands have examples of use. The developer also provides examples of how to use LSL scripts on their store page in Second Life Marketplace[08] (all are free to use and modify) and HTTP templates (e.g. 2D and 3D map, group chat, etc.).

4.2. License

Corrade and related materials are licensed under the 'Wizardry and Steamworks Project-Closed and Open-Derivatives License 1.0' (WAS PC & OD 1.0)[09], which does not substantially reverse engineer the program and requires Corrade to be identified in the materials it appears in.

The license allows you to modify the source code, use for private and commercial purposes, copy, distribute, sublicense, re-license any material created by the same company or organization under the WAS PC & OD 1.0 license.

The manufacturer provides the software 'is as is' without the guarantee of correct software operation.

The technical support is at a good level, in case of errors, it is enough to write an appropriate application on the manufacturer's website.

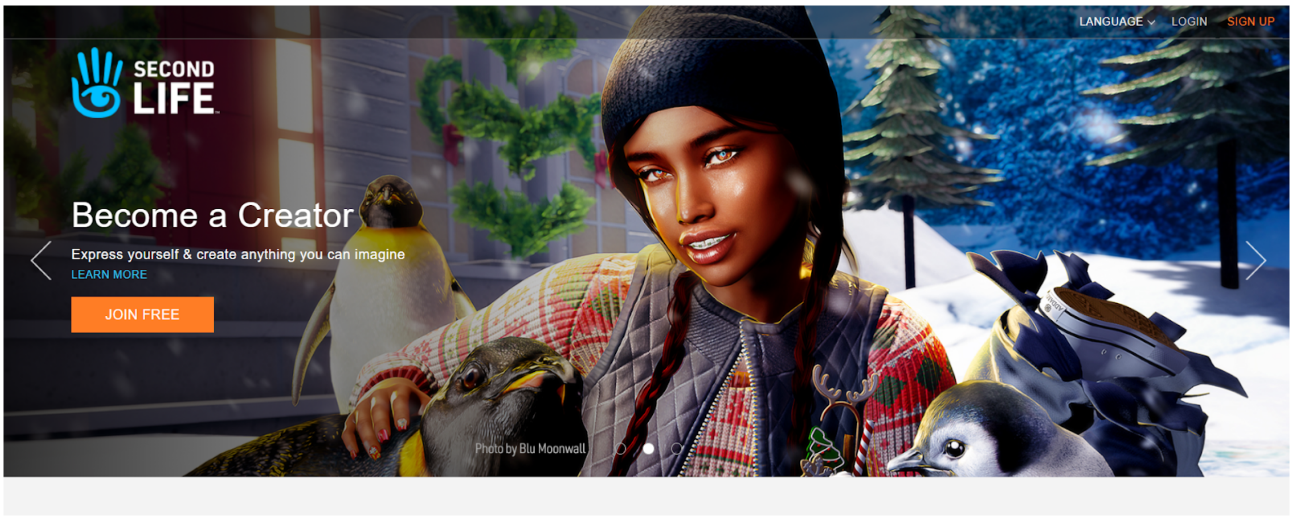
You may use the Corrade software for commercial purposes, however, you must visibly identify the manufacturer of Corrade.

Helper functions, e.g. wasURLEscape, wasCSVToList, etc. are licensed under WAS PC & OD 1.0.

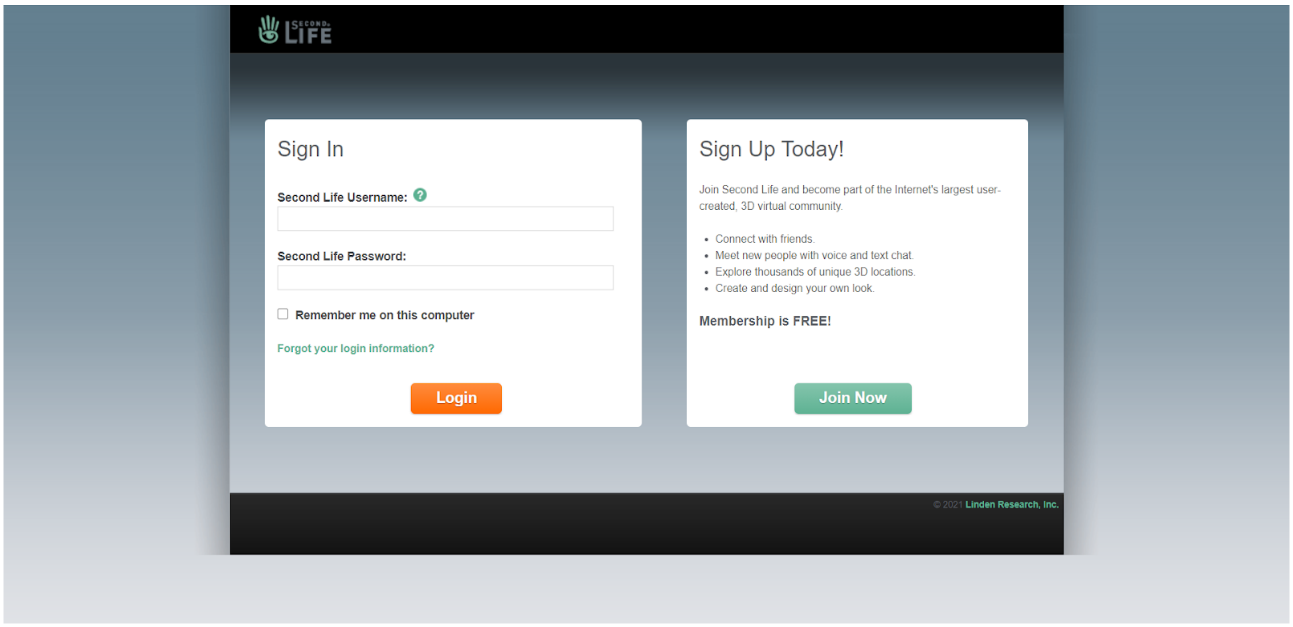
4.3. Creating an account on Second Life

Creating an account on Second Life is very simple. We go to the Second Life website[10] (Picture 4.3 - Second Life website), then either click the orange 'Join free' or 'Sign up' button on the top right of the screen (Picture 4.4 - Second Life - login or registration part).

08 Second Life Marketplace - https://marketplace.secondlife.com/stores/165275
09 License https://grimore.org/licenses/was - pc - od
10 Second Life website - https://secondlife.com/



Picture 4.3 - Second Life website



Sign In

Second Life Username:

Second Life Password:

Remember me on this computer

Forgot your login information?

Login

Sign Up Today!

Join Second Life and become part of the Internet's largest user-created, 3D virtual community.

- Connect with friends.
- Meet new people with voice and text chat.
- Explore thousands of unique 3D locations.
- Create and design your own look.

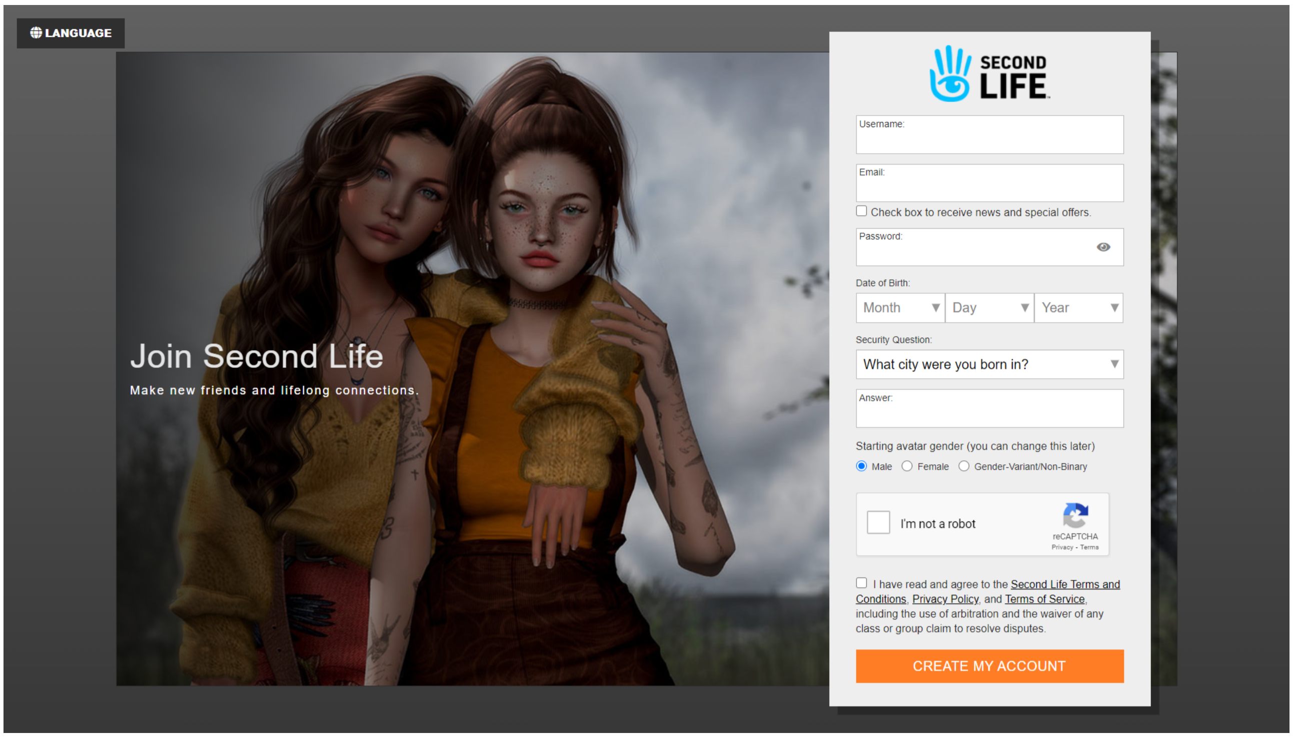
Membership is FREE!

Join Now

© 2021 Linden Research, Inc.

Picture 4.4 - Second Life - login or registration part

If you click 'Login' on the top right of the screen, you will be able to log into your account and manage this account.



Join Second Life
Make new friends and lifelong connections.

Username:

Email:

☐ Check box to receive news and special offers.

Password:

Date of Birth: Month Day Year

Security Question: What city were you born in?

Answer:

Starting avatar gender (you can change this later)
[✓ Male] Female Gender-Variant/Non-Binary

☐ I'm not a robot
reCAPTCHA
Privacy - Terms

☐ I have read and agree to the Second Life Terms and Conditions, Privacy Policy, and Terms of Service, including the use of arbitration and the waiver of any class or group claim to resolve disputes.

CREATE MY ACCOUNT

Picture 4.5 - Account registration page

When you come back, after clicking you will see a form to be filled in:

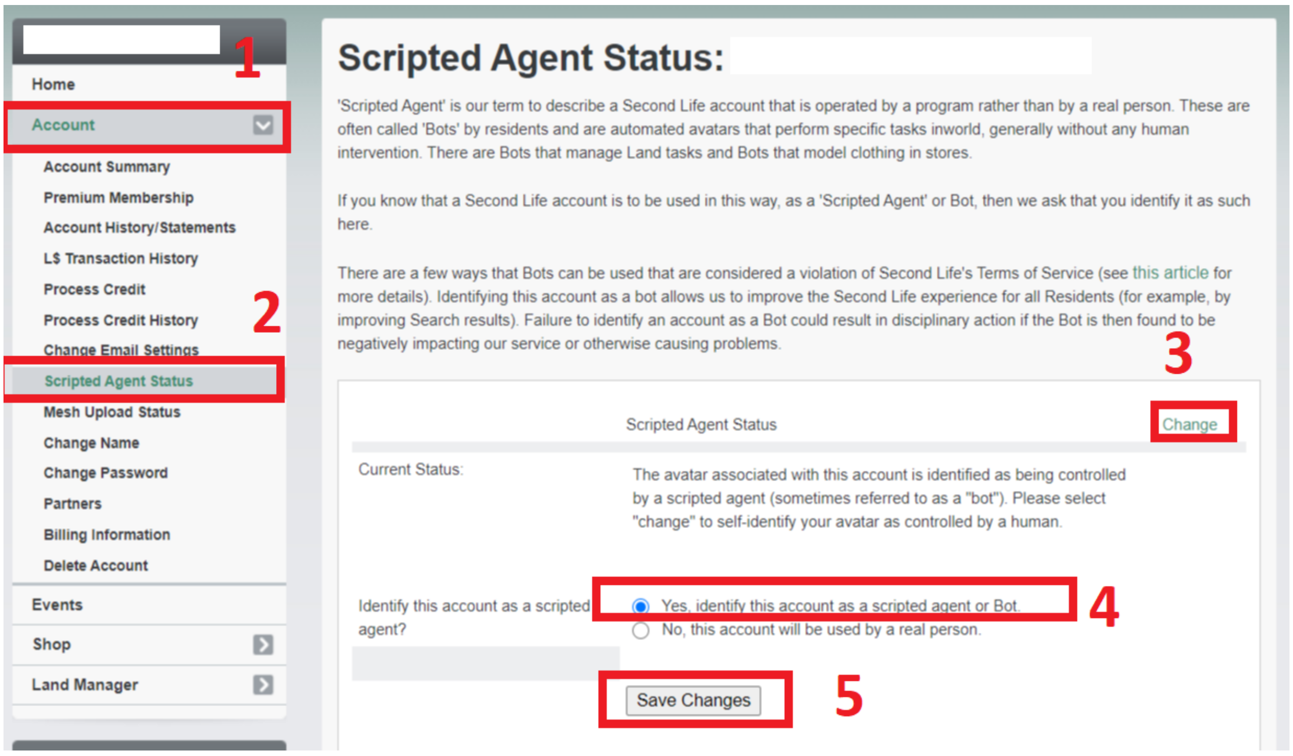
- Username - the name of the avatar in Second Life, it may contain letters and numbers, it is not possible to create an avatar with your own name and surname - they all receive the surname 'Resident',
- Email - enter your email address to which the avatar will be registered, on this email you will also receive Second Life notifications, e.g. saved offline messages,
- Check box to receive news and special offers - check if you want to receive news and special offers from Second Life,
- Password - enter the password for your avatar account,
- Date of Birth - your birthdate,
- Security Question - choose the type of question securing the account,
- Answer - enter your answer to the selected security question,

- Starting avatar gender - choose the gender of your avatar: male, female, or undefined, e.g. in a fictional character,
- I'm not robot - mark and, if necessary, solve the test that you are not setting up an account as an internet robot,
- I have read and agree (...) - check the option if you agree to the terms and regulations of Second Life.

After registration, you should probably confirm the willingness to create an account by clicking on the link that will be sent to the previously provided email.

After registration and approval, we can log in to the account on the Second Life website and to the world of Second Life.

It is recommended that you change the avatar status in your account (Picture 4.6 - Change your avatar status on the Second Life website), however, this has no benefit, rather it increases the statistics.



Scripted Agent Status:

'Scripted Agent' is our term to describe a Second Life account that is operated by a program rather than by a real person. These are often called 'Bots' by residents and are automated avatars that perform specific tasks inworld, generally without any human intervention. There are Bots that manage Land tasks and Bots that model clothing in stores.

If you know that a Second Life account is to be used in this way, as a 'Scripted Agent' or Bot, then we ask that you identify it as such here.

There are a few ways that Bots can be used that are considered a violation of Second Life's Terms of Service (see this article for more details). Identifying this account as a bot allows us to improve the Second Life experience for all Residents (for example, by improving Search results). Failure to identify an account as a Bot could result in disciplinary action if the Bot is then found to be negatively impacting our service or otherwise causing problems.

Scripted Agent Status

Current Status: The avatar associated with this account is identified as being controlled by a scripted agent (sometimes referred to as a "bot"). Please select "change" to self-identify your avatar as controlled by a human.

Identify this account as a scripted agent? [✓ Yes, identify this account as a scripted agent or Bot.] No, this account will be used by a real person.

Picture 4.6 - Change your avatar status on the Second Life website

4.4. Corrade installation and configuration

4.4.1. .NET Core installation

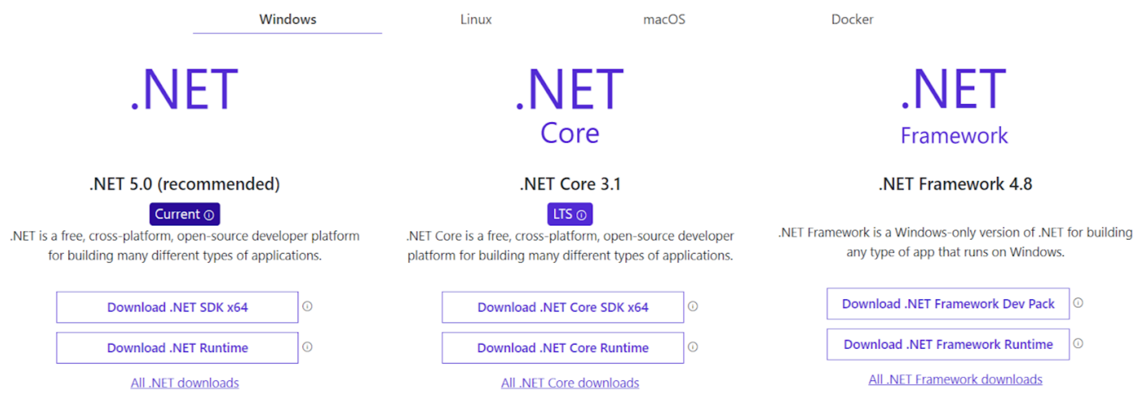
To run Corrade, you need .NET Core, which unifies the .NET framework for all known operating systems.

On the .NET download page[11] (Picture 4.7 - .NET download page) we can currently find: .NET (this is what we have to install), .NET Core (this is an older version with extended program support) and .NET Framework (an even older platform, used by older versions of Windows systems, e.g. Windows 7[12]). We are interested in the version on the left side of the screen - .NET. Next to this, click on the link 'All .NET downloads', then click on '.NET 5.0 (recommended)' as the version marked as recommended. At the top of the table, you have the latest version available for download. Once you have located the latest version, go to the right column of the table called 'Run apps - Runtime'. Download the '.NET Desktop Runtime' x64 version for Windows or '.NET Runtime' in the x64 version Installers. Here, too, we have the option of installing the program on other operating systems, such as Linux or MacOS, on other processor architectures.

Run the previously downloaded program for Windows or Mac OS, go through several stages of installation and close the program. Its operation is hassle - free.

If we will update .NET to a new version in the future, do not forget to uninstall earlier older versions via 'Add or Remove Programs'.

11 .NET download page - https://dotnet.microsoft.com/download
12 But you don't need to be installed, .NET can be installed



Windows Linux macOS Docker

.NET

.NET 5.0 (recommended)

Current

.NET is a free, cross-platform, open-source developer platform for building many different types of applications.

Download .NET SDK x64

Download .NET Runtime

All .NET downloads

.NET Core

.NET Core 3.1

LTS

.NET Core is a free, cross-platform, open-source developer platform for building many different types of applications.

Download .NET Core SDK x64

Download .NET Core Runtime

All .NET Core downloads

.NET Framework

.NET Framework 4.8

.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.

Download .NET Framework Dev Pack

Download .NET Framework Runtime

All .NET Framework downloads

Picture 4.7 - .NET download page

When it comes to a Linux - based system, it is worth checking first if we can install .NET via the snap program, so it is worth installing it first, and then in the terminal enter:

```
sudo snap install dotnet-sdk --classic
```

Which will install the SDK and Runtime for .NET

After that, we can postpone .NET normally through the snap:

```
sudo snap refresh dotnet-sdk
```

If our system does not allow you to install the snap program, follow the standard steps described in the instructions on the above-mentioned page.

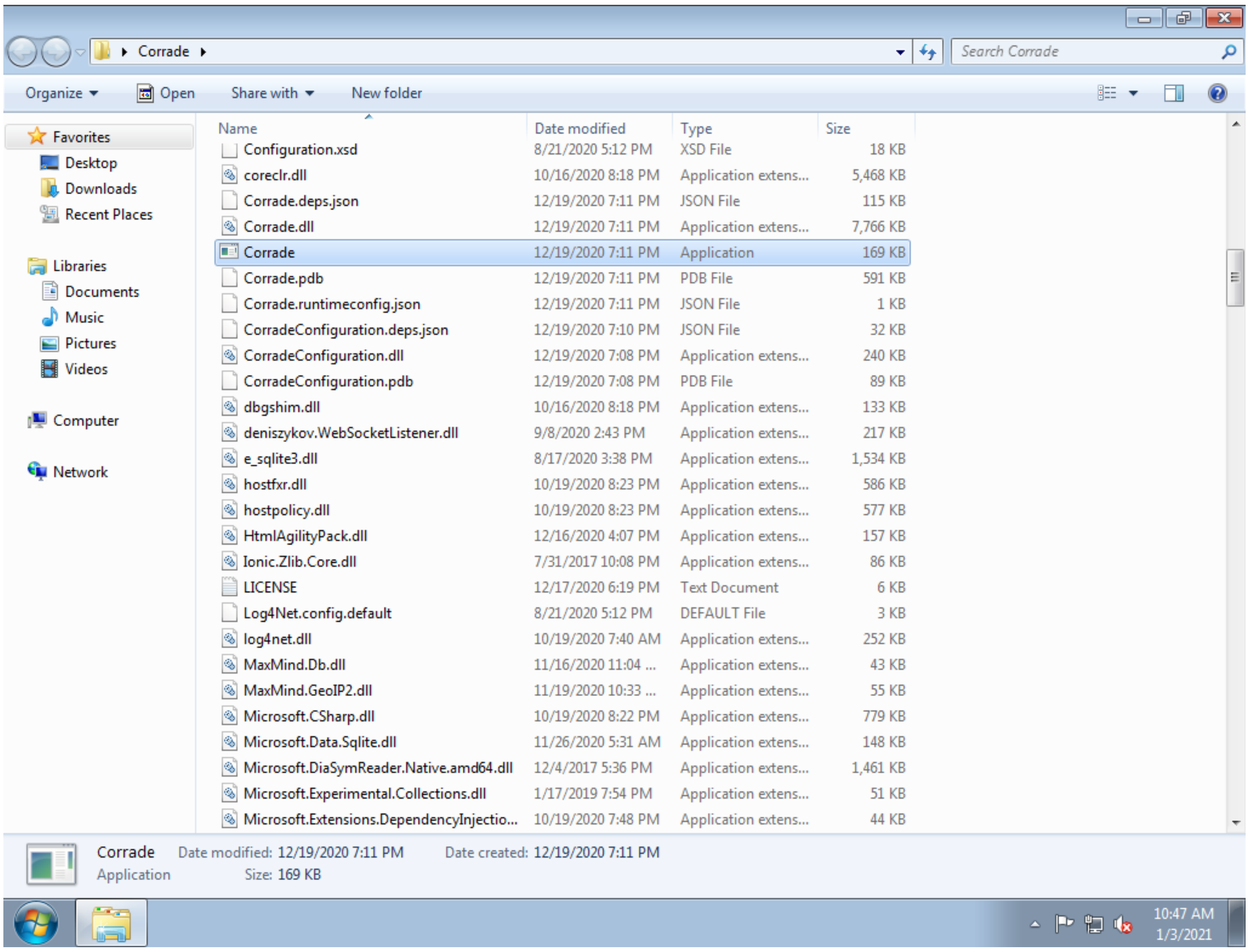
4.4.2. A system from the Windows family

In the additional materials you will find a folder with the script in PowerShell and instructions for quick installation of the bot.

1. Go to the website https://corrade.grimore.org/.
2. Click on the button DOWNLOAD.
3. Click on 'corrade'.
4. Click on 'win-x64'.

You will get faster by clicking on the link: https://corrade.grimore.org/download/corrade/win-x64/.

5. Click twice on 'Last modified' and the file list will be displayed by time from youngest to oldest.
6. Click on the first ZIP you see on the list: Corrade-[VERSION]-win-x64.zip.
7. Wait for it to download.
8. Unpack the ZIP archive into a folder.
9. Open the directory where the files unpacked.
10. Find and run Corrade.exe.



Name	Date modified	Type	Size
Configuration.xsd	8/21/2020 5:12 PM	XSD File	18 KB
coreclr.dll	10/16/2020 8:18 PM	Application extens...	5,468 KB
Corrade.deps.json	12/19/2020 7:11 PM	JSON File	115 KB
Corrade.dll	12/19/2020 7:11 PM	Application extens...	7,766 KB
Corrade	12/19/2020 7:11 PM	Application	169 KB
Corrade.pdb	12/19/2020 7:11 PM	PDB File	591 KB
Corrade.runtimeconfig.json	12/19/2020 7:11 PM	JSON File	1 KB
CorradeConfiguration.deps.json	12/19/2020 7:10 PM	JSON File	32 KB
CorradeConfiguration.dll	12/19/2020 7:08 PM	Application extens...	240 KB
CorradeConfiguration.pdb	12/19/2020 7:08 PM	PDB File	89 KB
dbgshim.dll	10/16/2020 8:18 PM	Application extens...	133 KB
deniszykov.WebSocketListener.dll	9/8/2020 2:43 PM	Application extens...	217 KB
e_sqlite3.dll	8/17/2020 3:38 PM	Application extens...	1,534 KB
hostfxr.dll	10/19/2020 8:23 PM	Application extens...	586 KB
hostpolicy.dll	10/19/2020 8:23 PM	Application extens...	577 KB
HtmlAgilityPack.dll	12/16/2020 4:07 PM	Application extens...	157 KB
Ionic.Zlib.Core.dll	7/31/2017 10:08 PM	Application extens...	86 KB
LICENSE	12/17/2020 6:19 PM	Text Document	6 KB
Log4Net.config.default	8/21/2020 5:12 PM	DEFAULT File	3 KB
log4net.dll	10/19/2020 7:40 AM	Application extens...	252 KB
MaxMind.Db.dll	11/16/2020 11:04 ...	Application extens...	43 KB
MaxMind.GeoIP2.dll	11/19/2020 10:33 ...	Application extens...	55 KB
Microsoft.CSharp.dll	10/19/2020 8:22 PM	Application extens...	779 KB
Microsoft.Data.Sqlite.dll	11/26/2020 5:31 AM	Application extens...	148 KB
Microsoft.DiaSymReader.Native.amd64.dll	12/4/2017 5:36 PM	Application extens...	1,461 KB
Microsoft.Experimental.Collections.dll	1/17/2019 7:54 PM	Application extens...	51 KB
Microsoft.Extensions.DependencyInjectio...	10/19/2020 7:48 PM	Application extens...	44 KB

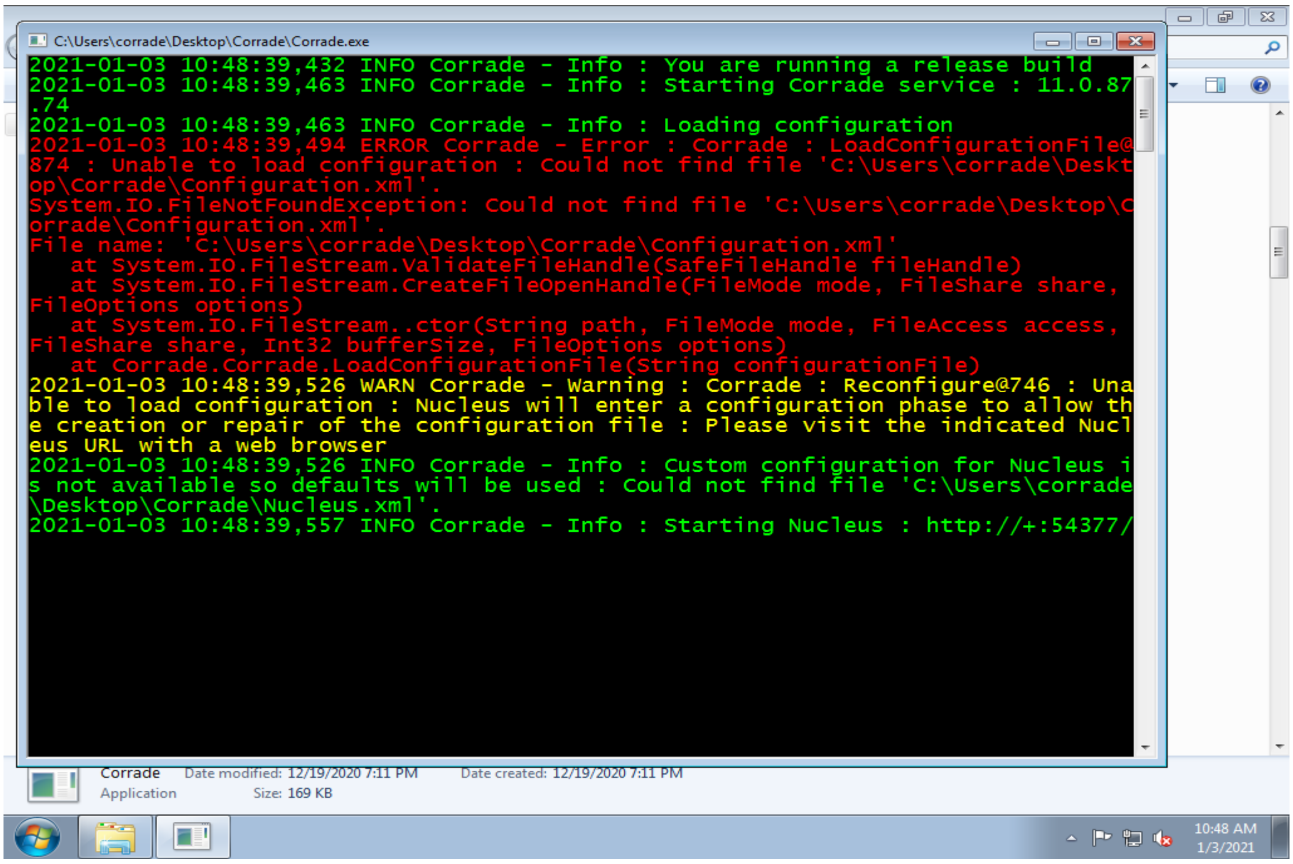
Corrade Date modified: 12/19/2020 7:11 PM Date created: 12/19/2020 7:11 PM
Application Size: 169 KB

Picture 4.8 - Folder with Corrade files and the selected program

The console version of the program will be started.

You will see real-time notifications regarding the Corrade software operation.

By default, this is information that since the configuration file was not found, the program started with default settings and Nucleus is available on port 54377, where you can configure the bot.



C:\Users\corrade\Desktop\Corrade\Corrade.exe

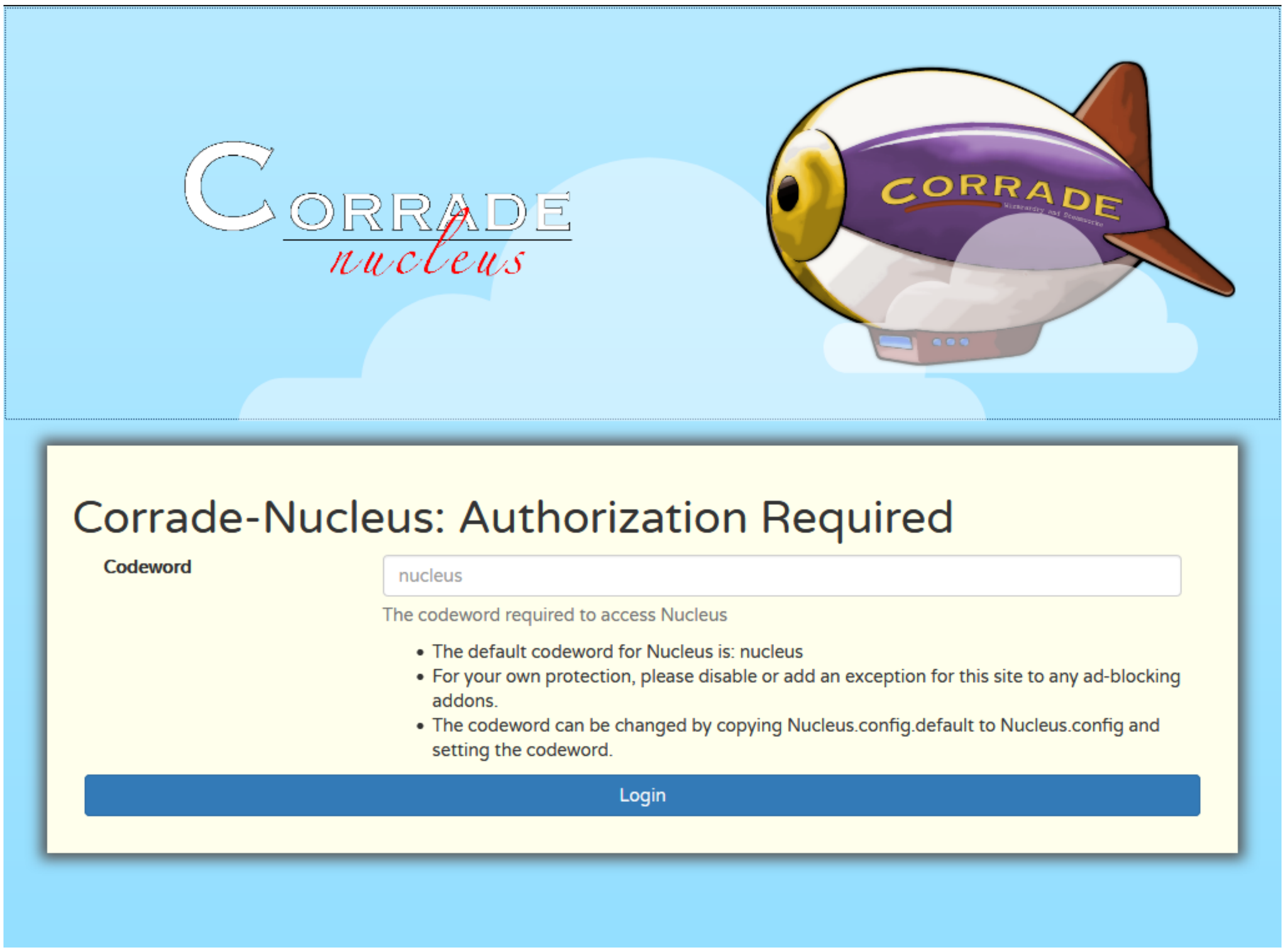
```
2021-01-03 10:48:39,432 INFO Corrade - Info : You are running a release build
2021-01-03 10:48:39,463 INFO Corrade - Info : Starting Corrade service : 11.0.87
.74
2021-01-03 10:48:39,463 INFO Corrade - Info : Loading configuration
2021-01-03 10:48:39,494 ERROR Corrade - Error : Corrade : LoadConfigurationFile@
874 : Unable to load configuration : Could not find file 'C:\Users\corrade\Deskt
op\Corrade\Configuration.xml'.
System.IO.FileNotFoundException: Could not find file 'C:\Users\corrade\Desktop\C
orrade\Configuration.xml'.
File name: 'C:\Users\corrade\Desktop\Corrade\Configuration.xml'
   at System.IO.FileStream.ValidateFileHandle(SafeFileHandle fileHandle)
   at System.IO.FileStream.CreateFileOpenHandle(FileMode mode, FileShare share,
FileOptions options)
   at System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access,
FileShare share, Int32 bufferSize, FileOptions options)
   at Corrade.Corrade.LoadConfigurationFile(String configurationFile)
2021-01-03 10:48:39,526 WARN Corrade - Warning : Corrade : Reconfigure@746 : Una
ble to load configuration : Nucleus will enter a configuration phase to allow th
e creation or repair of the configuration file : Please visit the indicated Nucl
eus URL with a web browser
2021-01-03 10:48:39,526 INFO Corrade - Info : Custom configuration for Nucleus i
s not available so defaults will be used : Could not find file 'C:\Users\corrade
\Desktop\Corrade\Nucleus.xml'.
2021-01-03 10:48:39,557 INFO Corrade - Info : Starting Nucleus : http://+:54377/
```

Corrade
Application
Date modified: 12/19/2020 7:11 PM
Size: 169 KB
Date created: 12/19/2020 7:11 PM
10:48 AM
1/3/2021

Picture 4.9 - Corrade in the console version

11. We run the web browser without ad-blocking add-ons.
12. We enter the address: http://127.0.0.1:54377.

    A screen should appear before our eyes:



Corrade-Nucleus: Authorization Required

Codeword: nucleus

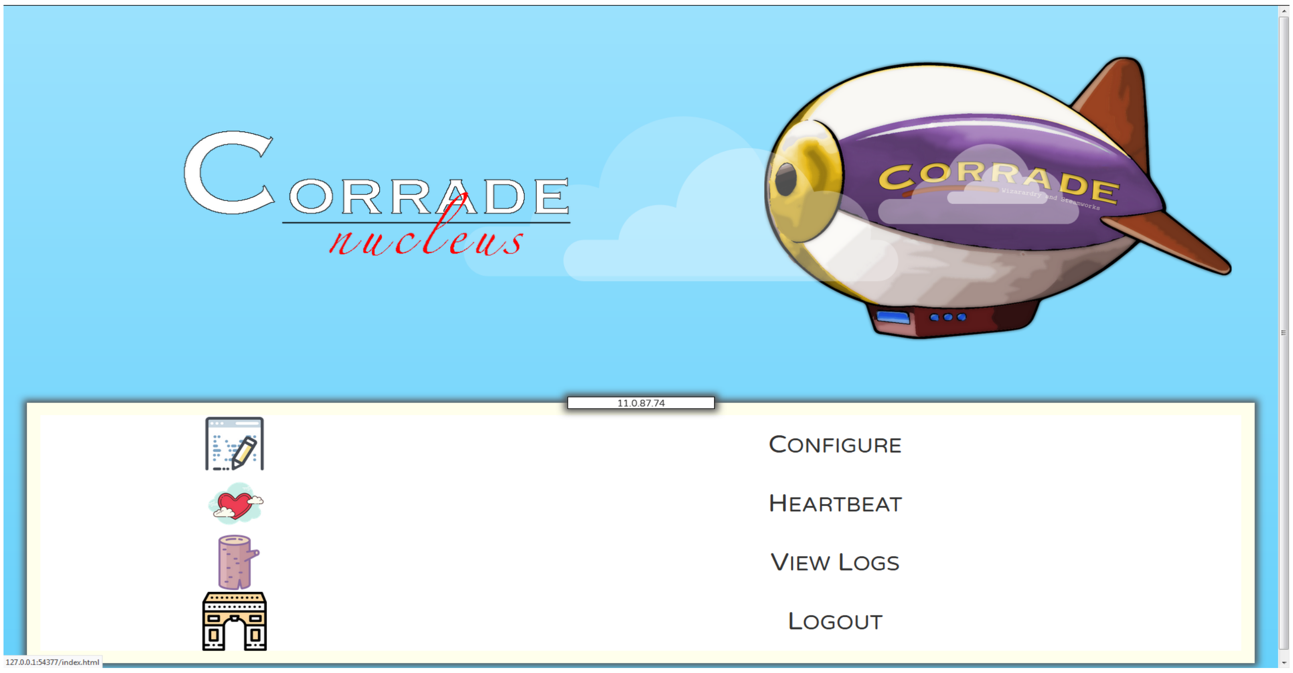
The codeword required to access Nucleus

- The default codeword for Nucleus is: nucleus
- For your own protection, please disable or add an exception for this site to any ad-blocking addons.
- The codeword can be changed by copying Nucleus.config.default to Nucleus.config and setting the codeword.

Picture 4.10 - Authorization of access to Nucleus

13. In the 'Codeword' field, enter the default password: 'nucleus' (without the quotes) and click the 'Login' button.

    Another screen will appear:

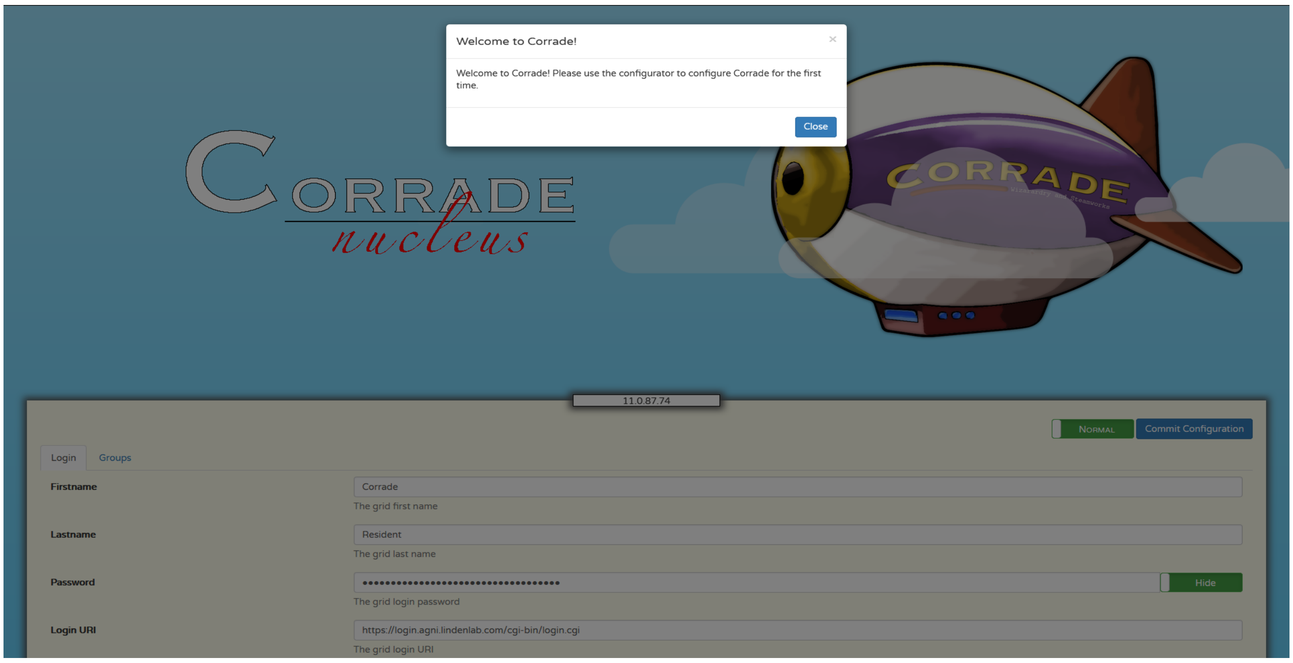


Picture 4.11 - Nucleus - options to choose from

Here we can go to:

- 'Configure' - bot configuration,
- 'Heartbeat' - RAM and CPU consumption of the bot,
- 'View Logs' - simple viewing of bot logs,
- 'Logout' - logging out of Nucleus.

14. We will proceed to configuring our bot, so click on 'Configure'.
15. Enter the password to access Nucleus.



Welcome to Corrade!

Welcome to Corrade! Please use the configurator to configure Corrade for the first time.

Firstname	Corrade The grid first name
Lastname	Resident The grid last name
Password	•••••••••••••••••••••••••••••••••••• The grid login password
Login URI	https://login.agni.lindenlab.com/cgi-bin/login.cgi The grid login URI

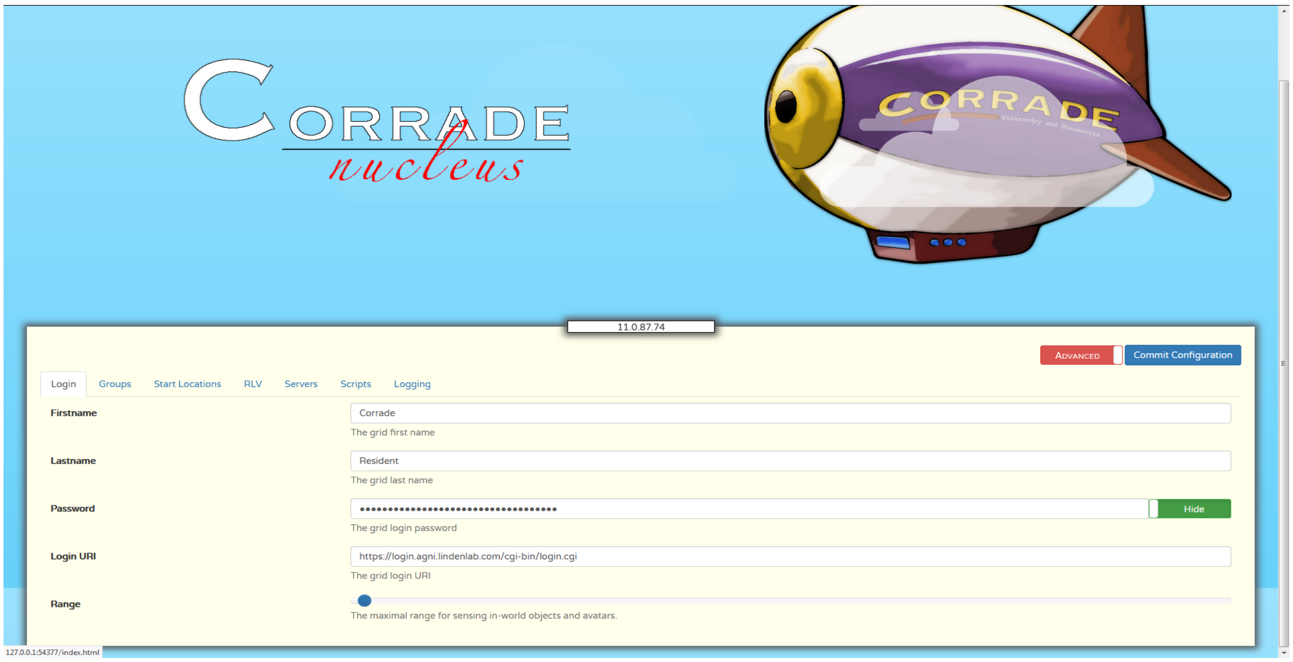
Picture 4.12 - Nucleus - information about the first bot configuration

Since the program does not detect the Configuration.xml configuration file, it will show us a nice message about the possibility of configuring the bot.

16. Click on Close.

Next to the 'Confirm Configuration' button, you can switch between the simple (Normal) and the advanced (Advanced) bot configuration.

I will cover the advanced version.



Firstname: Corrade
The grid first name

Lastname: Resident
The grid last name

Password: ••••••••••••••••••••••••••••••••••
The grid login password

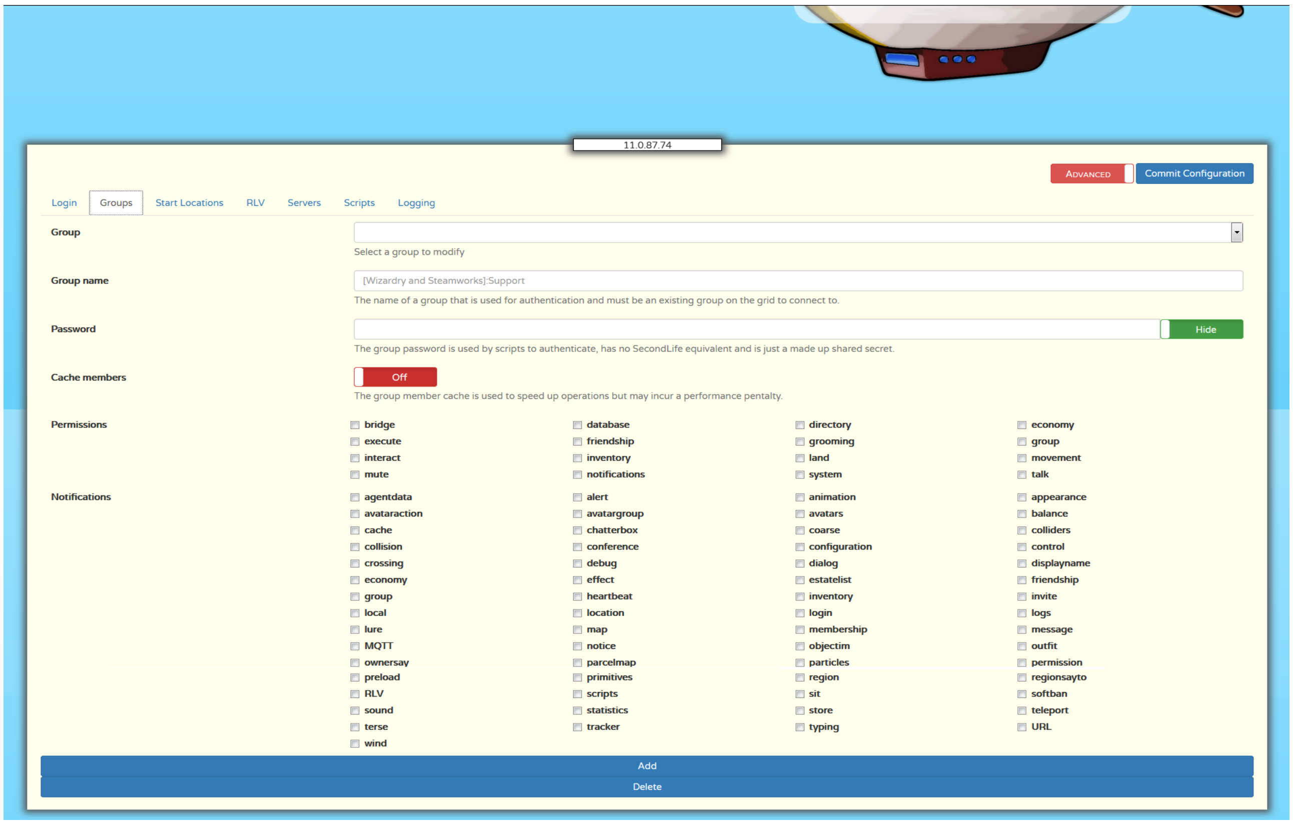
Login URI: https://login.agni.lindenlab.com/cgi-bin/login.cgi
The grid login URI

Range
The maximal range for sensing in-world objects and avatars.

Picture 4.13 - 'Login' tab

On the 'Login' tab (Picture 4.13 - 'Login' tab):

17. we give the first and last name of our bot (the default name is: Resident).
18. bot account password.
19. The grid login URL (we leave the default for Second Life as https://login.agni.lindenlab.com/cgi-bin/login.cgi).
20. maximum range of detection of objects and avatars in the world (you can leave the default value).



Group

Select a group to modify

Group name: [Wizardry and Steamworks]:Support

The name of a group that is used for authentication and must be an existing group on the grid to connect to.

Password

The group password is used by scripts to authenticate, has no SecondLife equivalent and is just a made up shared secret.

Cache members: Off

The group member cache is used to speed up operations but may incur a performance pentalty.

Permissions	☐ bridge	☐ database	☐ directory	☐ economy
	☐ execute	☐ friendship	☐ grooming	☐ group
	☐ interact	☐ inventory	☐ land	☐ movement
	☐ mute	☐ notifications	☐ system	☐ talk
Notifications	☐ agentdata	☐ alert	☐ animation	☐ appearance
	☐ avataraction	☐ avatargroup	☐ avatars	☐ balance
	☐ cache	☐ chatterbox	☐ coarse	☐ colliders
	☐ collision	☐ conference	☐ configuration	☐ control
	☐ crossing	☐ debug	☐ dialog	☐ displayname
	☐ economy	☐ effect	☐ estatelist	☐ friendship
	☐ group	☐ heartbeat	☐ inventory	☐ invite
	☐ local	☐ location	☐ login	☐ logs
	☐ lure	☐ map	☐ membership	☐ message
	☐ MQTT	☐ notice	☐ objectim	☐ outfit
	☐ ownersay	☐ parcelmap	☐ particles	☐ permission
	☐ preload	☐ primitives	☐ region	☐ regionsayto
	☐ RLV	☐ scripts	☐ sit	☐ softban
	☐ sound	☐ statistics	☐ store	☐ teleport
	☐ terse	☐ tracker	☐ typing	☐ URL
	☐ wind			

Picture 4.14 - 'Groups' tab

On the 'Groups' tab we manage groups.

21. The group must exist in the world of Second Life and the bot must be a member of this group (before configuration, it is best to log in as a bot, join the group, deselect group notifications).
22. To add a group, first click on the 'Add' button at the bottom.
23. The group 'new group' will be added to the list.
24. From the 'Group' drop - down list, select the newly added group 'new group'.
25. The settings and permissions for this group will be loaded. We can change them freely.
26. First of all, we change the name of the group to the correct one.
27. Then we enter the password we invented (it is best if it should be 10 or more characters long, consisting of lowercase letters, uppercase letters, numbers, with special characters it is best to skip it - Corrade may have a problem with encoding them).

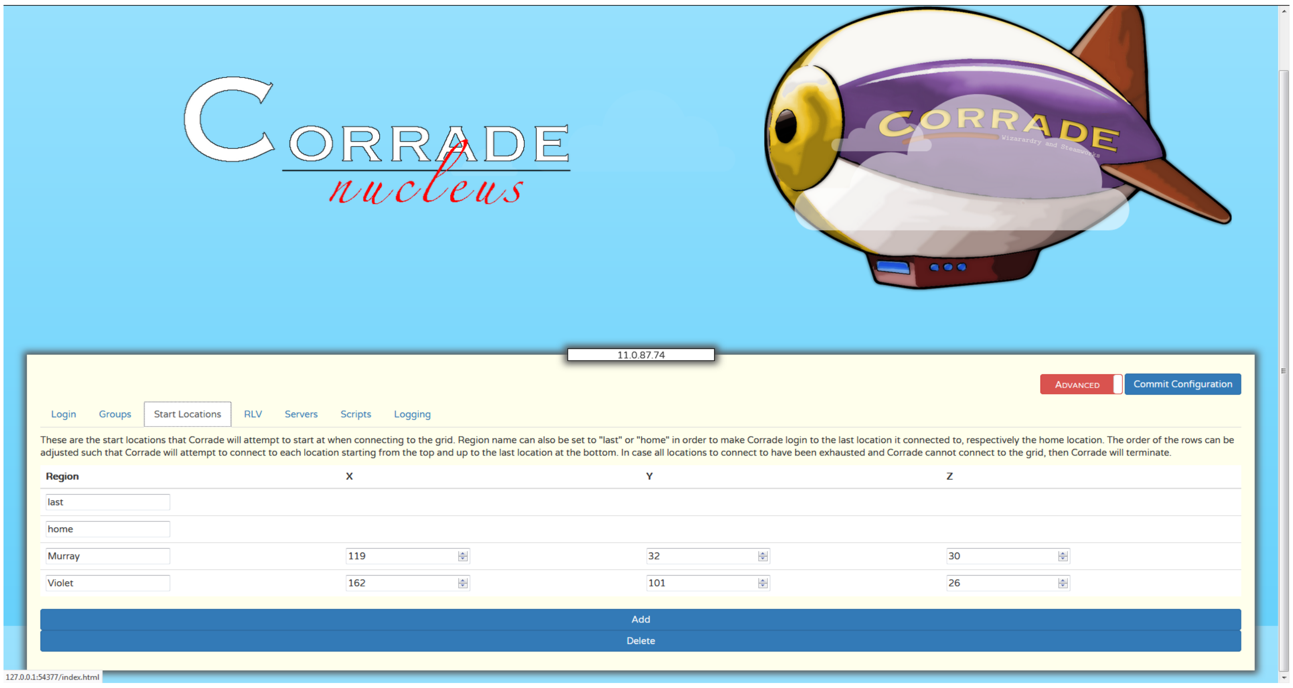
There is no such thing as a group password in Second Life.
The password set for the group in Corrade is an additional protection against

unauthorized access to the bot.

28. Then, in 'Cache members', we decide whether this option is to be turned on or off - it causes that the avatar data from the group will be saved on the disk so that later operations can be performed faster. Sometimes this option can fail and it is turned off for me.
29. Next we have a list of permissions.

    We mark the necessary ones, we mark unnecessary permissions.

    The permissions used by the commands we use should definitely be selected.
30. If everything is fine, go to the next tab, but here on this tab, I recommend deleting the default group called '[Wizardry and Steamworks]: Support'. Simply select this group from the 'Group' list and put it on the 'Delete' button.



These are the start locations that Corrade will attempt to start at when connecting to the grid. Region name can also be set to "last" or "home" in order to make Corrade login to the last location it connected to, respectively the home location. The order of the rows can be adjusted such that Corrade will attempt to connect to each location starting from the top and up to the last location at the bottom. In case all locations to connect to have been exhausted and Corrade cannot connect to the grid, then Corrade will terminate.

Region	X	Y	Z
last			
home			
Murray	119	32	30
Violet	162	101	26

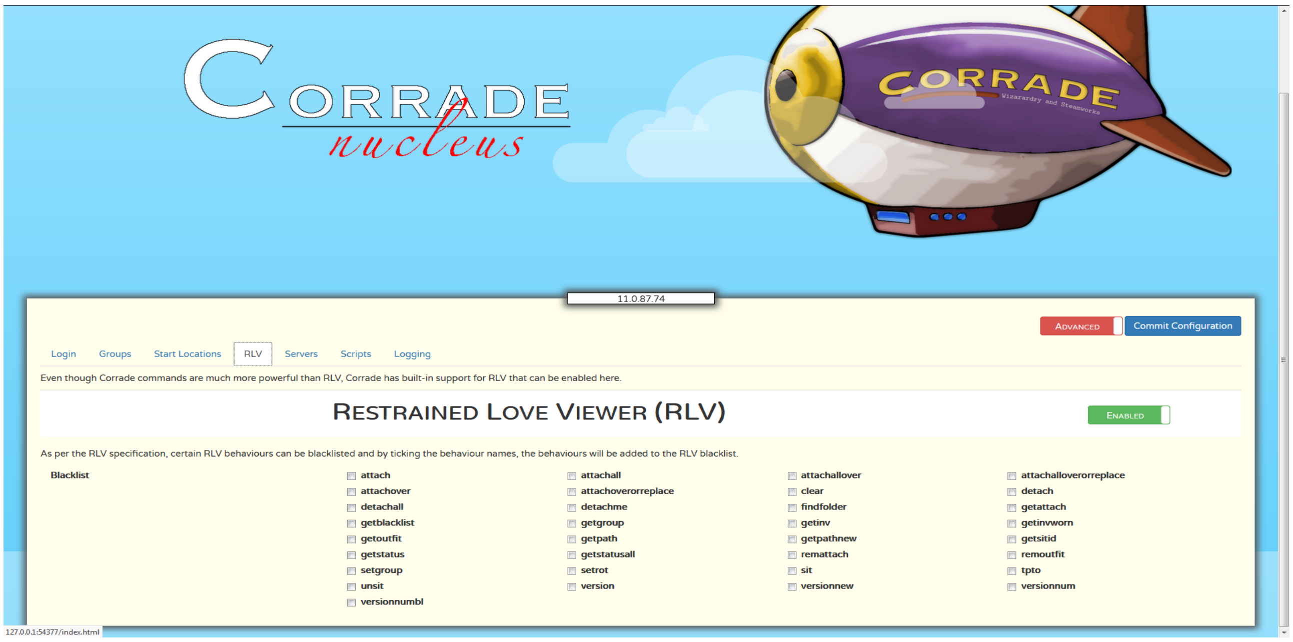
Picture 4.15 - 'Start Locations' tab

31. On the 'Start Locations' tab, indicate the starting place or places when the bot logs in to Second Life. If there are more of these starting locations, if one place is not available when the bot logs into Second Life, the bot will try to appear in the next place. If, however, all places are unavailable, the bot will not log into Second Life, and a message will appear in the logs and console that the places are unreachable.
32. To add a new starting point, we click on the 'Add' button.

Then we can enter into the last line:

- last - the bot logs in to the last saved place when logging out of Second Life,
- home - the bot logs in to the starting place,
- a specific position in Second Life by entering the name of the region, position X, Y, Z.

33. To delete a given starting point, click on the region field and then click on the 'Delete' button.



Even though Corrade commands are much more powerful than RLV, Corrade has built-in support for RLV that can be enabled here.

RESTRAINED LOVE VIEWER (RLV)

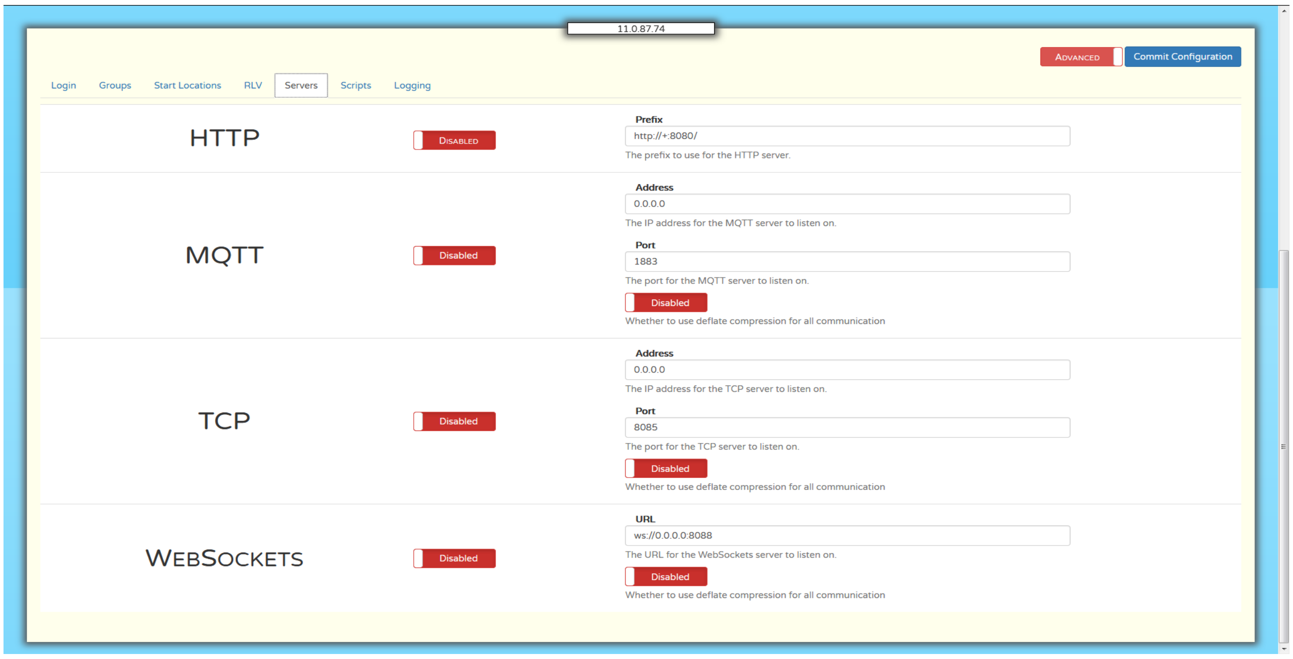
As per the RLV specification, certain RLV behaviours can be blacklisted and by ticking the behaviour names, the behaviours will be added to the RLV blacklist.

Blacklist	☐ attach	☐ attachall	☐ attachallover	☐ attachalloverorreplace
	☐ attachover	☐ attachoverorreplace	☐ clear	☐ detach
	☐ detachall	☐ detachme	☐ findfolder	☐ getattach
	☐ getblacklist	☐ getgroup	☐ getinv	☐ getinvworn
	☐ getoutfit	☐ getpath	☐ getpathnew	☐ getsitid
	☐ getstatus	☐ getstatusall	☐ remattach	☐ remoutfit
	☐ setgroup	☐ setrot	☐ sit	☐ tpto
	☐ unsit	☐ version	☐ versionnew	☐ versionnum
	☐ versionnumbl			

Picture 4.16 - 'RLV' tab

34. On the 'RLV' tab, you can click on the options you want to be banned.

    By default, I skip this tab.



HTTP — DISABLED
Prefix: http://+:8080/
The prefix to use for the HTTP server.

MQTT — Disabled
Address: 0.0.0.0
The IP address for the MQTT server to listen on.
Port: 1883
The port for the MQTT server to listen on.
Disabled
Whether to use deflate compression for all communication

TCP — Disabled
Address: 0.0.0.0
The IP address for the TCP server to listen on.
Port: 8085
The port for the TCP server to listen on.
Disabled
Whether to use deflate compression for all communication

WEBSOCKETS — Disabled
URL: ws://0.0.0.0:8088
The URL for the WebSockets server to listen on.
Disabled
Whether to use deflate compression for all communication

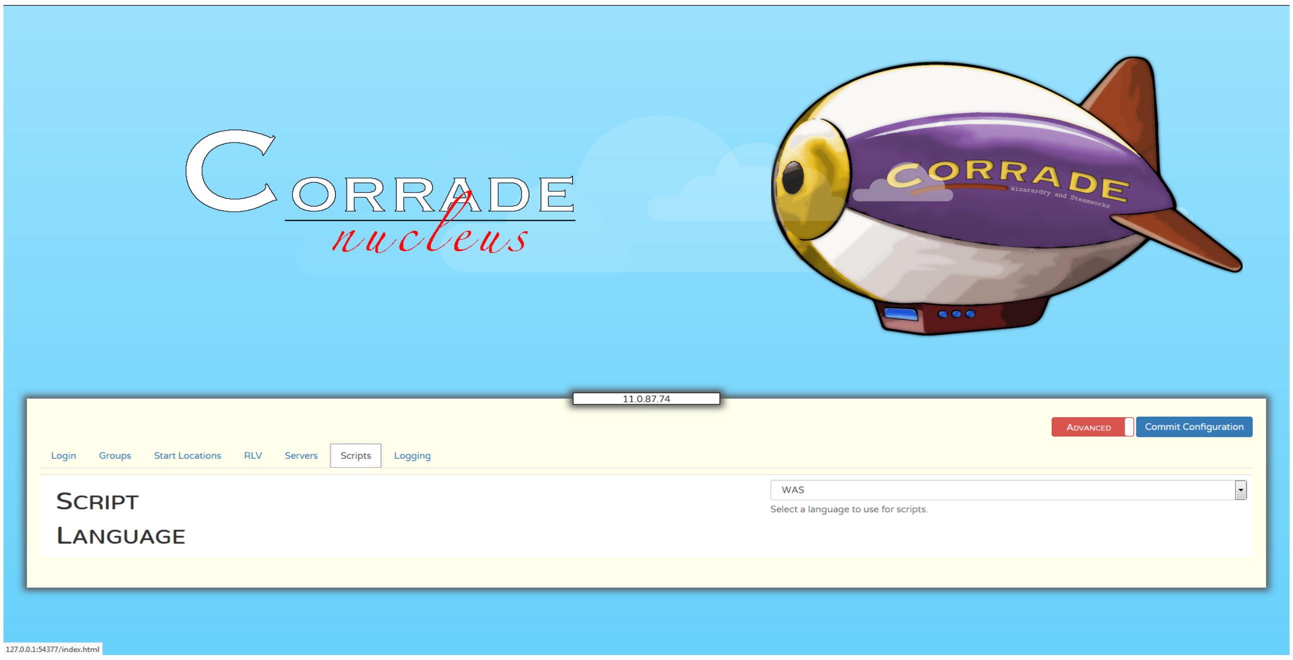
Picture 4.17 - 'Servers' tab

35. On the 'Servers' tab, define which servers are to receive requests and send responses to the bot. If they are all disabled, you can only send commands to the bot via LSL in Second Life. To enable a given server, next to it, click 'Disabled' and 'Enabled'.

At the server, we can specify the IP address and port (as well as whether the data should be compressed during communication) on which to listen.

The IP address can be:

- 127.0.0.1 - the server will listen locally, so other programs and scripts must be placed on the same computer as the bot,
- + (HTTP) lub 0.0.0.0 - the server will listen on each interface of the network card,
- specific IP address - the server will listen on the specified IP address.



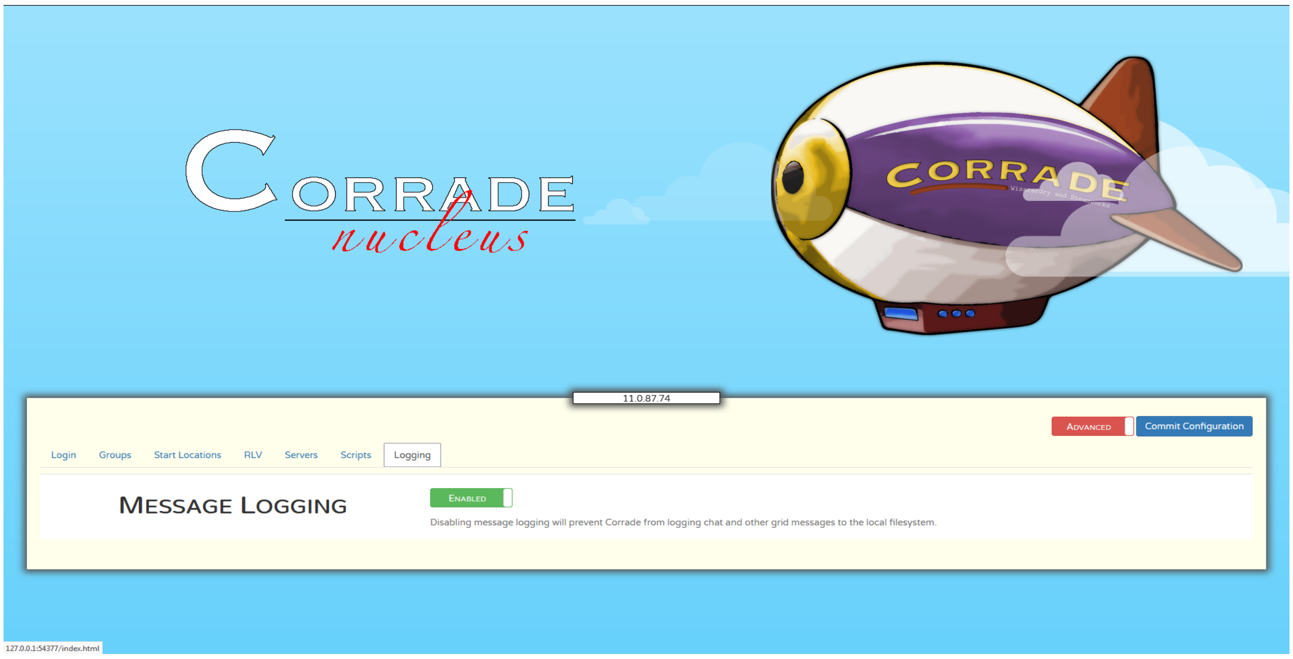
Script Language

WAS

Select a language to use for scripts.

Picture 4.18 - 'Scripts' tab

36. On the 'Scripts' tab, define whether the bot should process scripts according to WAS or JSON.



Message Logging

Enabled

Disabling message logging will prevent Corrade from logging chat and other grid messages to the local filesystem.

Picture 4.19 - 'Logging' tab

37. On the 'Logging' tab, define whether Corrade should log all messages from: chat, IM, etc.

38. If everything is ready, just click on the 'Commit Configuration' button, enter the authentication password and click on 'Submit'.
    If everything was successful, we will see a window that the changes have been applied, the bot should log into Second Life, and in the console we will see further messages without any errors.

4.4.3. Linux x64 family systems (Debian family)

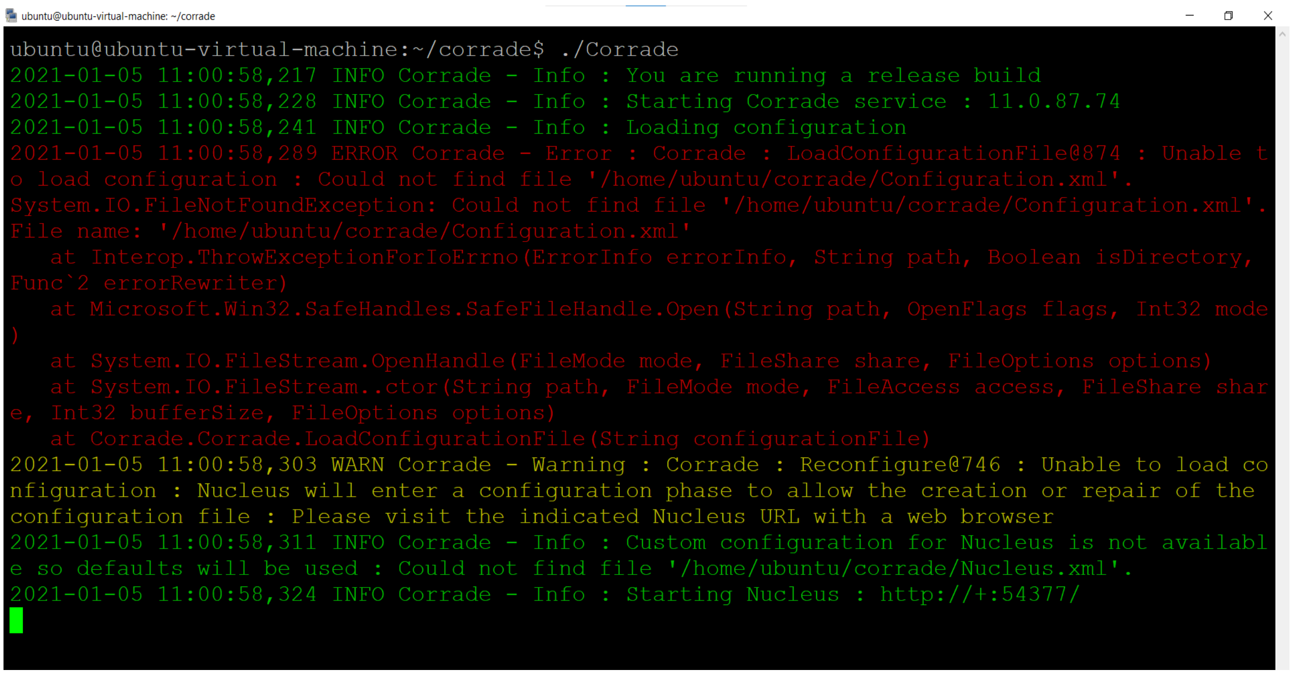
1. Launch the terminal,
2. Download the zip archive from the site corrade.grimore.org/download/corrade/linux-x64,
3. Unpack them into a folder,
4. Go to the Corrade folder,
5. Run Corrade by typing: ./Corrade
6. Further configuration is done via Nucleus, see the chapter: **A system from the Windows family**.

The commands that must be entered into the terminal are shown below.

```
cd /tmp/

zip=$( wget -O- "https://corrade.grimore.org/download/corrade/linux-x64/?C=M;O=D" |
grep -o "Corrade-.*\.zip\">" | sed 's/Corrade-//' | sed 's/">//' | sort -hr | head -n 1
)

wget --no-check-certificate -O corrade.zip
"https://corrade.grimore.org/download/corrade/linux-x64/Corrade-$zip"
unzip -o corrade.zip -d "$HOME/corrade"

cd $HOME/corrade

./Corrade
```

In the event of a problem, give permission to the folder where Corrade is located.
chmod -R 0757 corrade

After starting, messages from the program will appear:



ubuntu@ubuntu-virtual-machine: ~/corrade

```
ubuntu@ubuntu-virtual-machine:~/corrade$ ./Corrade
2021-01-05 11:00:58,217 INFO Corrade - Info : You are running a release build
2021-01-05 11:00:58,228 INFO Corrade - Info : Starting Corrade service : 11.0.87.74
2021-01-05 11:00:58,241 INFO Corrade - Info : Loading configuration
2021-01-05 11:00:58,289 ERROR Corrade - Error : Corrade : LoadConfigurationFile@874 : Unable t
o load configuration : Could not find file '/home/ubuntu/corrade/Configuration.xml'.
System.IO.FileNotFoundException: Could not find file '/home/ubuntu/corrade/Configuration.xml'.
File name: '/home/ubuntu/corrade/Configuration.xml'
   at Interop.ThrowExceptionForIoErrno(ErrorInfo errorInfo, String path, Boolean isDirectory,
Func`2 errorRewriter)
   at Microsoft.Win32.SafeHandles.SafeFileHandle.Open(String path, OpenFlags flags, Int32 mode
)
   at System.IO.FileStream.OpenHandle(FileMode mode, FileShare share, FileOptions options)
   at System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare shar
e, Int32 bufferSize, FileOptions options)
   at Corrade.Corrade.LoadConfigurationFile(String configurationFile)
2021-01-05 11:00:58,303 WARN Corrade - Warning : Corrade : Reconfigure@746 : Unable to load co
nfiguration : Nucleus will enter a configuration phase to allow the creation or repair of the
configuration file : Please visit the indicated Nucleus URL with a web browser
2021-01-05 11:00:58,311 INFO Corrade - Info : Custom configuration for Nucleus is not availabl
e so defaults will be used : Could not find file '/home/ubuntu/corrade/Nucleus.xml'.
2021-01-05 11:00:58,324 INFO Corrade - Info : Starting Nucleus : http://+:54377/
```

Picture 4.20 - Corrade running on Linux

4.4.4. Mac OS

First, we need to disable the program's signature checking as Corrade is unsigned and will be recognized as created by an unknown developer.

For this purpose:

1. Click on the 'Launchpad' icon at the bottom of the window (by default it is the second icon from the left).
2. Enter the word 'terminal' in the search box.
3. Select the 'Terminal' program from the list.
4. In Terminal, enter the command:

```
sudo spctl --master-disable
```

Enter your account password.

If all was successful, checking has been disabled. It's best not to turn back on the Corrade.

5. We run the Safari browser.
6. We go to the site https://corrade.grimore.org/download/corrade/osx-x64/?C=M;O=D and download the last fresh ZIP file.
7. We are waiting for the download of the file.
8. Safari will be so nice to us that it will unpack the ZIP file we downloaded for us.
9. The launched Terminal or activate the previously launched one.
10. We go to ~/Downloads (by the way we confirm the window that we want to give Terminal access to this directory).

```
cd ~/Downloads
```

1. Do a directory listing (using the ls command) to see the name of the Corrade folder where the latest version was extracted. Change its name to Corrade using the mv command.

11. Go to the Corrade directory.
12. Launch Corrade:

```
13. ./Corrade
```

14. Further configuration is done via Nucleus, see the chapter: **A system from the Windows family**.

4.5. Corrade configuration files

The default configuration files are:

- Configuration.xml.default,
- Nucleus.xml.default.

Just replace their names with:

- Configuration.xml,
- Nucleus.xml.

Configuration.xml is automatically created when saving changes to Nucleus.

4.5.1. Configuration.xml

The Configuration.xml file contains the bot configuration. It is saved in XML format.

Key	Description
FirstName	avatar name
LastName	avatar name (default)
Password	the password for the avatar account consists of the prefix 1 and then the password is encoded in MD5, the password should be 6 - 16 characters long
LoginURI	url to grid, the default will be Second Life
Servers	defines the settings for embedded servers
Servers => HTTPServer	settings for the HTTP server
Servers => HTTPServer => Enable	server is on (1) or off (0)
Servers => HTTPServer => Prefixes => Prefix	specifies the HTTP address at which the HTTP server will be accessible (IP address: port)

Key	Description
MQTTServer	settings for the MQTT server
MQTTServer => Enable	server is on (1) or off (0)
MQTTServer => IPAddress	IP address where the server will be available
MQTTServer => Port	port where the server will be available
MQTTServer => MQTTCertificate => Path	certificate path for MQTT
MQTTServer => Compression	enable (1) or disable compression (0)
TCPServer	settings for the TCP server
TCPServer => Enable	server is on (1) or off (0)
TCPServer => IPAddress	IP address where the server will be available
TCPServer => Port	port where the server will be available
TCPServer => TCPCertificate => Path	certificate path for TCP
TCPServer => TCPCertificate => Protocol	protocol used
TCPServer => TCPCertificate => Password	the password for the certificate
WebSocketsServer	settings for WebSockets
WebSocketsServer => Enable	server is on (1) or off (0)
WebSocketsServer => WebSocketsCertificate => Path	the path to the certificate
WebSocketsServer => WebSocketsCertificate => Password	password for the certificate
WebSocketsServer => URL	The URL where the WebSockets server will be available
WebSocketsServer => Compression	enable (1) or disable compression (0)
StartLocations	defines the starting places
StartLocations => StartLocation => Region	takes one of the values: • home - teleport to a place marked as 'home', • last - teleport to the place where the bot

Key	Description
	recently logged out, • specific name of the region.
StartLocations => StartLocation => Position	position X, Y, Z where the bot should log in
Groups	defines bot groups and permissions to these groups
Groups => Notifications	defines the available notifications
Groups => Password	defines a group password, the password is saved as SHA1
Groups => Name	name of the group (existing in Second Life)
Groups => Permissions	defines the available permissions for the group
Groups => CacheMembers	specifies whether the group member cache is enabled (1) or disabled (0)
RLV => Enable	turns on the RLV
RLV => Blacklist	defines which behaviors are disabled
Range	determines the detection range of avatars and objects
IgnoreOfflineMessages	enables (10) or disables (1) the ability to receive offline messages
MessageLogging	save messages to file
Feedback	determines whether diagnostic feedback is to be sent to the manufacturer's Corrade server
ScriptLanguage	defines which data format is used by Corrade for communication (WAS or JSON)

Table 1 - Table with a description of individual variables

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE Configuration [<!ATTLIST Configuration xmlns:xsi CDATA #IMPLIED
xsi:noNamespaceSchemaLocation CDATA #IMPLIED>]>
<Configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:corrade-configuration-
```

```
schema">
      <FirstName>MyBot</FirstName>
      <LastName>Resident</LastName>
      <Password>$1$842e4818f8ede223c9b920d4f7425c9b</Password>
      <LoginURI>https://login.agni.lindenlab.com/cgi-bin/login.cgi</LoginURI>
      <Servers>
      <HTTPServer>
            <Enable>1</Enable>
            <Prefixes>
                  <Prefix>http://127.0.0.1:9199/</Prefix>
            </Prefixes>
      </HTTPServer>
      <MQTTServer>
            <Enable>0</Enable>
            <IPAddress>0.0.0.0</IPAddress>
            <Port>1883</Port>
            <MQTTCertificate>
                  <Path>mqtt.pfx</Path>
            </MQTTCertificate>
            <Compression>0</Compression>
      </MQTTServer>
      <TCPServer>
            <Enable>0</Enable>
            <IPAddress>0.0.0.0</IPAddress>
            <Port>8085</Port>
            <TCPCertificate>
                  <Path>tcp.pfx</Path>
                  <Password>corrade</Password>
                  <Protocol>Tls12</Protocol>
            </TCPCertificate>
            <Compression>0</Compression>
      </TCPServer>
      <WebSocketsServer>
            <Enable>0</Enable>
            <WebSocketsCertificate>
                  <Path>ws.pfx</Path>
                  <Password>corrade</Password>
            </WebSocketsCertificate>
            <URL>ws://0.0.0.0:8088</URL>
            <Compression>0</Compression>
      </WebSocketsServer>
      </Servers>
            <StartLocations>
                  <StartLocation>
                        <Region>Lorena Pink</Region>
                        <Position>
                        <X>9</X>
                        <Y>127</Y>
                        <Z>23</Z>
                  </Position>
                  </StartLocation>
            </StartLocations>
      <Groups>
            <Group>
                  <Notifications>
                        <Notification>group</Notification>
                        <Notification>message</Notification>
                        <Notification>notice</Notification>
                        <Notification>local</Notification>
                        <Notification>dialog</Notification>
                        <Notification>permission</Notification>
                        <Notification>invite</Notification>
                        <Notification>sit</Notification>
                        <Notification>teleport</Notification>
                        <Notification>inventory</Notification>
                        <Notification>wind</Notification>
                        <Notification>sound</Notification>
                        <Notification>terse</Notification>
                        <Notification>RLV</Notification>
                        <Notification>ownersay</Notification>
                        <Notification>preload</Notification>
                        <Notification>MQTT</Notification>
                        <Notification>lure</Notification>
                        <Notification>economy</Notification>
                        <Notification>crossing</Notification>
                        <Notification>collision</Notification>
                        <Notification>cache</Notification>
                        <Notification>avataraction</Notification>
                        <Notification>agentdata</Notification>
                        <Notification>tracker</Notification>
                        <Notification>statistics</Notification>
                        <Notification>scripts</Notification>
                        <Notification>primitives</Notification>
                        <Notification>parcelmap</Notification>
                        <Notification>map</Notification>
                        <Notification>location</Notification>
                        <Notification>heartbeat</Notification>
                        <Notification>effect</Notification>
                        <Notification>debug</Notification>
                        <Notification>conference</Notification>
                        <Notification>alert</Notification>
                        <Notification>avatargroup</Notification>
                        <Notification>chatterbox</Notification>
                        <Notification>typing</Notification>
                        <Notification>store</Notification>
                        <Notification>region</Notification>
```

```
                <Notification>particles</Notification>
                <Notification>objectim</Notification>
                <Notification>membership</Notification>
                <Notification>login</Notification>
                <Notification>estatelist</Notification>
                <Notification>configuration</Notification>
                <Notification>coarse</Notification>
                <Notification>avatars</Notification>
                <Notification>animation</Notification>
                <Notification>URL</Notification>
                <Notification>softban</Notification>
                <Notification>regionsayto</Notification>
                <Notification>outfit</Notification>
                <Notification>logs</Notification>
                <Notification>friendship</Notification>
                <Notification>displayname</Notification>
                <Notification>control</Notification>
                <Notification>colliders</Notification>
                <Notification>appearance</Notification>
                <Notification>balance</Notification>
            </Notifications>
            <Password>4e000b85758746ec818d53513c3d3e791822fdb6</Password>
            <Name>My Group</Name>
            <Permissions>
                <Permission>inventory</Permission>
                <Permission>movement</Permission>
                <Permission>grooming</Permission>
                <Permission>interact</Permission>
                <Permission>notifications</Permission>
                <Permission>talk</Permission>
                <Permission>group</Permission>
                <Permission>land</Permission>
                <Permission>mute</Permission>
                <Permission>execute</Permission>
                <Permission>bridge</Permission>
                <Permission>friendship</Permission>
                <Permission>database</Permission>
                <Permission>system</Permission>
                <Permission>directory</Permission>
                <Permission>economy</Permission>
            </Permissions>
            <Cookies />
            <CacheMembers>1</CacheMembers>
        </Group>
    </Groups>
    <RLV>
        <Enable>0</Enable>
        <Blacklist>
```

```
                    <Behaviour>sendim</Behaviour>
                    <Behaviour>tplm</Behaviour>
              </Blacklist>
        </RLV>
        <Range>64</Range>
        <Language />
        <PublishLanguage>1</PublishLanguage>
        <IgnoreOfflineMessages>1</IgnoreOfflineMessages>
        <MessageLogging>1</MessageLogging>
        <MultipleSimulatorConnections>0</MultipleSimulatorConnections>
        <Feedback>0</Feedback>
        <NotificationSettings>
              <AutoPrune>
                    <Enable>1</Enable>
                    <Conditions>
                          <Condition>
                                <StatusCode>404</StatusCode>
                                <Expression>sim.+?agni.lindenlab.com:[0-9]+?\/cap/[0-
9a-fA-F\-]+?$</Expression>
                          </Condition>
                    </Conditions>
              </AutoPrune>
              <Scripts>
                    <Update>1000</Update>
              </Scripts>
              <Colliders>
                    <Update>1000</Update>
              </Colliders>
              <Statistics>
                    <Update>1000</Update>
              </Statistics>
              <Wind>
                    <Update>1000</Update>
              </Wind>
              <Location>
                    <Update>1000</Update>
              </Location>
              <Terse>
                    <Update>1000</Update>
              </Terse>
        </NotificationSettings>
        <ScriptLanguage>JSON</ScriptLanguage>
</Configuration>
```

Text 1 - Bot configuration file

4.5.2. Nucleus.xml

Nucleus configuration.

- Prefix - Specifies the URL where Nucleus will be available, + stands for each IP address.
- Enabled - Determines whether the Nucleus is enabled (1) or disabled (0).
- CodeWord - defines the password to access the Nucleus.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE Configuration [
<!ATTLIST Configuration xmlns:xsi CDATA #IMPLIED xsi:noNamespaceSchemaLocation
CDATA #IMPLIED>
]>
<Configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:corrade-nucleus-schema">
      <Prefix>http://+:54377/</Prefix>
      <Enabled>1</Enabled>
      <CodeWord>nucleus</CodeWord>
</Configuration>
```

Text 2 - Nucleus configuration file

4.6. Corrade installation and removal as a system service

As standard, Corrade does not have the option to install the program as a system service, so below I present how to install Corrade as a service in popular operating systems.

4.6.1. Windows

In the contrib\windows folder you will find the following files:

- install-corrade-service.bat - installs Corrade as a service (named Corrade Resident),

- nssm.exe - a program that allows you to add a handle to a program that cannot be handled as a service by default; the program installs as a service and intercepts all system requests (start, restart, stop) for the specific program,
- nssm-GUI.bat - the above program, but the ability to add and edit via the graphical user interface,
- uninstall-corrade-service.bat - uninstall the service.

You can also install and uninstall the service yourself:

1. Run command prompt with administrator privileges in the root folder of the bot.
2. Enter and confirm the command: powershell.
3. Below are the commands to copy: line by line.

Attention! The following lines install Corrade as a service named: Corrade.

You can change the name of the service to yours, however, adjust all lines to include the new name of your service.

```
$currentDir = $(Get-Location).Path
$binPath = "$currentDir\Corrade.exe"
.\contrib\windows\nssm.exe install Corrade "$binPath"
.\contrib\windows\nssm.exe set Corrade AppDirectory "$currentDir"
.\contrib\windows\nssm.exe set Corrade AppExit Default Restart
.\contrib\windows\nssm.exe set Corrade DisplayName Corrade
.\contrib\windows\nssm.exe set Corrade ObjectName LocalSystem
.\contrib\windows\nssm.exe set Corrade Start SERVICE_AUTO_START
.\contrib\windows\nssm.exe set Corrade Type SERVICE_WIN32_OWN_PROCESS
.\contrib\windows\nssm.exe set Corrade AppThrottle 1500
.\contrib\windows\nssm.exe set Corrade AppRestartDelay 1000
```

4. close the command prompt window.

To uninstall the program as a service, type and confirm at the command prompt:

```
sc stop corrade
sc delete corrade
```

4.6.2. Linux

For Linux, we find the precompiled corrade.service file in contrib\linux, which is handled by the SystemD daemon. You just need to copy it, configure it, and include it in the system service startup daemon.

Open a command prompt (terminal), enter the commands one by one and confirm them:

Adding a new user named 'corrade'.

```
sudo useradd -m corrade
```

Setting a new password for the user 'corrade'.
Enter a new password twice for this user.

```
sudo passwd corrade
```

Adding 'corrade' to the group to run sudo command.

```
sudo usermod -aG sudo corrade
```

Logging in as 'corrade'.

Enter the previously established password.

```
su corrade
```

Go to the user's home directory.

```
cd ~
```

Download or copy the corrade to this folder.

Grant permissions for this folder to the new user and group: corrade.

```
chown  -R corrade:corrade corrade
```

Go to the corrade folder.

Then go to the folder with corrade.service.

```
cd contrib/linux/
```

copy the file to the systemd folder.

```
sudo cp -af corrade.service /etc/systemd/system/
```

edit this file:

```
sudo nano /etc/systemd/system/corrade.service
```

Here:

- in WorkingDirectory, correct the path to the corrade folder,
- in ExecStart correct the path to the corrade file (same as in WorkingDirectory only add /Corrade) to/home/corrade/corrade.

If everything is fine, save your changes and go back to the console.

```
sudo systemctl enable corrade.service
sudo systemctl start corrade.service
```

Go through the browser to the address: http://COMPUTER_IP_ADDRESS:54377 and configure the bot.

If something is wrong, check the bot and system logs, maybe the folder permissions are wrong...

4.7. Corrade.log and Openmetaverse.log

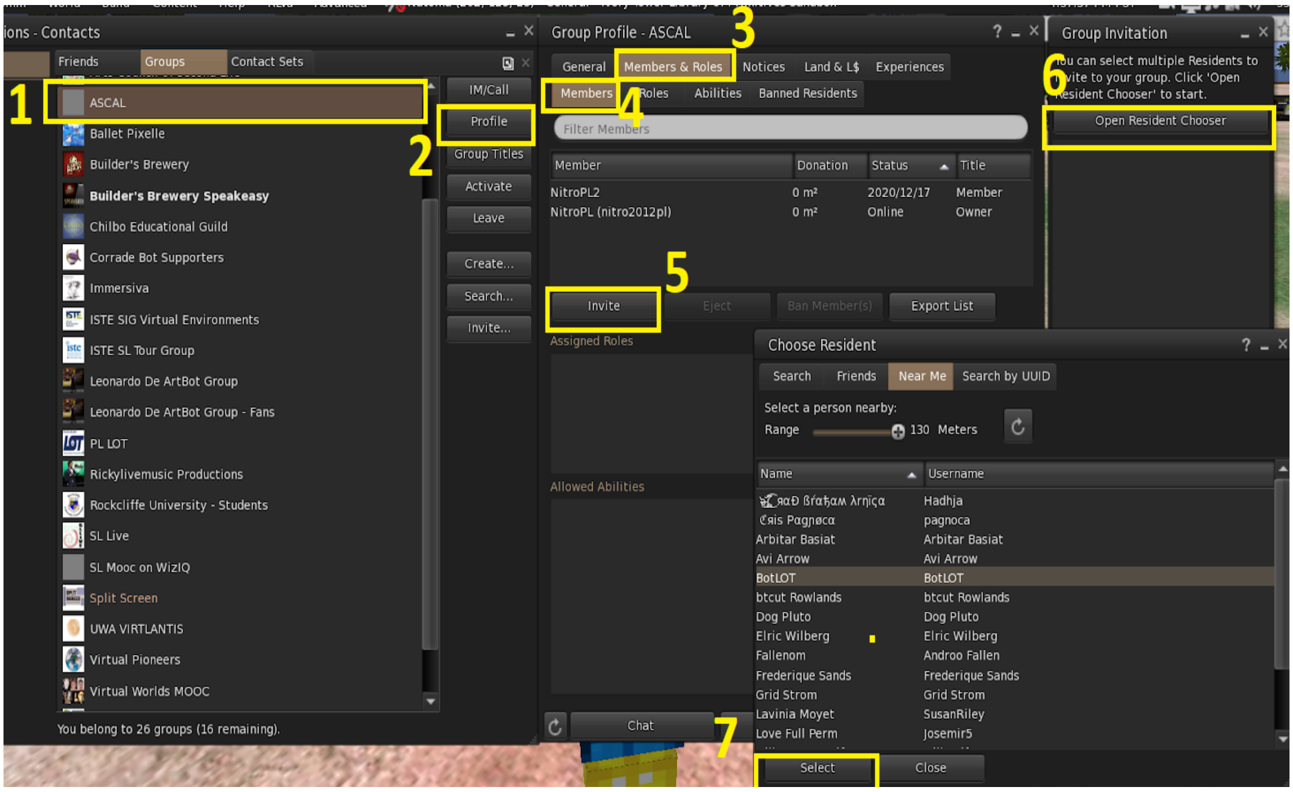
- **Logs**

  All text files to which Corrade registered the events are stored in the 'Logs' folder.

  - **Corrade.log** relates to the Corrade program,
  - **OpenMetaverse.log** refers to the world of Second Life,
  - .log files refer to the current log files,
  - files with the .logNUMER extension are files archived on a given day, e.g. Corrade.log20200912 refers to events recorded on 2020-09-12,
  - folder **Chat**
    - **IM** - saved private conversations with the bot, the file name is the UUID of the avatar with which the bot was talking,
    - **Local** - saved public chat conversations, the name of the file is the name of the region in which the bot was located,
    - **OwnerSay** - saved calls by registration from llOwnerSay() function (sent by prim/object owner), filename is prim UUID, owner, etc.

4.8. Adding a bot to a group in SL, giving it permissions, disabling the option of receiving group messages

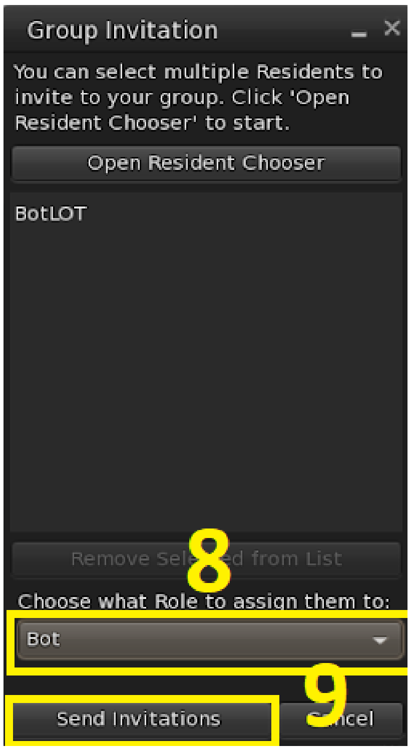
It's very easy to add a bot to a group and assign it a role with permissions.
A short chapter on how to do it using Firestorm Viewer.



Picture 4.21 - Selecting an avatar for a group invitation

1. Select a group from the list.
2. Go to the group details.
3. Go to the 'Members and roles' tab.
4. Go to the 'members' tab.
5. Click on the 'Invite' button.
6. Click on the button that allows you to select the avatars to be invited.

7. In this window, use the appropriate tab to find your bot, which you will add to the group. Enter its name in the appropriate field and put it on the search button, then add it to the invitation list. If everything is ok, press the 'Select' button.


Group Invitation
You can select multiple Residents to invite to your group. Click 'Open Resident Chooser' to start.
Open Resident Chooser
BotLOT
Remove Selected from List
8
Choose what Role to assign them to:
Bot
9
Send Invitations
Cancel

Picture 4.22 - Select a role and send an invitation to the avatar

8. We choose the role to which we will assign the bot. This role must be created beforehand, and it is best to assign group permissions to the bot beforehand.
9. We're sending an invitation.

Now you just need to log in as a bot and accept the group invitation.

4.9. Authorization notation system

There is a known permission system in Second Life, which is especially visible when editing objects. Corrade also supports this system by writing it in a specific way, e.g.

cdemvt------------cde-vtcdemvt

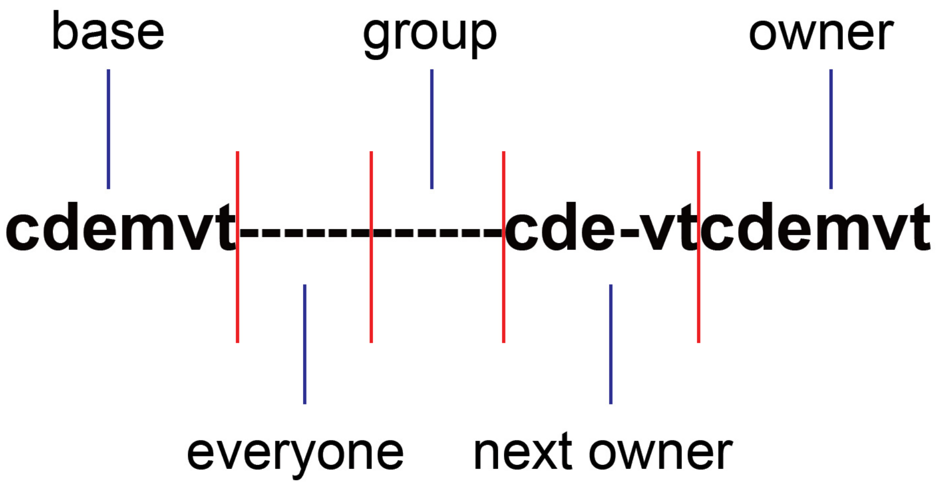
This entry consists of 5 segments, each of them consisting of 6 characters, which define the type of authorization.

The following authorization segments are: basic, everyone, group, next owner, owner.

Each character means:

- c (copy),
- d (damage) (used in the OpenSimulator simulator),
- e (export) (used in the OpenSimulator simulator),
- m (modify),
- v (move),
- t (transfer),
- - (dash) - no permission.

We will take a deeper look at a previously defined example.



Picture 4.23 - Distribution of object authority

- the first 6 characters - cdemvt - mean full permissions for the 'basic' mask,
- next 6 characters - ------ - mean no permissions for the mask 'everyone',
- next 6 characters - ------ - mean no permissions for the mask 'group',
- the next 6 characters - cde-vt - mean that the following permissions are set for the 'next owner' mask: copy, damage, export, move, transport, while the next owner cannot modify the object,
- the last 6 characters - cdemvt - indicate full permissions for the 'owner' mask.

4.10. Notifications

A notification is a bot function that receives data from Second Life in real time and based on which the bot can trigger a specific action.

For example, there is an 'alert' notification that receives messages regarding the region. Most avatars receive information about restarting the region. If the bot receives such information, it can, for example, temporarily teleport to another location.

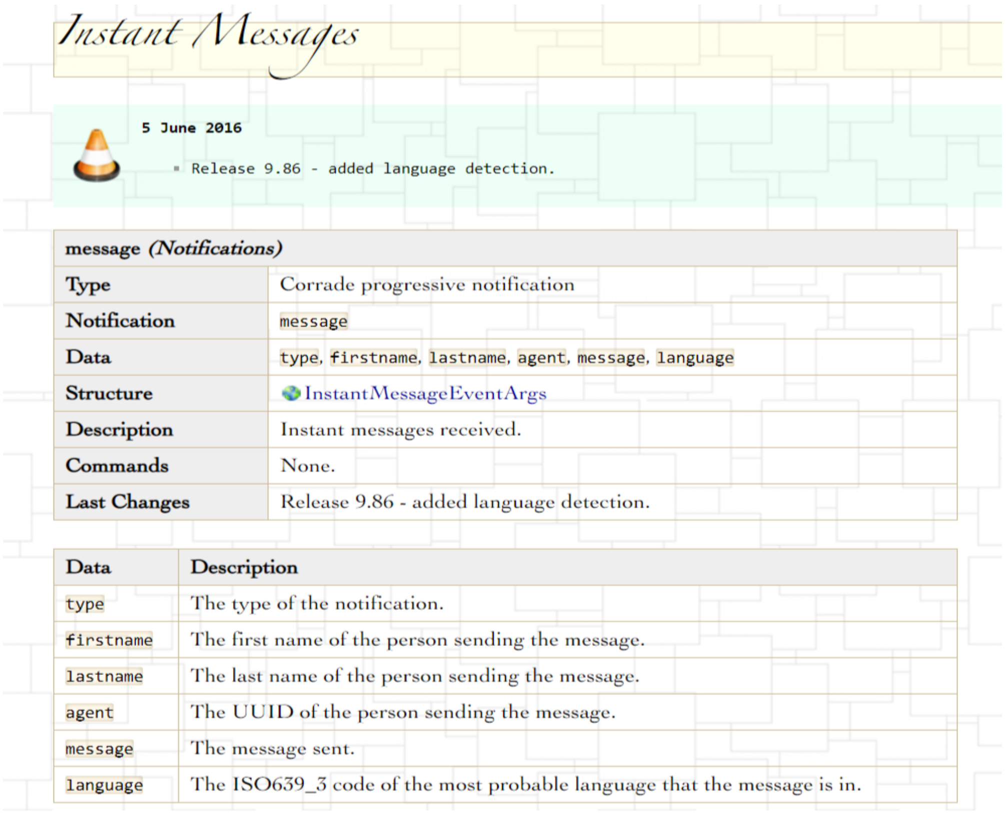
We have another example 'balance' notification that keeps you informed about the bot's account balance. For example, a bot can act as a seller of goods or a land rental manager in a region, which therefore accepts payments and returns the surplus (e.g. you can rent land for a maximum of a month and someone has paid for 2 months, so the bot has to return the money for an additional month). If the bot receives a notification that there is no money in his account (the balance is zero or even negative) he can send it to his administrator in the form of an IM or an email with a request to top up the money at the checkout.

The last example I would like to give is the 'terse' notification, which returns information about the new position and rotation of the avatar or object. You can use this notification, for example, to track avatars in a region and check by their position that they do not violate some area accessible only to authenticated people. If such a situation occurs, you can first send a warning to such a person via a private message, and after e.g. 30 seconds, ban him temporarily for 1 hour.

Above, I just presented possible scenarios with the use of some notifications. You can find all notifications on the Corrade software page[13].

Remember to clean the bot from old notification URLs. This can be done using the 'notify' command with the 'action' option set to 'purge' or before starting the bot, delete the State\Notifications.xml file, for this purpose it is best to create a batch file, where first delete the State\Notifications.xml file and then the bot is launched. If you have additional material, see the bot_installer\corrade\bot_ai\start.bat file.

I will now show the code in LSL how to install the notification, here 'message'[14] it will be used to receive private messages from avatars who write to it.


Instant Messages

5 June 2016

- Release 9.86 - added language detection.

message *(Notifications)*	
Type	Corrade progressive notification
Notification	`message`
Data	`type`, `firstname`, `lastname`, `agent`, `message`, `language`
Structure	InstantMessageEventArgs
Description	Instant messages received.
Commands	None.
Last Changes	Release 9.86 - added language detection.

Data	Description
`type`	The type of the notification.
`firstname`	The first name of the person sending the message.
`lastname`	The last name of the person sending the message.
`agent`	The UUID of the person sending the message.
`message`	The message sent.
`language`	The ISO639_3 code of the most probable language that the message is in.

Picture 4.24 - Documentation for the 'message' notification

The table on the page lists the data returned from each field: notification type; name, surname, UUID of the sender; message sent from the sender; the detected language of the message.

Below is the full LSL code for installing and using this notification.

13 Notifications - https://grimore.org/secondlife/scripted_agents/corrade/api/notifications
14 https://grimore.org/secondlife/scripted_agents/corrade/api/notifications/message

We define variables for bot UUID, group password, group name.
Under the URL variable, we will store the assigned URL from Second Life from which notifications will come.
The TAG variable will hold the name of the script.
Function:

- setDebug() - sends a message to the owner of the prim/object (for me it's a bot),
- strReplace() - replaces characters in the text,
- wasCSVToList() - WAS function to convert CSV to list,
- wasKeyValueGet() - WAS function to get a value based on a key from the data,
- wasKeyValueEncode() - WAS function for data encoding based on RFC 4180,
- wasURLUnescape() - WAS function for decoding data based on RFC 4180,
- wasKeyValueToJSON() - WAS function that transforms linked data '&' to JSON,
- wasJSONToKeyValueData() - WAS function, which converts JSON into linked data '&'.

In the default state it assigns the script name to a variable, shows the amount of free memory for the script, then calls the next state 'ReadConfigurationNotecard', where this state reads a note named 'configuration' which is placed in the same prim as the script.
The note must contain the following lines:

- BOTID=BOT'S_UUID,
- GROUP=GROUP_NAME,
- PASSWORD=GROUP'S PASSWORD.

After getting the variables from the note and assigning them to the variables in the script, another state called 'url' is triggered, in which an attempt is made to get the URL from Second Life. If everything is successful here, it moves to the next state called 'NotifyInstall'. In this state, we install the expected notification. In the URL parameter, we provide the URL variable to which we previously downloaded the URL assigned by Second Life. Of course, this could well be a URL to a script on an external server.
We send the entire parameter array to the bot in JSON format.

In http_request() we wait for data to be received from the bot, we process it in such a way that we just check if our command has been executed, if it is successful, we go to the next state called 'sense', if there is an error, it is executed state 'preNotifyInstall', which basically reverts to the current state as it will not be possible to recall the current state directly.

In the sense state, we 'listen to' incoming messages from the 'message' notification on an ongoing basis, here we also assigned the values discussed earlier in the table on the notification page to the variables.

It is still up to us what we do with it.

```
key CORRADE;
string PASSWORD;
string GROUP;
string URL = "";
string TAG;
string val;
list lmessage;
string sjump = "";
string body;

integer line;
list tuples = [];

list lprepare = [];

setDebug(string msg)
{
llOwnerSay("["+ TAG +"] " + msg);
}

string strReplace(string str, string search, string replace) {
    return llDumpList2String(llParseStringKeepNulls((str = "") + str, [search], []),
replace);
}

list wasCSVToList(string csv)
{
    list l = [];
    list s = [];
    string m = "";
    do
    {
        string a = llGetSubString(csv, 0, 0);
        csv = llDeleteSubString(csv, 0, 0);
        if(a == ",")
        {
            if(llList2String(s, -1) != "\"")
            {
                l += m;
                m = "";
                jump continue;
            }
            m += a;
            jump continue;
        }
        if(a == "\"" && llGetSubString(csv, 0, 0) == a)
        {
            m += a;
            csv = llDeleteSubString(csv, 0, 0);
            jump continue;
        }
        if(a == "\"")
        {
            if(llList2String(s, -1) != a)
            {
                s += a;
                jump continue;
            }
            s = llDeleteSubList(s, -1, -1);
            jump continue;
        }
        m += a;
        @continue;
    } while(csv != "");
    return l + m;
}

string wasKeyValueGet(string k, string data)
{
  if(llStringLength(data) == 0) return "";
  if(llStringLength(k) == 0) return "";
  list a = llParseStringKeepNulls(data, ["&", "="], []);
  integer i = llListFindList(llList2ListStrided(a, 0, -1, 2), [ k ]);
  if(i != -1){
    string ret = llList2String(wasCSVToList(wasURLUnescape(llList2String(a, 2*i+1))),
0);
    ret = strReplace(ret, "\\r", "");
    ret = strReplace(ret, "\\n", "");
    ret = llStringTrim(ret, STRING_TRIM);

    return ret;
  }
  return "";
}

string wasKeyValueEncode(list data)
{
  list k = llList2ListStrided(data, 0, -1, 2);
  list v = llList2ListStrided(llDeleteSubList(data, 0, 0), 0, -1, 2);
  data = [];
  do
  {
    data += llList2String(k, 0) + "=" + llList2String(v, 0);
    k = llDeleteSubList(k, 0, 0);
    v = llDeleteSubList(v, 0, 0);
  } while(llGetListLength(k) != 0);
  return llDumpList2String(data, "&");
}

string wasURLEscape(string i)
{
  string o = "";
  do
  {
    string c = llGetSubString(i, 0, 0);
    i = llDeleteSubString(i, 0, 0);
    if(c == "") jump continue;
    if(c == " ")
    {
      o += "+";
      jump continue;
    }
    if(c == "\n")
    {
      o += "%0D" + llEscapeURL(c);
      jump continue;
    }
    o += llEscapeURL(c);
    @continue;
  } while(i != "");
  return o;
}

string wasURLUnescape(string i)
{
    return llUnescapeURL( llDumpList2String( llParseStringKeepNulls( llDumpList2String(
llParseStringKeepNulls( i, ["+"], [] ), " " ), ["%0D%0A"], [] ), "\n" ) );
}

default
{
    state_entry()
    {
        TAG = llGetScriptName();
        setDebug("Free memory: " + (string)llGetFreeMemory());
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state ReadConfigurationNotecard
{
    state_entry()
    {
        if(llGetInventoryType("configuration") != INVENTORY_NOTECARD)
        {
            setDebug("Sorry, could not find a configuration inventory notecard.");
            return;
        }
        setDebug("Reading configuration file...");
        line = 0;
        llGetNotecardLine("configuration", line);
    }

    dataserver(key id, string data)
    {
        if(data == EOF)
        {
            if(llGetListLength(tuples) % 2 != 0)
            {
                setDebug("Error in configuration notecard.");
                return;
            }

            CORRADE = llList2Key(tuples, llListFindList(tuples, ["BOTID"])+1);

            if(CORRADE == NULL_KEY)
            {
                setDebug("Error in configuration notecard: BOT ID KEY");
                return;
            }

            GROUP = llList2String(tuples, llListFindList(tuples,["GROUP"])+1);

            if(GROUP == "")
            {
                setDebug("Error in configuration notecard: GROUP");
                return;
            }

            PASSWORD = llList2String(tuples, llListFindList(tuples, ["PASSWORD"])+1);

            if(PASSWORD == "")
            {
                setDebug("Error in configuration notecard: PASSWORD");
                return;
            }
            state url;
        }
        if(data == "") jump continue;
        integer i = llSubStringIndex(data, "#");
        if(i != -1) data = llDeleteSubString(data, i, -1);
        list o = llParseStringKeepNulls(data, ["="], []);
        string k =
llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String(o,0),
STRING_TRIM), ["\""], []), "\"");
        string v =
llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String( o, 1 ),
STRING_TRIM), ["\""], [] ), "\"");
        if(k == "" || v == "") jump continue;
        tuples += k;
        tuples += v;
        @continue;
        llGetNotecardLine("configuration", ++line);
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state url
{
    state_entry()
    {
        if ((GROUP == "BOTID") || (GROUP == "GROUP") || (GROUP == "PASSWORD")){
            llResetScript();
            return;
        }
        if ((PASSWORD == "BOTID") || (PASSWORD == "GROUP") || (PASSWORD == "PASSWORD"))
{
            llResetScript();
            return;
        }

        setDebug("Requesting URL...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        llRequestURL();
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(method != URL_REQUEST_GRANTED) return;
        URL = body;
        setDebug("Got URL...");
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state NotifyInstall {
    state_entry() {
        llOwnerSay("Binding to the Message Alert notification...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        lprepare =     [
            "group", wasURLEscape(GROUP),
            "password", wasURLEscape(PASSWORD),
            "callback", wasURLEscape(URL),
            "URL", wasURLEscape(URL),
            "command", "notify",
            "action", "set",
            "type", "message",
            "_script", TAG
        ];

        llOwnerSay(wasKeyValueEncode(lprepare));
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(wasKeyValueGet("success", body) != "True") {
            llOwnerSay("Failed to bind to the Message Alert notification...");
            state preNotifyInstall;
        }
        // DEBUG
        llOwnerSay("IM Alert notification installed...");
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state preNotifyInstall
{
    state_entry()
    {
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state sense {
    state_entry() {
        setDebug("Listen Region Restart...");
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        string firstname = wasKeyValueGet("firstname", body);
        string lastname = wasKeyValueGet("lastname", body);
        string agent = wasKeyValueGet("agent", body);
        string message = wasKeyValueGet("message", body);


//WHAT YOU DO NEXT?
```

```
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

Text 3 - Full script in LSL to install 'message' notification with data reception

4.11. Error codes

When Corrade cannot process the request, it returns a reply with the error code and message.

Below is an extensive table with error codes.

Error Code	Message (English)
48	text too long
303	unable to post proposal
337	no map items found
458	group members are by default in the everyone role
616	timeout ejecting agent
1253	cannot remove owner role
1382	unknown sound requested
1458	unknown update type
1488	no dialog specified
1536	primitive not for sale
1691	unable to post event
2021	invalid version provided
2087	no avatars provided
2169	insufficient funds
2188	could not get parcel info data
2193	no avatars to ban or unban
2380	unknown image format provided
2432	invalid grab position
2707	could not find title

3098	invalid proposal quorum
3475	invalid scale
3638	no data provided
4541	group invite not found
4797	friendship offer not found
4994	unable to decode asset data
5034	unable to create item
5762	primitives already linked
6097	unknown inventory type
6330	no notice found
6617	dialog button not found
7140	no SIML response received
7168	timeout getting profile pick
7255	timeout transferring asset
7457	no permissions for item
7628	invalid proposal duration
7703	general error
8169	SQL execution failed
8241	could not find classified
8339	no event identifier provided
8411	could not start process
8653	command not found
8842	unknown tree type
8846	no host provided
9111	timeout getting l and users

9348	unknown animation action
9541	timeout uploading item
9703	group not configured
9924	first life text too large
9935	maximum group list length reached
10348	primitives already delinked
10374	timeout retrieving estate covenant
10522	failed to read log file
10691	timeout mapping friend
10776	group schedules exceeded
10945	invalid surface coordinates
11050	timeout getting group roles
11229	timeout getting group titles
11287	could not create role
11502	not in group
11910	no executable file provided
11979	could not retrieve pick
12320	unable to upload item data
12408	group synchronization failed
12758	no role name specified
13030	no chatlog path provided
13053	too many characters for group name
13296	invalid binormal vector
13399	no access token provided
13491	ban would exceed maximum ban list length

13712	timeout retrieving estate list
13764	status not found
13857	unexpected item in path
14337	unknown restart action
14634	could not eject agent
14989	no avatars found
15345	already in group
15517	eject needs demote
15719	timeout retrieving group ban list
15964	transfer would exceed maximum count
16233	invalid asset
16263	could not retrieve attachments
16450	could not retrieve mute list
16572	invalid viewer effect
16667	empty terrain data
16927	no estate powers for command
17019	unable to set home
17894	description would exceed maximum size
18463	could not retrieve object media
18490	invalid interval
18680	no pattern provided
18737	unable to process data
19011	ambiguous path
19059	no l and rights
19143	timeout meshmerizing object

19325	too many characters for group title
19343	unable to retrieve data
19862	agent not in group
20048	timeout getting avatar data
20303	notice does not contain attachment
20541	timeout getting top scripts
20547	invalid width
20822	too many characters for notice subject
20900	could not leave group
20900	invalid xml path
21106	message may not contain HTML
21160	unable to write file
21296	no topic provided
21718	no location provided
21743	event posting rejected
21833	invalid date
21894	AI feature not enabled
22119	no proposal to reject
22576	timeout requesting to set home
22655	expected folder as target
22693	primitive not found
22733	unable to join group chat
22737	object not found
22786	conference member not found
22970	timeout getting profile classified

23114	timeout waiting for sensor
23123	nucleus server error
23309	friend offline
23364	timeout getting script state
23570	invalid schedules provided
23716	classified not found
23805	no session specified
23926	unable to delete event
23932	no position provided
24939	group not open
25003	no category provided
25119	unable to retrieve transactions
25252	no SQL string provided
25329	failed rezzing child primitive
25420	could not retrieve classified
25426	timeout getting group role members
25897	unknown estate list action
25984	inventory item not found
26178	too many characters for notice message
26356	timeout retrieving group notices
26393	timeout updating mute list
26623	unable to create folder
26715	mute entry not found
26749	timeout modifying group ban list
27605	cannot eject owners

27910	timeout getting parcel list
28002	group not found
28087	no database file configured
28126	auto return time outside limit range
28247	maximum amount of classifieds reached
28353	invalid feed provided
28429	unknown move action
28487	invalid normal vector
28613	could not sit
28625	maximum manager list length reached
28866	no permissions provided
28891	invalid terraform action
29345	empty attachments
29438	no task specified
29512	empty wearables
29530	could not read XML file
29745	attachments would exceed maximum attachment limit
29947	unknown agent access
30129	could not get current groups
30207	friend not found
30293	name too large
30384	timeout downloading terrain
30473	no group power for command
30556	no description for status
31049	no button index specified

31126	too many characters for proposal message
31237	timeout leaving group
31267	agent has been banned
31381	timeout waiting for execution
31417	too many characters for group role description
31493	unable to go home
31868	no database path provided
32157	no consumer key provided
32164	teleport failed
32355	execution returned no result
32362	timeout getting parcels
32366	the agent already is a friend
32404	timeout creating group
32453	could not get parcel info data
32472	role not found
32528	agent is soft banned
32709	unknown wearable type
32923	timeout getting profile
33047	failed rezzing root primitive
33381	unable to reach events page
33413	cannot delete the everyone role
33564	unable to save configuration
33714	could not retrieve wearables
33717	no flags provided
33821	no notice provided

33994	unable to retrieve form parameters
34084	timeout getting group roles members
34379	proposal rejected
34749	the specified folder contains no equipable items
34869	already subscribed to feed
34964	timeout getting region
35198	could not set display name
35277	notecard message body too large
35316	parcel not for sale
35392	could not join group
35447	region not found
36068	type can only be voice or text
36121	invalid offset
36123	teleport throttled
36616	invalid item type
36675	no server provided
36684	unable to upload item
36896	no index provided
37211	timeout creating item
37470	timeout getting group members
37559	destination too close
37841	could not get primitive properties
38125	unable to obtain money balance
38184	invalid price
38271	script compilation failed

38278	unable to read file
38289	timeout waiting for display name
38504	invalid permissions
38609	timeout changing links
38624	invalid secret provided
38798	invalid face specified
38858	effect not found
38931	no label or index specified
38945	no source specified
39016	timeout getting role powers
39359	unknown type
39391	expected item as source
39613	maximum number of groups reached
39647	mute entry already exists
39787	timeout reaching destination
39921	could not set agent access
40491	no transactions found
40665	no name provided
40762	no consumer secret provided
40773	invalid number of items specified
40901	SIML not enabled
40908	cannot delete a group member from the everyone role
41007	no secret provided
41190	invalid terraform brush
41211	no terraform brush specified

41256	maximum number of roles exceeded
41257	unable to post divorce
41574	timeout rezzing primitive
41612	unknown date time stamp
41676	unknown top type
41810	invalid proposal majority
41969	unable to start conference
42051	unknown access list type
42140	group already configured
42216	no channel specified
42248	unable to save Corrade configuration
42249	inventory offer not found
42351	timeout starting conference
42536	maximum user list length reached
42798	timeout retrieving notice
42903	unknown image format requested
43003	unknown syntax type
43156	no item specified
43615	setting permissions failed
43671	unable to revoke proposal
43683	description too large
43713	fly action can only be start or stop
43767	proposal already sent
43780	timeout during teleport
43898	session not found

43982	invalid amount
43982	no folder specified
44059	unable to get event identifier
44397	timeout uploading item data
44447	unknown directory search type
44537	unknown horde balancer
44554	primitives not in same region
45074	link would exceed maximum link limit
45173	invalid angle provided
45364	could not set script state
45568	could not retrieve group ban list
46316	timeout requesting sit
46612	no partner found
46804	could not send message
46858	timeout receiving SIML response
46942	empty classified name
46990	unknown estate list
47101	no message provided
47172	timeout getting top colliders
47350	no type provided
47469	timeout getting picks
47571	too many or too few characters for display name
47624	invalid pay target
47808	cannot remove user from owner role
48110	no history found

48467	no Corrade permissions
48775	no effect UUID provided
48899	empty pick name
49113	pick not found
49640	notification not allowed
49722	failed to get display name
50003	unable to reject proposal
50203	no permission for parcel
50218	item is not a script
50405	maximum amount of picks reached
50593	maximum ban list length reached
51050	script permission request not found
51086	no name or UUID provided
52299	too many or too few characters in message
52751	timeout requesting price
53059	could not update parcel list
53066	path not found
53274	unknown material type
53494	too many characters for event description
53549	no description provided
53829	timeout getting group account summary
53947	invalid days
54084	no peers matching context
54154	scale would exceed building constraints
54214	could not create group

54450	unable to agree to ToS
54456	unknown effect
54528	description may not contain HTML
54668	unable to authenticate
54854	avatar not in range
54956	timeout retrieving estate info
55051	message too long
55091	no access token secret provided
55110	unknown mute type
55394	no matching dialog found
55755	MQTT publish failed
55979	unknown direction
56094	no schedule found
56345	unable to reach partnership page
56379	second life text too large
56901	no terraform action specified
57005	timeout uploading terrain
57085	failed to download asset
57196	no duration provided
57961	agent not found
58183	invalid rotation
58212	platform not supported
58478	could not get parcel resources
58493	default folder not found
58619	could not terraform

58751	name may not contain HTML
58870	unable to divorce
59048	invalid asset data
59103	timeout getting classifieds
59282	no path provided
59524	invalid position
59526	unknown action
60025	unknown asset type
60073	teleport lure not found
60269	asset upload failed
60427	could not get current outfit folder
60515	position would exceed maximum rez altitude
61018	folder not found
61067	unable to convert to requested format
61113	unknown entity
61317	invalid workers provided
61473	invalid status supplied
61492	unknown language
61983	unable to accept proposal
62130	invalid texture coordinates
62531	could not get l and resources
62646	effect UUID belongs to different effect
62753	could not rebake
63024	unable to load configuration
63486	invalid height

63597	no date provided
63713	timeout joining group
63915	no time provided
64123	invalid mute target
64179	unable to serialize primitive
64368	unknown grass type
64390	could not find parcel
64420	primitives are children of object
64423	timeout getting folder contents
64450	unknown sift
64698	could not compile regular expression
64868	invalid proposal text
65003	friend does not allow mapping
65101	no search text provided
65241	no title provided
65303	invalid url provided
65327	invalid notification types

Table 2 - Error codes

4.12. Restrictions on Second Life

The world of Second Life has its limitations, which have been imposed by the project development team, so that being in this world is fluid and not bothered by certain problems.

These limitations should be considered when building your own bot:

- Number of prims

One region can make 15,000 to 30,000 prims available to all. We have to think how many prims we want (including the reserve) and the minimum and maximum plot size in the region. The size of the plot and the number of primes are related - the larger the plot, the more primes are available, however this number cannot exceed the maximum number of primes in the region. On average, for 1024 m^2 of a plot with 469 prims, we will pay approximately $2.00, it all depends on which tenant we will rent the plot from.

- The maximum length of the message is 1024 characters, longer Text will be truncated to this size,
- llInstantMessage() - max. the number of messages sent by all sites is: 2500 messages/30 min. which gives about 83 messages/1 s,
- Maximum size of the LSL script in mono mode: 64KB,
- Maximum coordinates:
  - X - 256 m,
  - Y - 256 m,
  - Z - 4,096 m.

4.13. Communication with the bot

First, let's deal with the communication with the bot via the LSL script that we create inside the prim. It also all depends on whether this prime will be part of the bot's clothes or whether it will be standing somewhere on the plot.

If the prim is part of the bot's clothes, then the prim must be owned by the bot (we create prim as a bot - it's best to log in as a bot). This prime will always be at the bottom. The only downside to this is that when the bot is teleported during a region restart to another location where scripts are forbidden by other avatars, the prim script that is part of the avatar's clothes will not run. To deal with this problem, it is best to create a script, e.g. in PHP, which will be run e.g. every 5 - 15 minutes by the system task schedule, which will check if the bot is on the right plot [the script is included in additional materials]. For this solution, it is best to use the llOwnerSay() command[15], which forwards the message directly to the prim owner and has virtually no limits on the number of messages sent.

15 http://wiki.secondlife.com/wiki/LlOwnerSay

If the prim will be somewhere on the plot, it is best to use the llInstantMessage() function in the script[16], which sends a private message to the bot, no matter where it is in Second Life. This feature has a message limit of 2500/30 minutes. and applies to all properties in a given region. If the region is reset, it can send info to the bot so that it will teleport back to the plot. The downside of this solution is that when the region is restarted, we do not have access to this prim with this script. It is also worth mentioning here that if we choose this solution, it is best to check in the script from time to time whether the bot is actually present in Second Life.

What solution should you choose here? It really depends on us. I prim with the scripts connected as invisible bot clothes, I do not have any trouble editing them, I just log in manually as a bot, edit, save, turn off the browser, turn on Corrade and check if it works according to my expectations, if not, then I turn off the program and repeat the steps. However, there are also primes that use the latter solution.

Another thing to discuss is the structure of our bot's heart and brain in the virtual world. Script everything in LSL in Second Life is unprofitable and will quickly take revenge on us, because programming in LSL is very limited, and a single script cannot exceed 64KB in Mono compilation mode, where 16KB is allocated to: bytecode, heap, stack.

This script limitation has the advantage of reducing the script resource consumption, as there are several regions running on one server and countless scripts running on each of them. This makes everything run smoothly.

The downside, however, is the inability to create scripts that would do advanced things in the virtual world. You can either break a large script into successive smaller scripts and try to combine them with each other, which would be a laborious task in Second Life, or transfer a large part of the bot functionality beyond Second Life and receive the status of the implemented parts of the functionality only in the virtual world. It also has the advantages of smaller LSL files in prim, and a lower risk of getting a memory limit exceeded error for the compiled code (e.g. 'Script Execution Error: Stack Collision', 'ERROR: Bytecode Set, Failure - Out of Memory'). Regardless of which option we choose, remember that prim cannot contain many LSL scripts - they will cause a very high load on the server, and when the region manager or Second Life service detects it, then it can order us to slim down the scripts in prim or order us to leave the region.

16 http://wiki.secondlife.com/wiki/LlInstantMessage

Therefore, the best idea is to move the entire processing process by the bot outside the Second Life environment, and in Second Life itself, in LSL scripts only retrieve the execution status by the external environment. In this situation, the 'execute' command from the bot API comes in handy, which allows you to execute any command with parameters in the environment of the system on which the bot is running. Additionally, the bot handles requests to invoke bot API commands over HTTP, MQTT, TCP or WebSockets, however HTTP is the most commonly used. What does this give us? By combining these two properties, we can run a console shell of any programming language, such as Python, Java, C#, C++, PHP, etc., which will allow you to make HTTP requests to the bot and receive responses from it. It also has another plus in the form of the possibility of connecting cloud computing, etc.

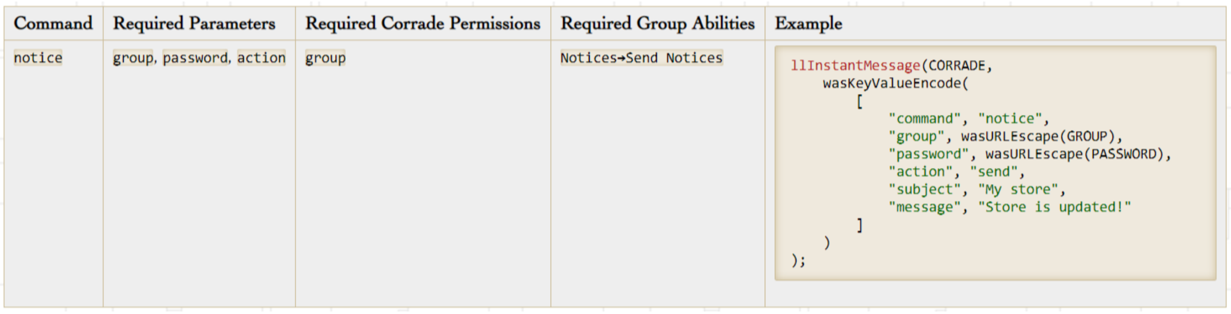
4.14. Practical use of the bot on the example of sending a group advertisement.

Time for our bot to do something specific besides providing only dry information in this book. All commands can be found on the website grimore.org[17].
Although we will choose one command, we will discuss all use cases, using additional functions prepared by WAS and mine on example scripts written in LSL and PHP, sending data to the bot using WAS and JSON.

Our example command will be 'notice'[18], which allows you to send a group advertisement in Second Life. So we come to the command detail page, which is easy to read and understand. We have at our disposal: a list of changes, examples of use, a table with parameters, etc.

First, let's take a look at the table below.

17 https://grimore.org/secondlife/scripted_agents/corrade/api/commands
18 https://grimore.org/secondlife/scripted_agents/corrade/api/commands/notice



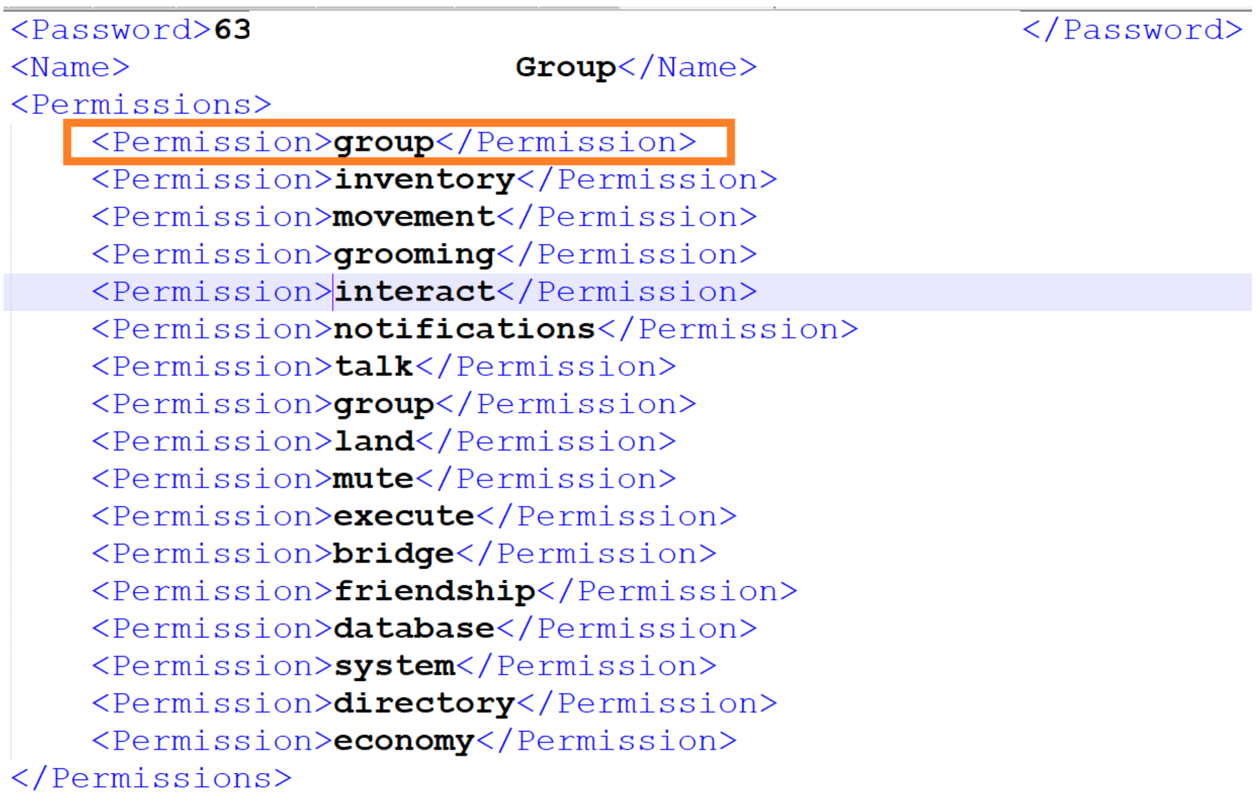
Command	Required Parameters	Required Corrade Permissions	Required Group Abilities	Example
notice	group, password, action	group	Notices→Send Notices	see below

```
llInstantMessage(CORRADE,
    wasKeyValueEncode(
        [
            "command", "notice",
            "group", wasURLEscape(GROUP),
            "password", wasURLEscape(PASSWORD),
            "action", "send",
            "subject", "My store",
            "message", "Store is updated!"
        ]
    )
);
```

Picture 4.25 - 'Notice' command - required parameters

Here we see the required parameters: group, password (they will be repeated over and over again with other commands), action (its detailed description is in the next table at the bottom of the page).

The next column is 'Corrade Permissions Required'. These permissions must be enabled in the bot group in the Corrade configuration to be able to operate the command. The easiest way is to open the Configuration.xml file and check if there is an appropriate entry for the group. If there is no such entry, we can add the permission to the file ourselves, then save the file and restart the bot.



```
<Password>63                                           </Password>
<Name>                      Group</Name>
<Permissions>
    <Permission>group</Permission>
    <Permission>inventory</Permission>
    <Permission>movement</Permission>
    <Permission>grooming</Permission>
    <Permission>interact</Permission>
    <Permission>notifications</Permission>
    <Permission>talk</Permission>
    <Permission>group</Permission>
    <Permission>land</Permission>
    <Permission>mute</Permission>
    <Permission>execute</Permission>
    <Permission>bridge</Permission>
    <Permission>friendship</Permission>
    <Permission>database</Permission>
    <Permission>system</Permission>
    <Permission>directory</Permission>
    <Permission>economy</Permission>
</Permissions>
```

Picture 4.26 - Permissions in the configuration file

The second way is to configure the Nucleus website, where in a given group we mark the missing permissions, but after saving the changes, Nucleus will reload the bot configuration itself, so you do not have to manually restart the bot yourself.

Another column that may appear is 'Required Group Abilities'. They inform what permissions the bot on the Second Life side should have to perform a given action.
This usually applies to the group or group role to which the bot has been assigned.
Group permissions will also be required for the following commands:

- ban,
- setgroupdata,
- softban,
- batcheject,
- moderate,
- notice,
- eject,
- setrolepowers,
- createrole.

We have to assign the bot the possibility of sending group advertisements. It's best to create a new role to which we assign only the bot and enable the required permissions for this role. The last column shows an example of using the command.

Now let's look at the next table.
All parameters and sub - parameters for a given command are listed here. We read the table from the left side.
We have parameters in the first column. For our command, there is only one, called 'action', which is required when uploading to the bot.
In the second column, we have the possible values that have been assigned to 'action', for our command they are: send (sending the advertisement), list (displaying all group advertisements), accept (accepting the attachment from the advertisement), decline (rejecting the attachment from the advertisement) .
In our case, we choose 'send' because we will send the advertisement.

Next, we have the following parameters that can be set - message (ad message), subject (ad subject), item (ad attachment), permissions (permissions to the ad attachment). We can choose the ones that suit us. We choose 'message' and 'subject'.

Our LSL code with sending data using WAS will look like this:

```
key ownerID = "YOUROWNERID"
key CORRADE;
string PASSWORD;
string GROUP;
string URL = "";
string TAG;
string val;
list lmessage;
string sjump = "";
string body;
key query;

integer line;
list tuples = [];

setDebug(string msg)
{
        llInstantMessage(ownerID, "["+ TAG +"] " + msg);
}

integer StringMatch(list lwhatsearch, string sinwhere){
  sinwhere = llToLower(sinwhere);
  sinwhere = strReplace(sinwhere, "!", "");
  sinwhere = strReplace(sinwhere, "?", "");

  list lsinwhere = llParseString2List(sinwhere,[" "],[]);

  integer imatch = llGetListLength(lwhatsearch);
  integer imatched = 0;

  integer i=0;
  for(i=0; i<imatch; i++){

    list lwords = llParseString2List(llList2String(lwhatsearch,i),["|"],[]);

    integer j=0;
    for(j=0; j<llGetListLength(lwords); j++){
      if(llListFindList(lsinwhere, [llList2String(lwords,j)]) >= 0){
        imatched++;
      }
    }
    }
    if(imatch == imatched){
        return 1;
    }else{
        return 0;
    }
}

string strReplace(string str, string search, string replace) {
    return llDumpList2String(llParseStringKeepNulls((str = "") + str, [search], []), replace);
}

list wasCSVToList(string csv)
{
    list l = [];
    list s = [];
    string m = "";
    do
    {
        string a = llGetSubString(csv, 0, 0);
        csv = llDeleteSubString(csv, 0, 0);
        if(a == ",")
        {
            if(llList2String(s, -1) != "\"")
            {
                l += m;
                m = "";
                jump continue;
            }
            m += a;
            jump continue;
        }
        if(a == "\"" && llGetSubString(csv, 0, 0) == a)
        {
            m += a;
            csv = llDeleteSubString(csv, 0, 0);
            jump continue;
        }
        if(a == "\"")
        {
            if(llList2String(s, -1) != a)
            {
                s += a;
                jump continue;
            }
            s = llDeleteSubList(s, -1, -1);
            jump continue;
        }
        m += a;
```

```
      @continue;
    } while(csv != "");
    return l + m;
}

string wasKeyValueGet(string k, string data)
{
  if(llStringLength(data) == 0) return "";
  if(llStringLength(k) == 0) return "";
  list a = llParseStringKeepNulls(data, ["&", "="], []);
  integer i = llListFindList(llList2ListStrided(a, 0, -1, 2), [ k ]);
  if(i != -1){
    string ret = llList2String(wasCSVToList(wasURLUnescape(llList2String(a, 2*i+1))), 0);
    ret = strReplace(ret, "\\r", "");
    ret = strReplace(ret, "\\n", "");
    ret = llStringTrim(ret, STRING_TRIM);

    return ret;
  }
  return "";
}

string wasKeyValueEncode(list data)
{
  list k = llList2ListStrided(data, 0, -1, 2);
  list v = llList2ListStrided(llDeleteSubList(data, 0, 0), 0, -1, 2);
  data = [];
  do
  {
    data += llList2String(k, 0) + "=" + llList2String(v, 0);
    k = llDeleteSubList(k, 0, 0);
    v = llDeleteSubList(v, 0, 0);
  } while(llGetListLength(k) != 0);
  return llDumpList2String(data, "&");
}

string wasURLEscape(string i)
{
  string o = "";
  do
  {
    string c = llGetSubString(i, 0, 0);
    i = llDeleteSubString(i, 0, 0);
    if(c == "") jump continue;
    if(c == " ")
    {
      o += "+";
      jump continue;
    }
    if(c == "\n")
    {
      o += "%0D" + llEscapeURL(c);
      jump continue;
    }
    o += llEscapeURL(c);
    @continue;
  } while(i != "");
  return o;
}

string wasURLUnescape(string i)
{
    return
llUnescapeURL( llDumpList2String( llParseStringKeepNulls( llDumpList2String( llParseStringKeepNulls( i,
["+"], [] ), " " ), ["%0D%0A"], [] ), "\n" ) );
}

default
{
    state_entry()
    {
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state ReadConfigurationNotecard
{
    state_entry()
    {
        if(llGetInventoryType("configuration") != INVENTORY_NOTECARD)
        {
            setDebug("Sorry, could not find a configuration inventory notecard.");
            return;
        }
        line = 0;
        llGetNotecardLine("configuration", line);
    }

    dataserver(key id, string data)
    {
        if(data == EOF)
        {
            if(llGetListLength(tuples) % 2 != 0)
            {
                setDebug("Error in configuration notecard.");
                return;
            }

            CORRADE = llList2Key(tuples, llListFindList(tuples, ["BOTID"])+1);

            if(CORRADE == NULL_KEY)
            {
                setDebug("Error in configuration notecard: BOT ID KEY");
                return;
            }

            GROUP = llList2String(tuples, llListFindList(tuples,["GROUP"])+1);

            if(GROUP == "")
            {
                setDebug("Error in configuration notecard: GROUP");
                return;
            }

            PASSWORD = llList2String(tuples, llListFindList(tuples, ["PASSWORD"])+1);

            if(PASSWORD == "")
            {
                setDebug("Error in configuration notecard: PASSWORD");
                return;
            }

            state url;
        }
        if(data == "") jump continue;
        integer i = llSubStringIndex(data, "#");
        if(i != -1) data = llDeleteSubString(data, i, -1);
        list o = llParseStringKeepNulls(data, ["="], []);
        string k = llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String(o,0), STRING_TRIM),
  ["\""], []), "\"");
        string v = llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String( o, 1 ), STRING_TRIM),
  ["\""], [] ), "\"");
        if(k == "" || v == "") jump continue;
        tuples += k;
        tuples += v;
        @continue;
        llGetNotecardLine("configuration", ++line);
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state url
{
    state_entry()
    {
        if ((GROUP == "BOTID") || (GROUP == "GROUP") || (GROUP == "PASSWORD")){
            llResetScript();
            return;
        }

        if ((PASSWORD == "BOTID") || (PASSWORD == "GROUP") || (PASSWORD == "PASSWORD")){
            llResetScript();
            return;
        }

        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        llRequestURL();
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(method != URL_REQUEST_GRANTED) return;
        URL = body;
        state CheckBotAvailable;
    }
    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state CheckBotAvailable
{
    state_entry()
    {
                        llSetTimerEvent(1);
    }

    timer(){
        llSetTimerEvent(10);
                        query = llRequestAgentData(ownerID, DATA_ONLINE);
    }

    dataserver(key queryid, string data)
    {
        if (query == queryid)
        {
                                if((integer)data == 1){
                                        state SendNotice;
                                }
        }
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

```
state SendNotice {
    state_entry() {
                        llInstantMessage(CORRADE,
                                wasKeyValueEncode(
                                        [
                                                "command", "notice",
                                                "group", wasURLEscape(GROUP),
                                                "password", wasURLEscape(PASSWORD),
                                                "action", "send",
                                                "subject", "test subject",
                                                "message", "test message",
                                                "callback", wasURLEscape(URL)
                                        ]
                                )
                        );

    }
    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(wasKeyValueGet("success", body) != "True") {
                                setDebug("Failed to send notice");
        }
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

Text 4 - Full script for sending a group advertisement using WAS

Our LSL code with sending data using JSON will look like this:

```
key ownerID = "YOUROWNERID"
key CORRADE;
string PASSWORD;
string GROUP;
string URL = "";
string TAG;
string val;
list lmessage;
string sjump = "";
string body;
key query;

integer line;
list tuples = [];

setDebug(string msg)
{
        llInstantMessage(ownerID, "["+ TAG +"] " + msg);
}

integer StringMatch(list lwhatsearch, string sinwhere){
   sinwhere = llToLower(sinwhere);
   sinwhere = strReplace(sinwhere, "!", "");
   sinwhere = strReplace(sinwhere, "?", "");

   list lsinwhere = llParseString2List(sinwhere,[" "],[]);

   integer imatch = llGetListLength(lwhatsearch);
   integer imatched = 0;

   integer i=0;
   for(i=0; i<imatch; i++){

      list lwords = llParseString2List(llList2String(lwhatsearch,i),["|"],[]);

      integer j=0;
      for(j=0; j<llGetListLength(lwords); j++){
         if(llListFindList(lsinwhere, [llList2String(lwords,j)]) >= 0){
            imatched++;
         }
      }
   }
   if(imatch == imatched){
      return 1;
   }else{
      return 0;
   }
}
```

```
string strReplace(string str, string search, string replace) {
   return llDumpList2String(llParseStringKeepNulls((str = "") + str, [search], []), replace);
}

list wasCSVToList(string csv)
{
   list l = [];
   list s = [];
   string m = "";
   do
   {
      string a = llGetSubString(csv, 0, 0);
      csv = llDeleteSubString(csv, 0, 0);
      if(a == ",")
      {
         if(llList2String(s, -1) != "\"")
         {
            l += m;
            m = "";
            jump continue;
         }
         m += a;
         jump continue;
      }
      if(a == "\"" && llGetSubString(csv, 0, 0) == a)
      {
         m += a;
         csv = llDeleteSubString(csv, 0, 0);
         jump continue;
      }
      if(a == "\"")
      {
         if(llList2String(s, -1) != a)
         {
            s += a;
            jump continue;
         }
         s = llDeleteSubList(s, -1, -1);
         jump continue;
      }
      m += a;
      @continue;
   } while(csv != "");
   return l + m;
}

string wasKeyValueGet(string k, string data)
{
  if(llStringLength(data) == 0) return "";
  if(llStringLength(k) == 0) return "";
  list a = llParseStringKeepNulls(data, ["&", "="], []);
  integer i = llListFindList(llList2ListStrided(a, 0, -1, 2), [ k ]);
  if(i != -1){
    string ret = llList2String(wasCSVToList(wasURLUnescape(llList2String(a, 2*i+1))), 0);
    ret = strReplace(ret, "\\r", "");
    ret = strReplace(ret, "\\n", "");
    ret = llStringTrim(ret, STRING_TRIM);

    return ret;
  }
  return "";
}

string wasKeyValueEncode(list data)
{
  list k = llList2ListStrided(data, 0, -1, 2);
  list v = llList2ListStrided(llDeleteSubList(data, 0, 0), 0, -1, 2);
  data = [];
  do
  {
    data += llList2String(k, 0) + "=" + llList2String(v, 0);
    k = llDeleteSubList(k, 0, 0);
    v = llDeleteSubList(v, 0, 0);
  } while(llGetListLength(k) != 0);
  return llDumpList2String(data, "&");
}

string wasURLEscape(string i)
{
  string o = "";
  do
  {
    string c = llGetSubString(i, 0, 0);
    i = llDeleteSubString(i, 0, 0);
    if(c == "") jump continue;
    if(c == " ")
    {
      o += "+";
      jump continue;
    }
    if(c == "\n")
    {
      o += "%0D" + llEscapeURL(c);
      jump continue;
    }
    o += llEscapeURL(c);
    @continue;
  } while(i != "");
  return o;
}

string wasURLUnescape(string i)
{
    return
llUnescapeURL( llDumpList2String( llParseStringKeepNulls( llDumpList2String( llParseStringKeepNulls( i,
["+"], [] ), " " ), ["%0D%0A"], [] ), "\n" ) );
}

default
{
    state_entry()
    {
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state ReadConfigurationNotecard
{
    state_entry()
    {
        if(llGetInventoryType("configuration") != INVENTORY_NOTECARD)
        {
            setDebug("Sorry, could not find a configuration inventory notecard.");
            return;
        }
        line = 0;
        llGetNotecardLine("configuration", line);
    }

    dataserver(key id, string data)
    {
        if(data == EOF)
        {
            if(llGetListLength(tuples) % 2 != 0)
            {
                setDebug("Error in configuration notecard.");
                return;
            }

            CORRADE = llList2Key(tuples, llListFindList(tuples, ["BOTID"])+1);

            if(CORRADE == NULL_KEY)
            {
                setDebug("Error in configuration notecard: BOT ID KEY");
                return;
            }

            GROUP = llList2String(tuples, llListFindList(tuples,["GROUP"])+1);

            if(GROUP == "")
            {
                setDebug("Error in configuration notecard: GROUP");
                return;
            }

            PASSWORD = llList2String(tuples, llListFindList(tuples, ["PASSWORD"])+1);

            if(PASSWORD == "")
            {
                setDebug("Error in configuration notecard: PASSWORD");
                return;
            }

            state url;
        }
        if(data == "") jump continue;
        integer i = llSubStringIndex(data, "#");
        if(i != -1) data = llDeleteSubString(data, i, -1);
        list o = llParseStringKeepNulls(data, ["="], []);
        string k = llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String(o,0), STRING_TRIM),
  ["\""], []), "\"");
        string v = llDumpList2String( llParseStringKeepNulls( llStringTrim( llList2String( o, 1 ), STRING_TRIM),
  ["\""], [] ), "\"");
        if(k == "" || v == "") jump continue;
        tuples += k;
        tuples += v;
        @continue;
        llGetNotecardLine("configuration", ++line);
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
```

```
            {
                llResetScript();
            }
        }

        on_rez(integer start_param)
        {
            llResetScript();
        }
    }

    state url
    {
        state_entry()
        {
            if ((GROUP == "BOTID") || (GROUP == "GROUP") || (GROUP == "PASSWORD")){
                llResetScript();
                return;
            }

            if ((PASSWORD == "BOTID") || (PASSWORD == "GROUP") || (PASSWORD == "PASSWORD")){
                llResetScript();
                return;
            }

            llSetTimerEvent(0.5);
        }

        timer(){
            llSetTimerEvent(60);
            llRequestURL();
        }

        http_request(key id, string method, string body)
        {
            llHTTPResponse(id, 200, "OK");
            if(method != URL_REQUEST_GRANTED) return;
            URL = body;
            state CheckBotAvailable;
        }

        changed(integer change)
        {
            if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
            {
                llResetScript();
            }
        }
        on_rez(integer start_param)
        {
            llResetScript();
        }
    }

    state CheckBotAvailable
    {
        state_entry()
        {
                        llSetTimerEvent(1);
        }

        timer(){
            llSetTimerEvent(10);
                        query = llRequestAgentData(ownerID, DATA_ONLINE);
        }

        dataserver(key queryid, string data)
        {
            if (query == queryid)
            {
                                if((integer)data == 1){
                                        state SendNotice;
                                }
            }
        }

        changed(integer change)
        {
            if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
            {
                llResetScript();
            }
        }

        on_rez(integer start_param)
        {
            llResetScript();
        }
    }

    state SendNotice {
        state_entry() {
                        llInstantMessage(CORRADE,
                                wasKeyValueEncode(
                                        [
                                                "command", "notice",
                                                "group", wasURLEscape(GROUP),
                                                      "password", wasURLEscape(PASSWORD),
                                                      "action", "send",
                                                      "subject", "test subject",
                                                      "message", "test message",
                                                      "callback", wasURLEscape(URL)
                                             ]
                                  )
                        );

        }
        http_request(key id, string method, string body)
        {
            llHTTPResponse(id, 200, "OK");
            if(wasKeyValueGet("success", body) != "True") {
                                  setDebug("Failed to send notice");
            }
        }

        changed(integer change)
        {
            if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
            {
                llResetScript();
            }
        }

        on_rez(integer start_param)
        {
            llResetScript();
        }
    }
```

Text 5 - Full script for sending the group advertisement using JSON

However, in PHP, if you use my shipping function, the code will be shorter (it doesn't matter if the shipping is using WAS or JSON):

```
$params = array(
	"command" => "notice",
	"group" => BOT_GROUP,
	"password" => BOT_PASSWORD,
	"subject" => "test subject",
	"message" => "test message",
);
$ret = SendToBot($params);
print_r($ret);
```

Text 6 - Sending a group advertisement in PHP

I will now discuss both codes.

key ownerID = "YOUROWNERID" - here instead of YOUROWNERID we insert our avatar UUID, which we can easily get from Firestorm Viewer.

state CheckBotAvailable - state, where we check whether the bot is available in Second Life at all, if it is not, it simply waits for it to appear and then executes further instructions. This check is recommended if the script prim is not part of the clothes and is standing freely somewhere on a plot in Second Life. The check is performed every 10 seconds by the timer() function.

state SendNotice - our state where we post the announcement. We formulated our command with additional parameters in the form of an LSL list [], where each parameter has its own value.

You may notice that some are sent to the wasURLEscape() function which encodes special characters into a format accepted by Second Life [see FAQ].

The entire list is encoded with the wasKeyValueEncode() function, which converts parameter1=value1&parameter2=value2&... to the URL and is sent to the bot using the llInstantMessage() command.

When sending with JSON, it is much easier, because we only create a list where the values are not covered by any functions, which is first encoded into URL form, and then into JSON form in object-oriented form.

http_request(key id, string method, string body) - this function expects and returns data from the bot, because we previously defined a callback (i.e. feedback) in the parameters, which will be set to the URL given by Second Life. This is where we process further. At this point I chose to check if the command was correctly executed.

changed(integer change) - detects changes related to the change of the owner, region or restart of the region here and takes appropriate actions, for me it is simply restarting the script.

on_rez(integer start_param) - performs an action related to the object's reservation, for me, restart the script.

Remember to check your code if it sends the command with arguments correctly and returns the expected data.

One more topic needs to be discussed here.
A parameter value usually has one item, what if it has more than one list item? The wasListToCSV() function comes in handy, as it converts the list to a comma separated value.

Text 7 - Convert the list to CSV

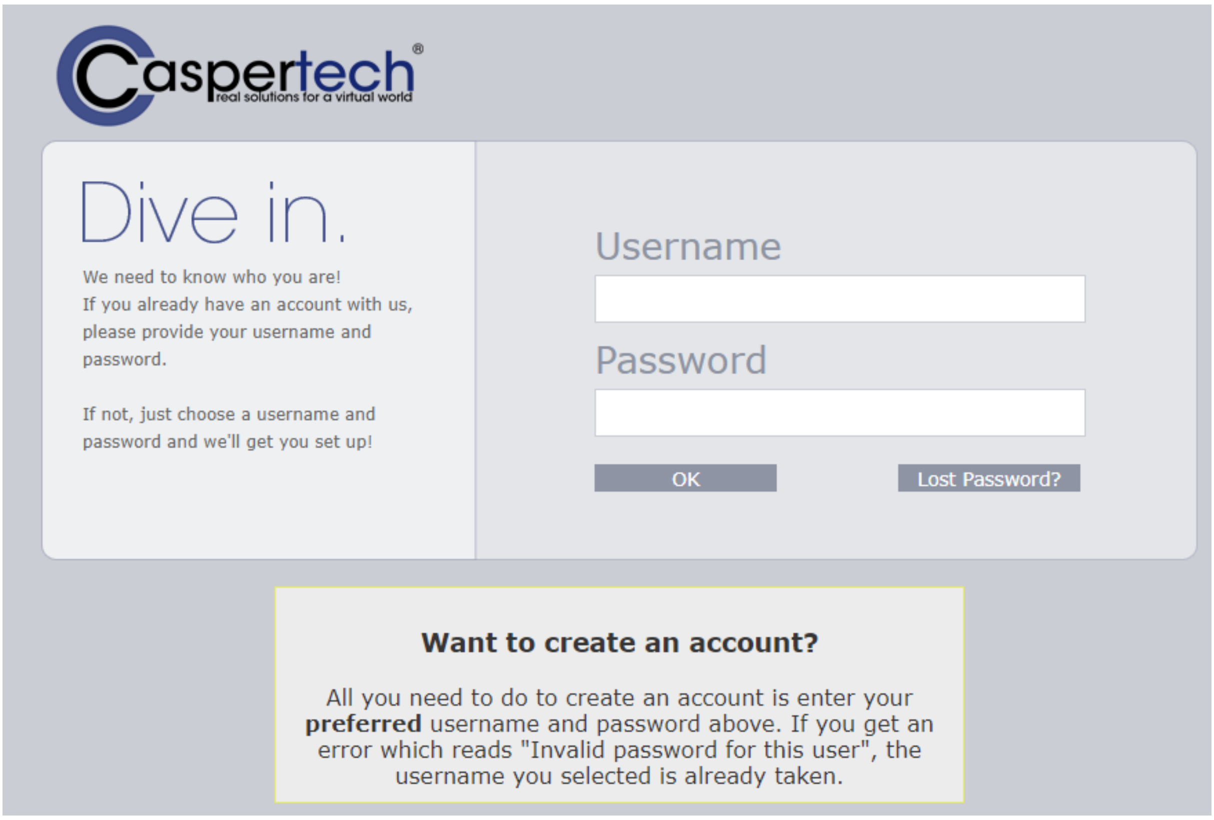
However, if we have data returned from the bot in CSV format, we can convert it to a list using wasCSVToList().
If you use PHP, the equivalents of these functions will be: wasArrayToCSV() [we give elements of type array()] and wasCSVToArray().
What about encoding in JSON? In the case of PHP, we don't have to create fancy functions here, because PHP already has a built - in function to encode data in JSON, and if you use my function, you don't have to worry about it at all.
In LSL, when encoding JSON, we also replace the value array with the wasListToCSV() function, but we do not use the wasKeyValueEncode() function anywhere.
Then we encode everything with wasKeyValueEncode() followed by wasKeyValueToJSON(). This way we have a ready request to the bot in JSON format.
In short, it might look like this:

```
llInstantMessage(CORRADE, wasKeyValueToJSON(wasKeyValueEncode(lprepare), JSON_OBJECT));
```

4.15. Corrade integration with CasperTech[19]

CasperTech enables simple, automated receipt of money for renting l and or selling products. When renting the l and with the use of the CasperTech system, the system will renew the lease itself after payment by the tenant.

1. Log in to the CasperTech website - https://let.casperdns.com/.


Dive in.

We need to know who you are!
If you already have an account with us, please provide your username and password.

If not, just choose a username and password and we'll get you set up!

Username

Password

OK   Lost Password?

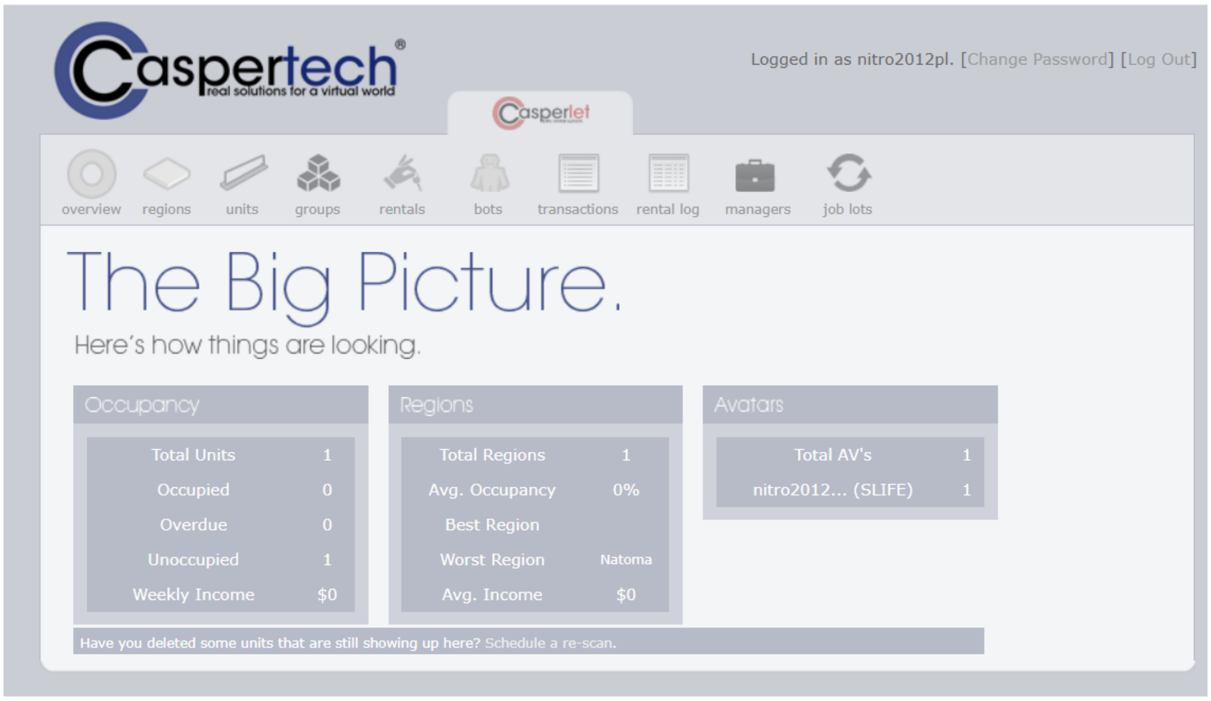
Want to create an account?

All you need to do to create an account is enter your **preferred** username and password above. If you get an error which reads "Invalid password for this user", the username you selected is already taken.

Picture 4.27 - CasperTech login page

2. Then select 'Bots' from the menu.
3. Click on the 'Add Bot' button on the screen.

19 Based: https://wiki.casperdns.com/index.php/Corrade


Caspertech real solutions for a virtual world
Logged in as nitro2012pl. [Change Password] [Log Out]
Casperlet
overview
regions
units
groups
rentals
bots
transactions
rental log
managers
job lots

The Big Picture.

Here's how things are looking.

Occupancy	
Total Units	1
Occupied	0
Overdue	0
Unoccupied	1
Weekly Income	$0

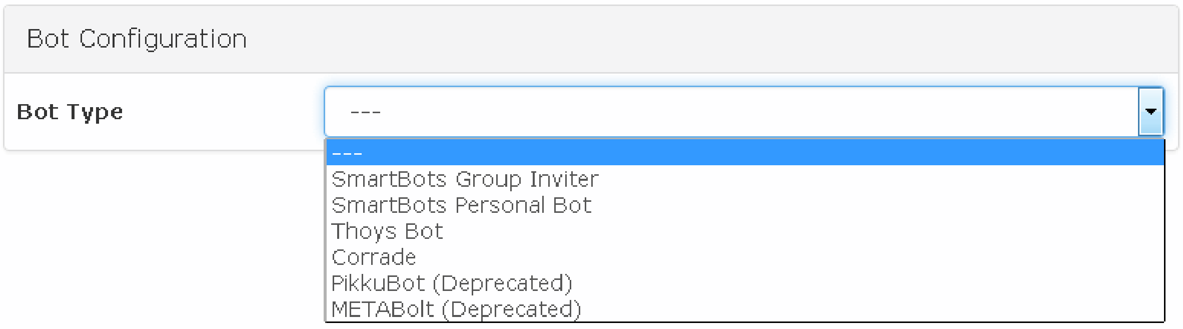
Regions	
Total Regions	1
Avg. Occupancy	0%
Best Region	
Worst Region	Natoma
Avg. Income	$0

Avatars	
Total AV's	1
nitro2012... (SLIFE)	1

Have you deleted some units that are still showing up here? Schedule a re-scan.

Picture 4.28 - CasperTech website after logging in

4. Select 'Corrade' from the drop - down list.

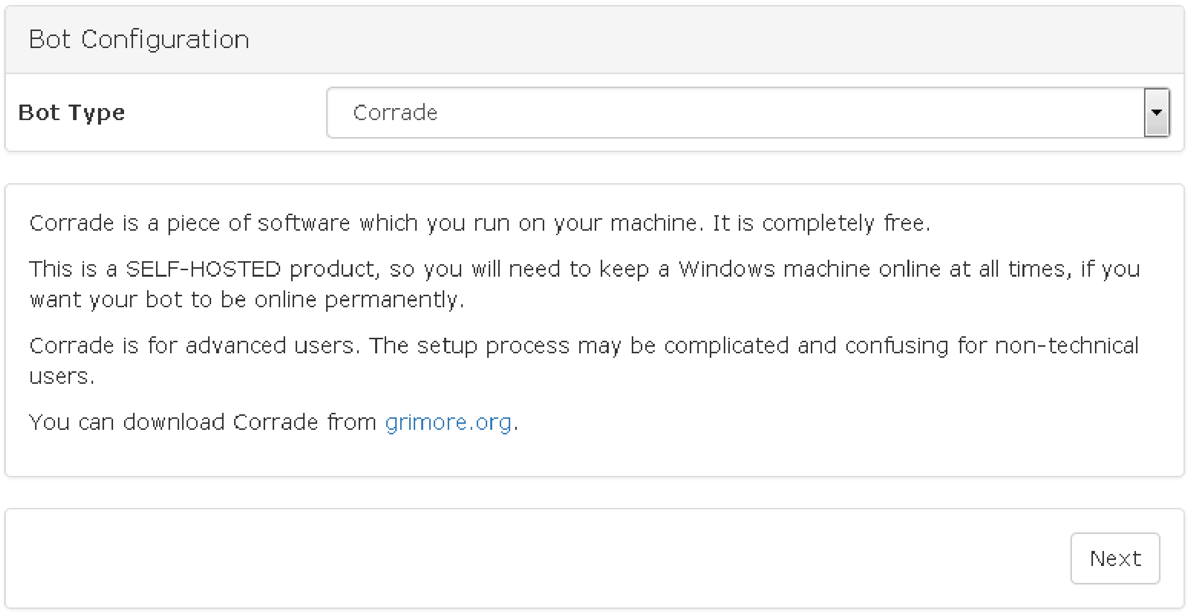

Bot Configuration

Bot Type: ---

SmartBots Group Inviter
SmartBots Personal Bot
Thoys Bot
Corrade
PikkuBot (Deprecated)
METABolt (Deprecated)

Picture 4.29 - Supported bot types by CasperTech

5. On the screen you will see a short information about Corrade.
6. If everything is correct, click on the 'Next' button.


Bot Configuration

Bot Type: Corrade

Corrade is a piece of software which you run on your machine. It is completely free.

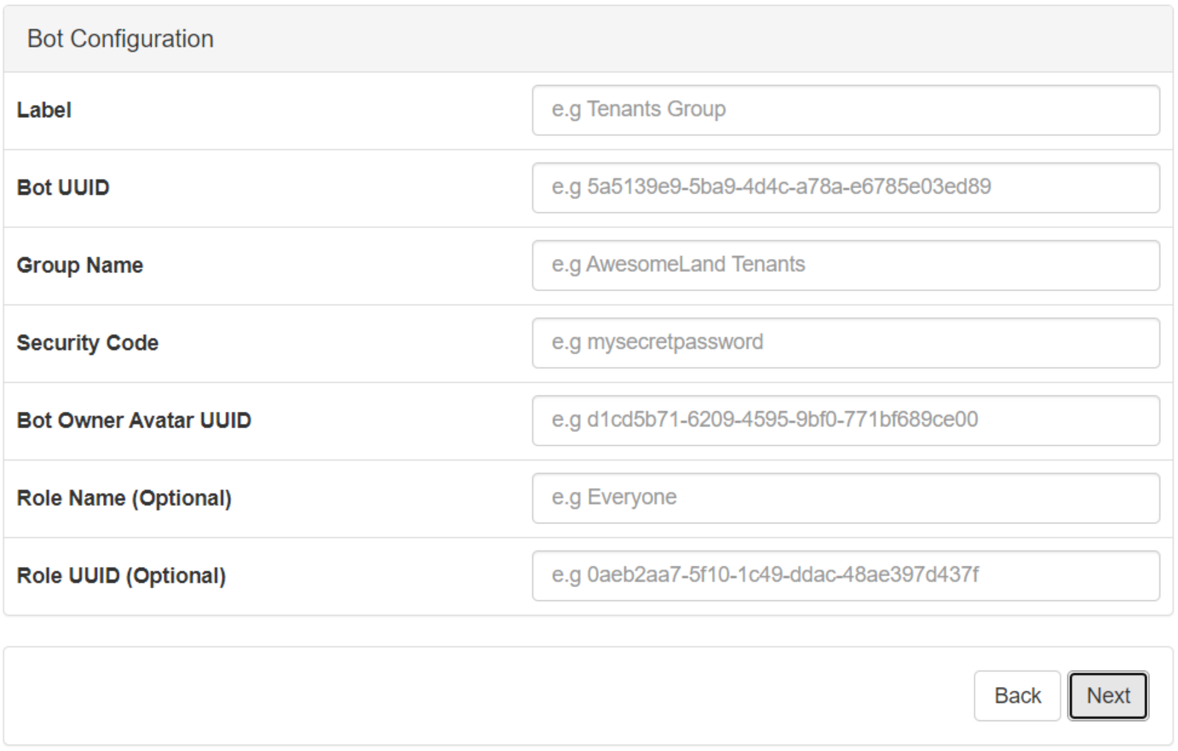
This is a SELF-HOSTED product, so you will need to keep a Windows machine online at all times, if you want your bot to be online permanently.

Corrade is for advanced users. The setup process may be complicated and confusing for non-technical users.

You can download Corrade from grimore.org.

Picture 4.30 - Description of choosing a bot as Corrade

7. On the next screen, complete the necessary data:
   - Label - name displayed on the list on the CasperTech website,
   - Bot UUID - bot's UUID,
   - Group name - name of the group to which the bot is assigned,
   - Secure Code - password for the group, which was previously set and entered into the bot configuration file,
   - Bot Owner Avatar UUID - UUID of your avatar as the owner of the bot,
   - Role Name - additional option; role name,
   - Role UUID - additional option; UUID of the role.

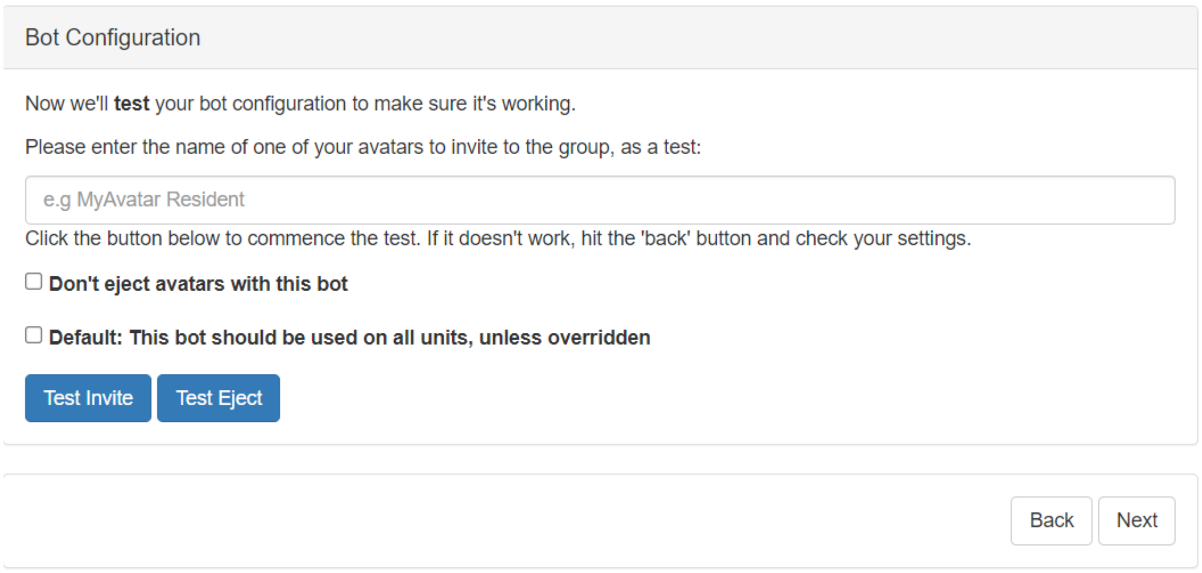


Bot Configuration

Label	e.g Tenants Group
Bot UUID	e.g 5a5139e9-5ba9-4d4c-a78a-e6785e03ed89
Group Name	e.g AwesomeLand Tenants
Security Code	e.g mysecretpassword
Bot Owner Avatar UUID	e.g d1cd5b71-6209-4595-9bf0-771bf689ce00
Role Name (Optional)	e.g Everyone
Role UUID (Optional)	e.g 0aeb2aa7-5f10-1c49-ddac-48ae397d437f

Picture 4.31 - Bot configuration screen

8. On the next screen, you have the option to test your bot configuration.
9. Enter the name of the avatar to which the bot should send invitations to the group or remove from the group.
10. We can also skip this step.
11. We give the name of the avatar that will be tested by the bot.
12. We have the option of selecting additional options:
    - Don't eject avatars with this bot - don't throw avatars with this bot,
    - This bot should be used on all units, unless overridden - whether the bot should serve all lease units, unless other settings override this option.
13. When we click on 'Test invite', the website will send a command to the bot, which will invite the avatar selected earlier to the group.
14. When we click on 'Test eject' the website will send a command to the bot, which will kick the avatar out of the group.
15. If everything is ok then click on the 'Next' button.


Bot Configuration

Now we'll **test** your bot configuration to make sure it's working.

Please enter the name of one of your avatars to invite to the group, as a test:

e.g MyAvatar Resident

Click the button below to commence the test. If it doesn't work, hit the 'back' button and check your settings.

☐ **Don't eject avatars with this bot**

☐ **Default: This bot should be used on all units, unless overridden**

Picture 4.32 - Screen with the possibility of testing the bot

16. If everything is fine, click on OK or next, which will save the configuration of our bot and return to the list of bots.

4.16. Data sieving

The 2 KB limit imposed by Second Life is used when receiving data from the http_request() bot in the LSL. Above this limit, the data is truncated. To get the complete data, there are two options: either move the command call to PHP or use the so-called data screening. Data screening allows you to select the data that interests us and not to cut the data imposed by the above - mentioned limit.

We have the following sifter at our disposal:

- count - gives the number of items that match the previously given regular expression

Input data: a,1,b,2,c,3,d,4,e,5,f,6,g,7,h,8,i,9

```
"sift", wasListToCSV([
    "count", "[0-9]"
])
```

Output data: 9

Comment: After processing the data, Corrade selects as many elements from the list that are numbers, that is: 1,2,3,4,5,6,7,8,9, so the length of this list is 9 elements.

- distinct - only returns unique items in the list

Input data: a,b,c,a,b,c,a,b,c,d,a,b,d,a,e,a,h,a

```
"sift", wasListToCSV([
    "distinct", ""
])
```

Output data: a,b,c,d,e,h

- each - returns every specified item in the list

Input data: a,1,b,2,c,3,d,4,e,5,f,6,g,7,h,8,i,9

```
"sift", wasListToCSV([
    "each", 2
])
```

Output data: a,b,c,d,e,f,g,h,i

Comment: Returns every other item in the list in the example.

- html - returns elements that match xpath HTML
- jsonpath - returns items that match a JSON path
- match - returns elements or parts of elements that match the regular expression

Input data: Item Cost 100,2,Item Cost 500,3,Item Cost 10,4,Item Cost 15

```
"sift", wasListToCSV([
    "match", "Item Cost ([0-9]+)"
])
```

Output data: 100,500,10,15

- md5 - generates an MD5 hash for the output text
- sha1 - generates a SHA1 hash for the output text
- permute - shows the list items on the basis of a given rotation

- random - returns a random item in the list
- reverse - returns the list of items in reverse order
- select - selects values based on a predefined key

Input data: Price,10,Item,Book,Price,50,Item,Chair,Price,200,Item,PC

```
"sift", wasListToCSV([
    "select", "Price"
]),
```

Output data: 10,50,200

- skip - skips the specified number of leading items from the list
- take - gets the specified number of items from the list
- xpath - returns items based on an XPath

The above data sifters can be combined with each other.

4.17. Weak and strong reference by object name and UUID

In Second Life, everything has an ID, the so-called UUID (Universally unique identifier).

In Corrade, when we refer to objects, we can refer to them by their name or UUID. However, calling an object by name is not recommended as there may be more than one object in the same region with the same name as the searched object. Instead of operating on only one object, we will operate on all found. We call it a poor reference. It's a much better idea to refer to an object by its UUID, which is unique to each object in the region. We call it a strong reference.

4.18. Confirmation of receiving data from the bot in the LSL script.

Whenever you get data from the bot in the LSL script, you should confirm to the bot that the data was received without problems.

Example:

```
http_request(key id, string method, string body){
  llHTTPResponse(id, 200, "OK"); //confirm
  //.....
}
```

If this provision is omitted, Corrade waits for a return confirmation for approximately one minute. This one line shortens this time and Corrade may close one link sooner.

4.19. Bot rack size.

You should log in as an avatar of your bot from time to time and inspect his wardrobe. You should delete the things he doesn't use, because on the one h and they clutter his wardrobe and on the other h and they make Corradde need more time to cache them.

4.20. Corrade protection and defense against attacks

1. Is the configuration file properly secured? (e.g. relevant file permissions, access only to relevant users)
2. Does the bot have access to the appropriate groups?
3. Does the bot have access to the appropriate permissions and notifications in groups?
4. Does the computer it is running on connects safely to the Internet and trusted servers?
5. Have you set resource cap to Corrade?
6. Do you use llRequestSecureURL() instead of llRequestURL() [MITM protection] in your LSL scripts?
7. Is your computer protected against DDoS attacks?

8. Do only trusted people have access to the computer?
9. Does the bot folder have the appropriate permissions?
10. Have you correctly defined the command to execute when using the 'execute' command? Will unexpected changes to the system occur, e.g. deletion of important files?
11. Are you using llSensor and llSensorRepeat to get the UUID of objects?

Notes

- MD5 or SHA1 are weak passwords for generating a password, if you suspect that the bot's account has been hacked - change the password immediately to a stronger one, which you re-enter in the configuration file. Review the above list to see if you have done everything to avoid hacking your account.
- llRequestSecureURL() obtains secure URLs from Second Life (HTTPS), however Second Life uses self-signed SSL certificates, which makes it necessary to disable certificate issuer checking. If the secure URL from Second Life causes a problem (e.g. for notifications) use llRequestURL().

4.21. Comments

1. Commands such as getavatarpositions, getmapavatarpositions return the wrong Z coordinate - the maximum number is 1000, so they are not recommended when getting bot coordinates or other avatars. If you want the exact coordinates use the tracker notification or the LSL llGetObjectDetails command.

5. Leonardo De ArtBot

5.1. Computer parameters ('bot house')

Picture 5.1 - Fujitsu ESPRIMO e5730 - front

Picture 5.2 - Fujitsu ESPRIMO e5730 - back

Leonardo is run on an SFF computer, which is closed in a small handheld case. The computer was manufactured around 2008, but it works fine.
The computer parameters are:

- operating system: Windows 7 SP1 64-bit,
- CPU: Intel Core 2 Duo E8600 @ 3.33 GHz (2 cores, 2 threads),
- memory: 4 GB RAM DDR2 - SDRAM, 400 MHz (PC2 - 6400) clock speed; only 3.65 GB available in the system,
- graphics card: integrated Intel Q45/Q43,
- hard drive: Hitachi 160 GB HDD, SATA, disk speed - 7200 rpm,
- network:
  - (main) LAN; Intel 82567LF-3, speed: 100 Mbps, connected to a WiFi router with mobile Internet (Orange PL network),
  - (spare) mobile modem: Huawei E3372 HiLink, max. 150 Mbps; Orange PL network,
- software:
  - Corrade, version: 11.0.88.76 (publication date: 2021 - 01 - 06 07:12),
  - Web server (XAMPP software):
    - Apache 2.4.41,
    - database: MariaDB 10.4.11,
    - phpMyAdmin 4.9.2,
    - OpenSSL 1.1.0g,
    - PHP 7.4.1 (VC15 x64, thread safe)

5.2. Bot brain structure and LSL line communication - external script

Leonard's bot brain was built around PHP and LSL.

I chose PHP for the following reasons: I know the language very well, you write code in this language very quickly, it allows you to limit the maximum amount of memory you can use.

The bot folder itself is currently around 2.22 GB in size, which consists of Corrade files and folders, log files, bot brain files.

This folder consists of subfolders and subfiles (Picture 5.4 - The structure of folders and files in the Leonardo bot):

- Other files and directories related to Corrade,
- Cache - saves Second Life data to the cache (e.g. resources, avatar data, etc.),
- Contrib (linux, windows, macosx) - contains instructions and programs on how to install Corrade as a service in the system,
- Logs - program logs and private conversations between the bot and avatars saved to files, public conversations from a public chat, group chat, etc.,
- mcorrade.ps1 + other files - my Powershell script to manage the bot along with other files,
- Nucleus - Nucleus files,
- State - saves notification files with assigned URLs from Second Life,
- php - bot brain written in PHP
  - botconfig.ini - bot INI configuration file,
  - cache.ini - cache file,
  - composer.phar - a file for managing the Composer project dependencies,
  - config.php - PHP configuration file,
  - cron.bat - cron batch program,
  - corrademonitorservice.php - PHP file that allows you to monitor the bot process,
  - functions.php - PHP functions,
  - inPolygon.class.php - PHP class that checks whether a given point belongs to a polygon,
  - PHP files starting with part* - files that perform various functions by the bot; are connected via the main servebot.php file,
  - phplint.bat - batch program that checks the correctness of the syntax in the folder with the bot brain,
  - servebot.php - main PHP file that performs bot functions; connects other parts saved in PHP files; is invoked by a bot from Second Life,
  - logs - folder with files with saved events by serverbot.php,

  - lsl - LSL files,
  - vendor - Composer dependencies folder.

If we estimate what percentage is occupied by PHP scripts and what percentage of LSL scripts in the form of the number of lines of code, then after calculating the ratio is approx. 53 – 57% / 47 - 43%, which is quite a small difference.

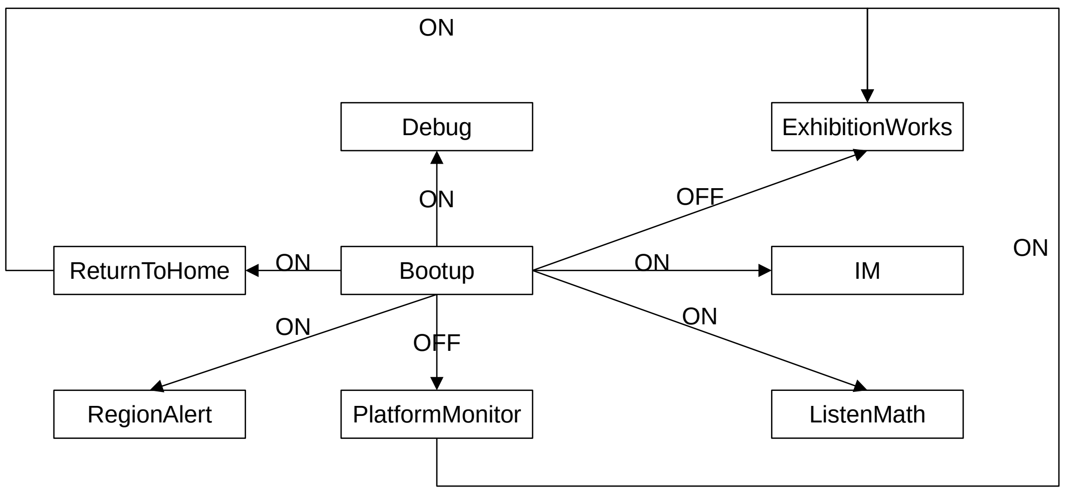
LSL works as standard on the Second Life side, where it makes PHP requests (execute the serverbot.php file) via the 'execute' command from the Corrade API, which are executed on the computer where the bot is running.

The bot's runtime is in one prim called ArtBotStartupV1, which is worn as an invisible garment by the bot. The bot owns this prim, and any changes to this prim require you to manually log in as a bot and edit that prim. I do not have a problem with this here, because I rarely log in as a bot, and even if it does, it is not bothersome for me. However, this is a plus, because the bot has everything with it. Avatar scripts will not run if this one is on a plot where the option runs avatar scripts, therefore, in the system's task schedule, a script is run periodically that checks if the bot is on the right plot, and if not, it tries to teleport it.

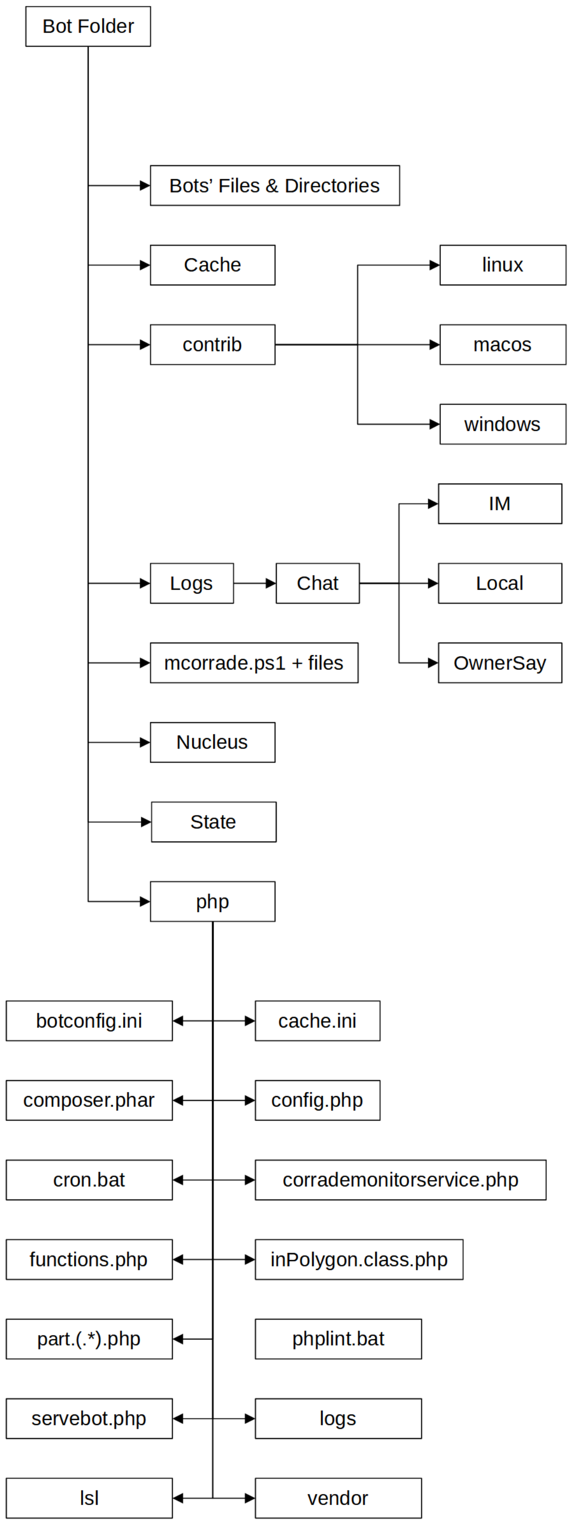
Prima includes the following scripts:

- **Bootup** - is the first script to decide which scripts to run first (Picture 5.3 - The dependency of running individual scripts in prim); it is triggered as soon as the bot logs into Second Life or reconnects prim as part of the clothes; the bot author decides the order in which the scripts are run,
- **Debug** - a script that tries to restart another script that reported an error (usually of the type: 'script not found'),
- **ExhibitionWorks** - a script that handles the textures received from the avatars and places them on a joint exhibition (it is quite close to the bot stand),
- **IM** - a script that handles private messages from avatars,
- **ListenMath** - a script that handles some math functions,
- **PlatformMonitor** - a script that monitors the bot's workstation - if there is no object present - creates or resizes it from the bot's cabinet,

- **RegionAlert** - a script that monitors whether a region restart has been ordered and in this case sends group information to Facebook and Twitter,
- **ReturnToHome** - script that checks if the bot is on the right plot.



Picture 5.3 - The dependency of running individual scripts in prim



Picture 5.4 - The structure of folders and files in the Leonardo bot

Regardless of the script or its functions, when the data is processed, the input data is forwarded to external PHP files that are in the bot folder, and a bot in Second Life can receive the status of this processing. Dispatch is done with the execute command with parameters: group name; group password; path of the file to be executed; parameters passed to the executable file; the name of the LSL script from which the call comes for possible further recognition in the logs.

An example function to send to a PHP file looks like this:

```
sendPHP(list value){
  lprepare = [
      "group", GROUP,
      "password", PASSWORD,
      "command", "execute",
      "file", "php",
      "parameter", "php/servebot.php " + llDumpList2String(value, "|"),
      "_script", TAG
  ];

  llOwnerSay(wasKeyValueToJSON(wasKeyValueEncode(lprepare), JSON_OBJECT));
}
```

Text 8 - function to send data to an external PHP file

The PHP runtime file is entered into the global PATH system path on Windows.

The php/servebot.php file is always called and handles all PHP requests from the bot.

llDumpList2String is an LSL function that converts a list to a character string, separating the list entries with an appropriate delimiter (here '|' - with a vertical bar).

_script is an additional parameter that will not be processed by the bot software, however, when analyzing the logs it allows to detect which LSL script triggered the command. Its value is stored by the TAG variable, which holds the name of the script that is assigned when the default() {} state is called through the llGetScriptName() function.

Everything is sent directly to the bot by the llOwnerSay command, which does not delay the forwarding of messages or ignore messages due to the bandwidth of other similar functions. Everything is coded to JSON before sending.

```
default
{
    state_entry()
    {
        TAG = llGetScriptName();
        setDebug("Free memory: " + (string)llGetFreeMemory());
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

Text 9 - sample default() {} state in LSL script

An example use somewhere in LSL could look like this:

```
sendPHP(["action", "ACTIONAME", "param2", varname2]);
```

where the following values of the list are:

- action - ACTIONNAME = action name which must match the action name in the servebot.php file,
- param2 - varname2 = parameter value (there can be many parameters).

The list must have an even number of items.

Since we are sending parameters to the console version of the executable program without the graphical user interface, remember that the maximum length of the string passed as a parameter to the executable file must not exceed 2047 characters[20].

5.3. Bot operation analysis

In the period 2020-10-28 - 2020-11-23, information about the bot was collected using the built - in 'heartbeat' notification, which sent data to an external script, which was finally saved to the database. Since the 'heartbeat' notification returns results every 1 second, approximately 1.7 million rows were received in the database (Picture 5.5 - Table where the data from the 'heartbeat' notification was saved). I decided to leave only those lines that were written every full hour and minute - as a result, out of 1.7 million lines there were about 31,000 (Picture 5.6 - Slimmed Table with saved data from the 'heartbeat' notification). I generated charts and tables below for the 3 searched data: average CPU usage, average RAM usage, and the number of threads running.

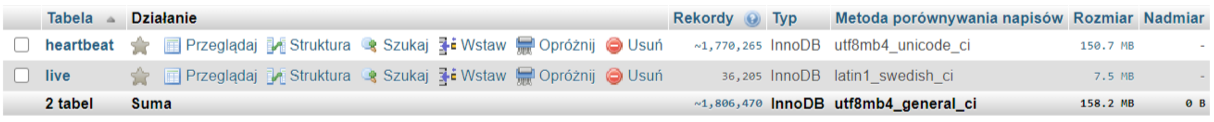


	Tabela	Działanie	Rekordy	Typ	Metoda porównywania napisów	Rozmiar	Nadmiar
☐	heartbeat	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	~1,770,265	InnoDB	utf8mb4_unicode_ci	150.7 MB	-
☐	live	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	36,205	InnoDB	latin1_swedish_ci	7.5 MB	-
	2 tabel	Suma	~1,806,470	InnoDB	utf8mb4_general_ci	158.2 MB	0 B

Picture 5.5 - Table where the data from the 'heartbeat' notification was saved

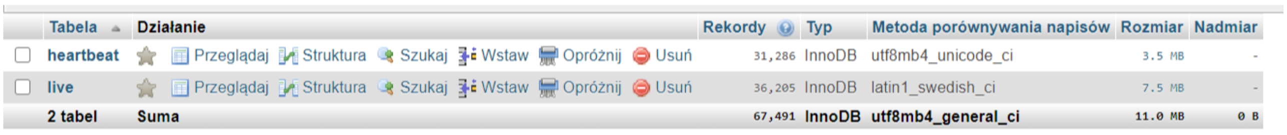
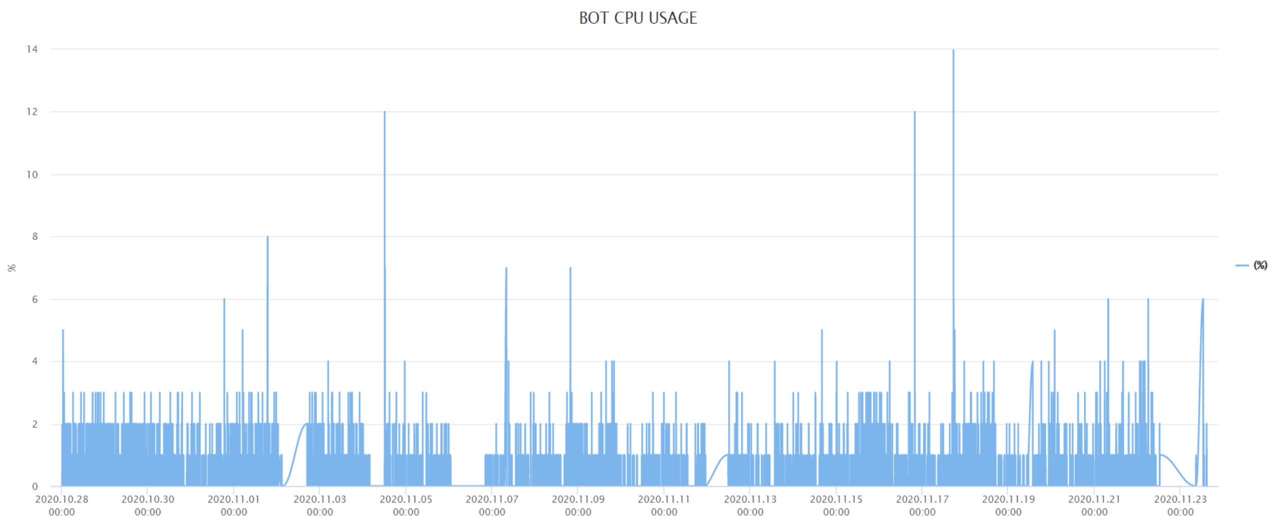


	Tabela	Działanie	Rekordy	Typ	Metoda porównywania napisów	Rozmiar	Nadmiar
☐	heartbeat	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	31,286	InnoDB	utf8mb4_unicode_ci	3.5 MB	-
☐	live	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	36,205	InnoDB	latin1_swedish_ci	7.5 MB	-
	2 tabel	Suma	67,491	InnoDB	utf8mb4_general_ci	11.0 MB	0 B

Picture 5.6 - Slimmed Table with saved data from the 'heartbeat' notification

20 https://docs.microsoft.com/en - us/troubleshoot/windows - client/shell - experience/command - line - string - limitation

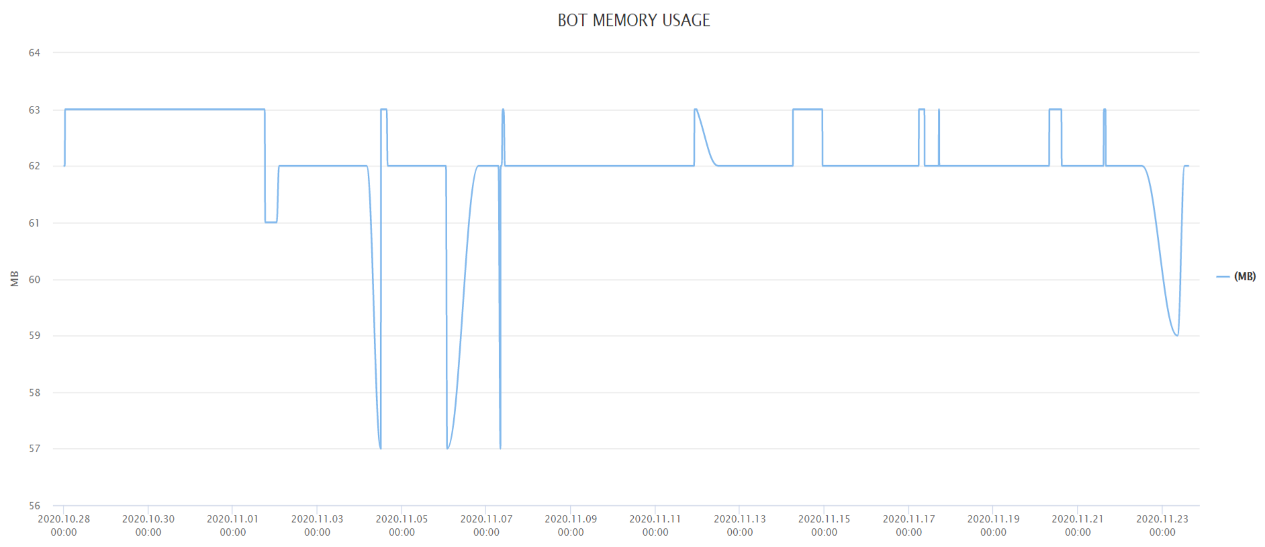


Picture 5.7 - Graph of CPU usage by bot

	%
MIN	0
MAX	14
AVG	0.15

Table 3 - Table with short CPU usage statistics

Based on the analysis of the graph and the table, it can be concluded that the CPU consumption by the Corrade program is very low - at its peak it reached the value of 14%.

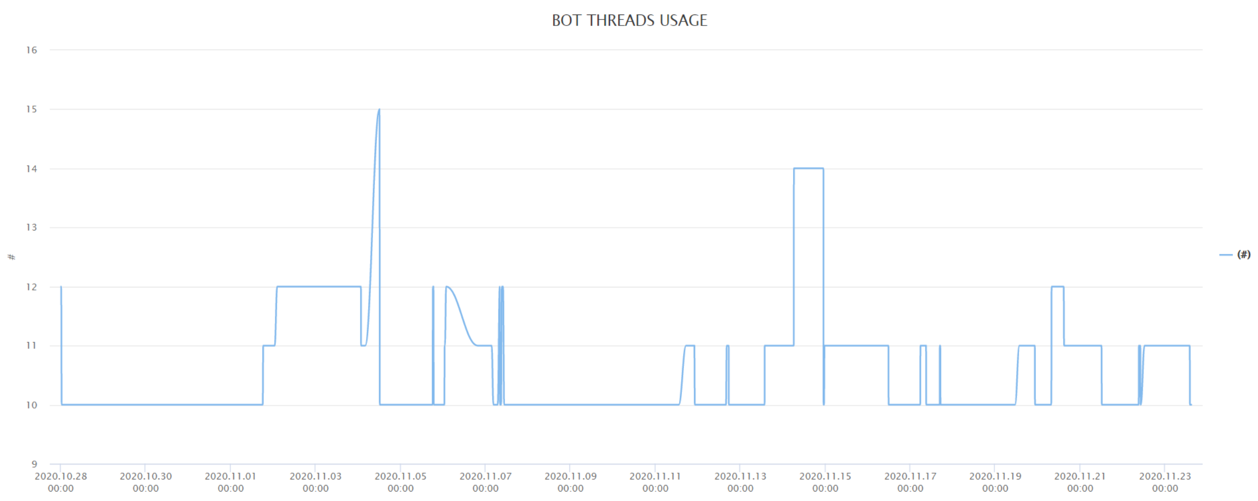


Picture 5.8 - Graph of RAM usage by the bot

	%
MIN	57
MAX	63
AVG	62

Table 4 - Table with a brief memory statistic

Based on the analysis of the graph and the table, it can be concluded that the RAM memory consumption by the program is in the range of 57 - 63 MB. The computer on which the bot was running has 4 GB of RAM, which means that the program uses memory in the range 1.39% - 1.53%.



Picture 5.9 - Corrade thread usage graph

	%
MIN	10
MAX	15
AVG	10

Table 5 - A table with a short usage statistic of the number of threads

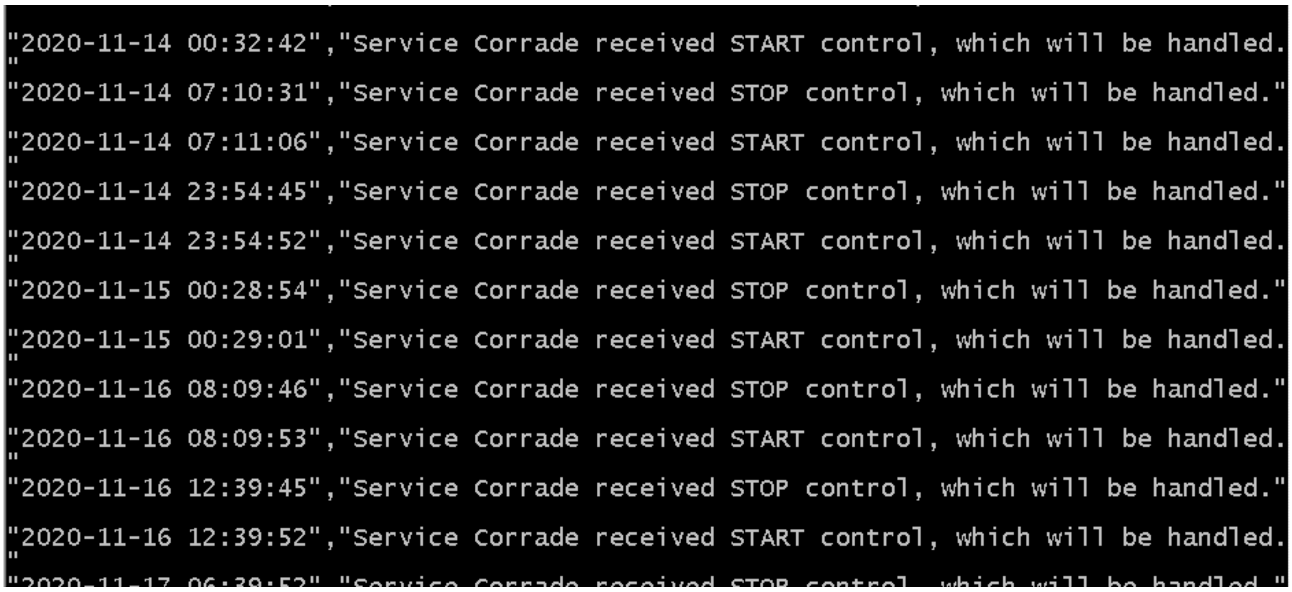
Based on the analysis of the graph and the table, it can be concluded that the Corrade program uses on average 10 threads to handle the bot.

The availability of the bot will be analyzed next. Availability was tested in the period 2020-04-15 - 2020-12-25. The data was retrieved from the Windows event log using powershell, which retrieves events logged by NSSM (the one from registering the program as a service), which then logs events with the status 'START control' or 'STOP control'.

```
powershell -Command "Get-WinEvent -FilterHashtable @{LogName=\"Application\";
ProviderName=\"nssm\";StartTime=\"2020-04-14 00:00:00\";EndTime=\"2020-12-31
00:00:00\"} | Sort-Object -Property TimeCreated | Where-Object -Property Message -Match
'START control|STOP control'| Select-Object -Property TimeCreated,Message | ConvertTo-
Csv"
```

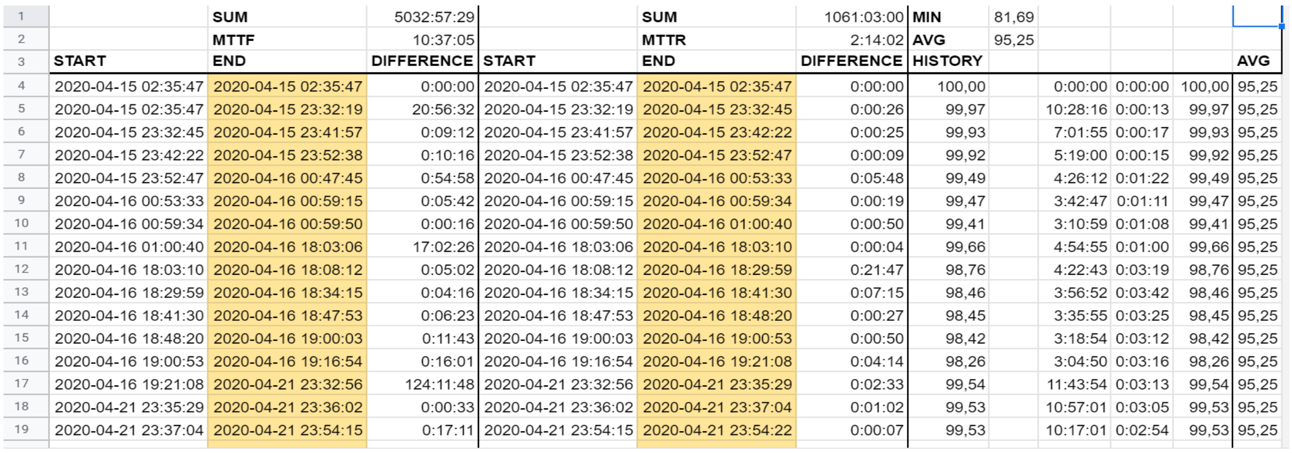
The result is a CSV result, where we have the time of registering the event and the message. Everything now depends on us how we process this data (Picture 5.10 - An example of getting information about starting the NSSM service).

I entered the data manually into the Google spreadsheet, where I automatically calculated the differences between the dates, totals and availability of the bot (Picture 5.11 - My example sheet for entering bot startup and crash times).



```
"2020-11-14 00:32:42","Service Corrade received START control, which will be handled.
"
"2020-11-14 07:10:31","Service Corrade received STOP control, which will be handled."
"2020-11-14 07:11:06","Service Corrade received START control, which will be handled.
"
"2020-11-14 23:54:45","Service Corrade received STOP control, which will be handled."
"2020-11-14 23:54:52","Service Corrade received START control, which will be handled.
"
"2020-11-15 00:28:54","Service Corrade received STOP control, which will be handled."
"2020-11-15 00:29:01","Service Corrade received START control, which will be handled.
"
"2020-11-16 08:09:46","Service Corrade received STOP control, which will be handled."
"2020-11-16 08:09:53","Service Corrade received START control, which will be handled.
"
"2020-11-16 12:39:45","Service Corrade received STOP control, which will be handled."
"2020-11-16 12:39:52","Service Corrade received START control, which will be handled.
"
"2020-11-17 06:39:53","Service Corrade received STOP control, which will be handled."
```

Picture 5.10 - An example of getting information about starting the NSSM service



1		SUM	5032:57:29		SUM	1061:03:00	MIN	81,69				
2		MTTF	10:37:05		MTTR	2:14:02	AVG	95,25				
3	START	END	DIFFERENCE	START	END	DIFFERENCE	HISTORY					AVG
4	2020-04-15 02:35:47	2020-04-15 02:35:47	0:00:00	2020-04-15 02:35:47	2020-04-15 02:35:47	0:00:00	100,00		0:00:00	0:00:00	100,00	95,25
5	2020-04-15 02:35:47	2020-04-15 23:32:19	20:56:32	2020-04-15 23:32:19	2020-04-15 23:32:45	0:00:26	99,97		10:28:16	0:00:13	99,97	95,25
6	2020-04-15 23:32:45	2020-04-15 23:41:57	0:09:12	2020-04-15 23:41:57	2020-04-15 23:42:22	0:00:25	99,93		7:01:55	0:00:17	99,93	95,25
7	2020-04-15 23:42:22	2020-04-15 23:52:38	0:10:16	2020-04-15 23:52:38	2020-04-15 23:52:47	0:00:09	99,92		5:19:00	0:00:15	99,92	95,25
8	2020-04-15 23:52:47	2020-04-16 00:47:45	0:54:58	2020-04-16 00:47:45	2020-04-16 00:53:33	0:05:48	99,49		4:26:12	0:01:22	99,49	95,25
9	2020-04-16 00:53:33	2020-04-16 00:59:15	0:05:42	2020-04-16 00:59:15	2020-04-16 00:59:34	0:00:19	99,47		3:42:47	0:01:11	99,47	95,25
10	2020-04-16 00:59:34	2020-04-16 00:59:50	0:00:16	2020-04-16 00:59:50	2020-04-16 01:00:40	0:00:50	99,41		3:10:59	0:01:08	99,41	95,25
11	2020-04-16 01:00:40	2020-04-16 18:03:06	17:02:26	2020-04-16 18:03:06	2020-04-16 18:03:10	0:00:04	99,66		4:54:55	0:01:00	99,66	95,25
12	2020-04-16 18:03:10	2020-04-16 18:08:12	0:05:02	2020-04-16 18:08:12	2020-04-16 18:29:59	0:21:47	98,76		4:22:43	0:03:19	98,76	95,25
13	2020-04-16 18:29:59	2020-04-16 18:34:15	0:04:16	2020-04-16 18:34:15	2020-04-16 18:41:30	0:07:15	98,46		3:56:52	0:03:42	98,46	95,25
14	2020-04-16 18:41:30	2020-04-16 18:47:53	0:06:23	2020-04-16 18:47:53	2020-04-16 18:48:20	0:00:27	98,45		3:35:55	0:03:25	98,45	95,25
15	2020-04-16 18:48:20	2020-04-16 19:00:03	0:11:43	2020-04-16 19:00:03	2020-04-16 19:00:53	0:00:50	98,42		3:18:54	0:03:12	98,42	95,25
16	2020-04-16 19:00:53	2020-04-16 19:16:54	0:16:01	2020-04-16 19:16:54	2020-04-16 19:21:08	0:04:14	98,26		3:04:50	0:03:16	98,26	95,25
17	2020-04-16 19:21:08	2020-04-21 23:32:56	124:11:48	2020-04-21 23:32:56	2020-04-21 23:35:29	0:02:33	99,54		11:43:54	0:03:13	99,54	95,25
18	2020-04-21 23:35:29	2020-04-21 23:36:02	0:00:33	2020-04-21 23:36:02	2020-04-21 23:37:04	0:01:02	99,53		10:57:01	0:03:05	99,53	95,25
19	2020-04-21 23:37:04	2020-04-21 23:54:15	0:17:11	2020-04-21 23:54:15	2020-04-21 23:54:22	0:00:07	99,53		10:17:01	0:02:54	99,53	95,25

Picture 5.11 - My example sheet for entering bot startup and crash times

Based on the collected data from the spreadsheet, I created a table of my bot's availability in individual months (Picture 5.12 - Collected data on bot availability and failure rate).

At the beginning, an explanation of a few symbols:

- 'TOTAL MONTH'S HOURS' - gives the number of hours in a given month,
- 'RU' - sum of all times for which the bot was available,
- 'RE' - sum of all times the bot was unavailable,
- 'RU [%]' - the ratio of the bot's availability to the number of hours in a given month, expressed as a percentage,
- 'RE [%]' - the ratio of bot unavailability to the number of hours in a given month, expressed as a percentage,
- 'MTTR - F' - number of failures in a given month,
- 'MTTR' - average time from failure to repair,
- 'MTTF' - the average time from the last failure or launch of the bot to the next failure,
- 'MTBF' - mean time between failures - it is the sum of MTTR and MTTF,
- 'AVAILABILITY' - bot availability expressed as a percentage.

MTTR is calculated from the formula:

$$MTTR=\sum_{i=1}^{N} T_i$$

where:

- Ti - failure time,
- N - number of failures.

The MTTF is calculated from the formula:

$$MTTF=\frac{T_D-\sum_{i=1}^{N} T_i}{N+1}$$

where:

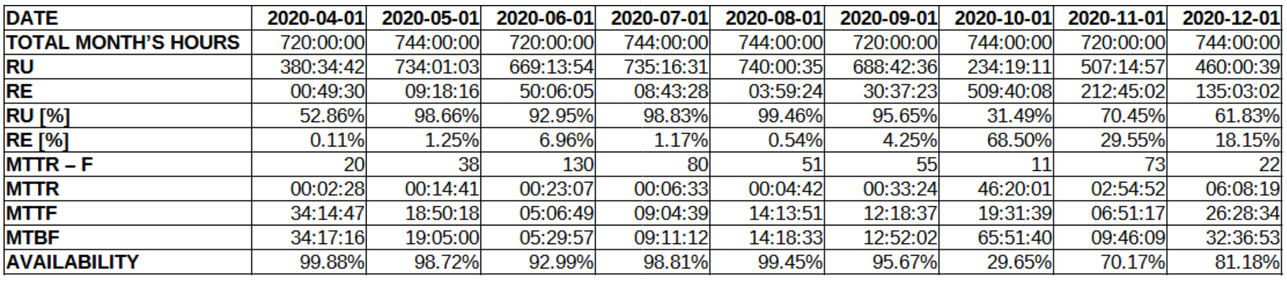
- T_D - bot availability time,
- T_i - bot failure time.
- N - number of failures.

The availability of the bot is calculated from the formula:

$$AVAILABILITY=\frac{MTTF}{MTTF+MTTR}\cdot 100$$

By a failure, I mean any event when my bot was unavailable.

Now let's analyze the picture with the results.



DATE	2020-04-01	2020-05-01	2020-06-01	2020-07-01	2020-08-01	2020-09-01	2020-10-01	2020-11-01	2020-12-01
TOTAL MONTH'S HOURS	720:00:00	744:00:00	720:00:00	744:00:00	744:00:00	720:00:00	744:00:00	720:00:00	744:00:00
RU	380:34:42	734:01:03	669:13:54	735:16:31	740:00:35	688:42:36	234:19:11	507:14:57	460:00:39
RE	00:49:30	09:18:16	50:06:05	08:43:28	03:59:24	30:37:23	509:40:08	212:45:02	135:03:02
RU [%]	52.86%	98.66%	92.95%	98.83%	99.46%	95.65%	31.49%	70.45%	61.83%
RE [%]	0.11%	1.25%	6.96%	1.17%	0.54%	4.25%	68.50%	29.55%	18.15%
MTTR – F	20	38	130	80	51	55	11	73	22
MTTR	00:02:28	00:14:41	00:23:07	00:06:33	00:04:42	00:33:24	46:20:01	02:54:52	06:08:19
MTTF	34:14:47	18:50:18	05:06:49	09:04:39	14:13:51	12:18:37	19:31:39	06:51:17	26:28:34
MTBF	34:17:16	19:05:00	05:29:57	09:11:12	14:18:33	12:52:02	65:51:40	09:46:09	32:36:53
AVAILABILITY	99.88%	98.72%	92.99%	98.81%	99.45%	95.67%	29.65%	70.17%	81.18%

Picture 5.12 - Collected data on bot availability and failure rate

The ratio of the bot's availability to the number of hours in a given month, expressed as a percentage, varies in the range of 31.49% - 99.46%. In April, this value was small, because the bot was launched from mid - April, then we see in the following months that the value remains in the range of 92% - 99%, then there is a sharp drop in October, then in November it increases and in December the bot is closed with the availability of approx. 62%. This is also visible in the ratio of the bot's unavailability to the number of hours in a given month, expressed as a percentage, where it is the highest in October, and then it drops. Before November, it is at a low acceptable level. On average, the bot ran from about 5 - 34 hours a month until there was a failure. The failure in the form of the bot's unavailability lasted from 2 minutes to even about 46 hours. At the end, you can read the availability of the bot, which is based on the bot's availability and failure time. It can be seen as before that in October there was a dramatic decrease in the availability of the bot due to failures.

Why has the number of failures been so high since October? Because after updating the program, there were various problems when the bot could not start.

On November 26, 2020, errors began to occur with the correct display of avatars - the appearance of the avatar resembled a cloud, which usually symbolizes loading its appearance while in Second Life. The incident was called 'Thanksgiving Bakefail'. This bug arrived to Corrade sometime in early December, when bot closet unavailability issues began to be reported, which prevented him from having a look at items or his appearance. The bug on the Second Life side was fixed after about 2 - 3 weeks.

Another failure also occurred on the Second Life side in November, where the region where the bot was located had huge delays, which caused either the inability to fire key scripts in the bot's clothes or their slow performance. In this situation, I decided to move to another region where the delays were minimal.

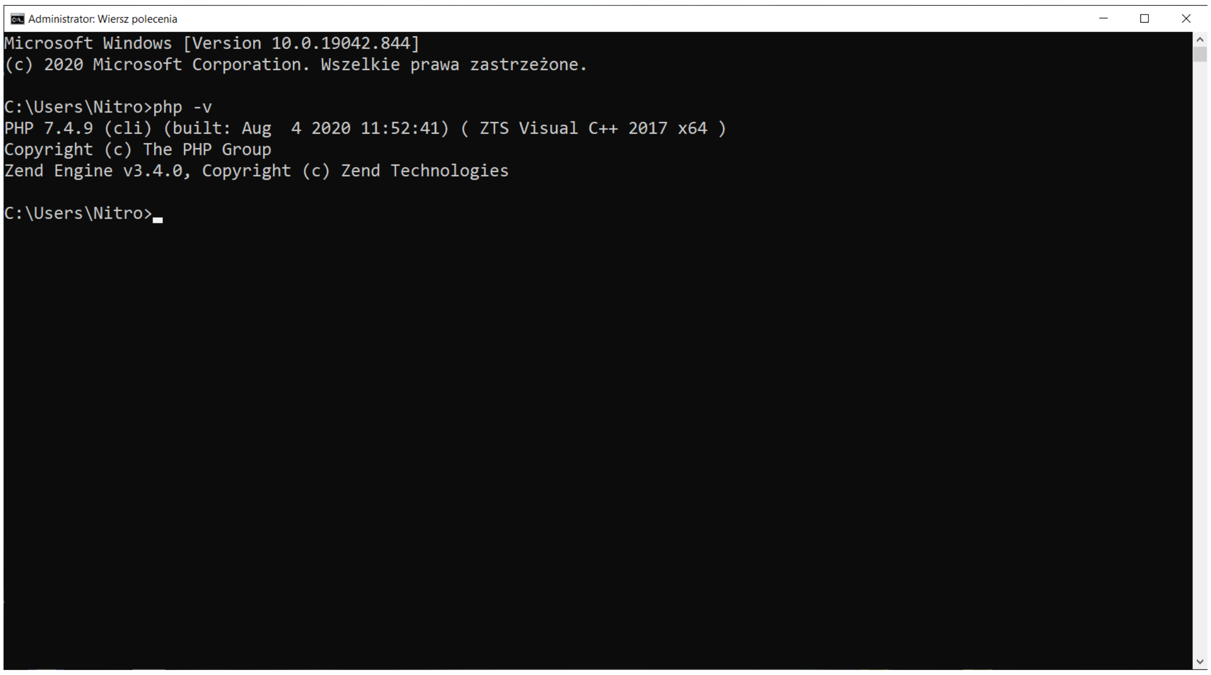
5.4. The basics of building a bot

5.4.1. Introduction

In this chapter I will show you how to build a basic bot.
Our assumptions:

- we do all processing outside the Second Life environment,
- the programming language in which we will create the bot's brain and heart outside Second Life is PHP,
- The heart of a bot in Second Life will be in the form of a prim that will be part of the bot's clothing.

First, check if PHP is added to the system path, which will allow us to call the PHP program from anywhere in the system.

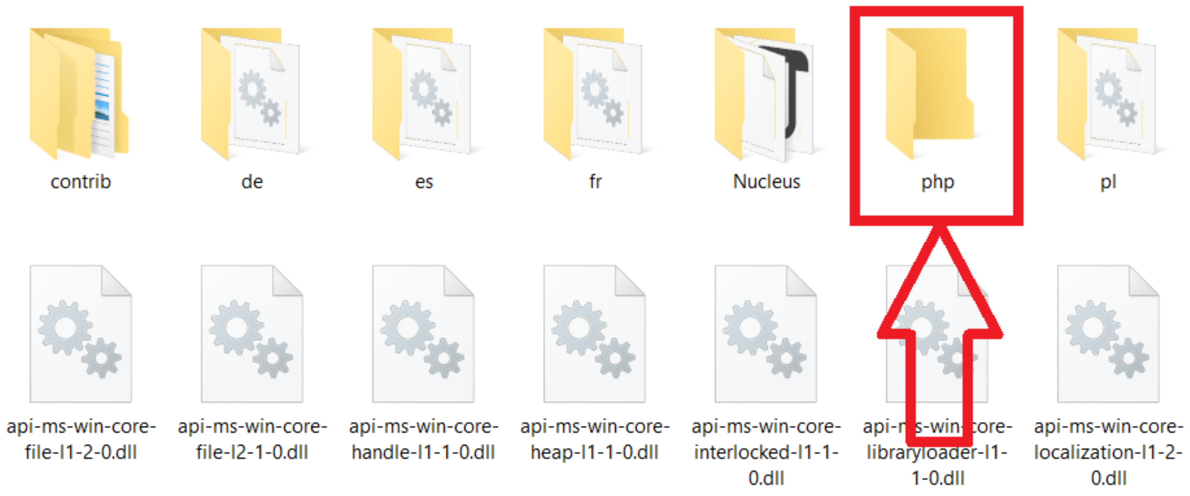


```
Microsoft Windows [Version 10.0.19042.844]
(c) 2020 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Nitro>php -v
PHP 7.4.9 (cli) (built: Aug  4 2020 11:52:41) ( ZTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies

C:\Users\Nitro>
```

Picture 5.13 - Check if PHP is added to the system path

If after confirming the command does not appear, among others PHP version and the information that the program was not found appears, download PHP version 7.x for Windows from the website https://windows.php.net/download/ in the form of a ZIP archive. We choose the x64 Thread Safe version. Unpack the archive, e.g. to C:\php. Go to the system properties (Right - click on 'This computer' or 'My computer' and then select 'Properties'). Then select 'Advanced system settings' => on the 'Advanced' tab click on 'Environment variables'. In the 'System Variables' section, find 'Path', add the path to the C:\php folder to it. Confirm everything, restart the computer.

In your bot folder create a folder called 'php'. In this folder, we're going to keep the bot's heart. It is also an easy way to call individual parts written in PHP.


contrib
de
es
fr
Nucleus
php
pl
api-ms-win-core-file-l1-2-0.dll
api-ms-win-core-file-l2-1-0.dll
api-ms-win-core-handle-l1-1-0.dll
api-ms-win-core-heap-l1-1-0.dll
api-ms-win-core-interlocked-l1-1-0.dll
api-ms-win-core-libraryloader-l1-1-0.dll
api-ms-win-core-localization-l1-2-0.dll

Picture 5.14 - The PHP folder in the Corraed folder

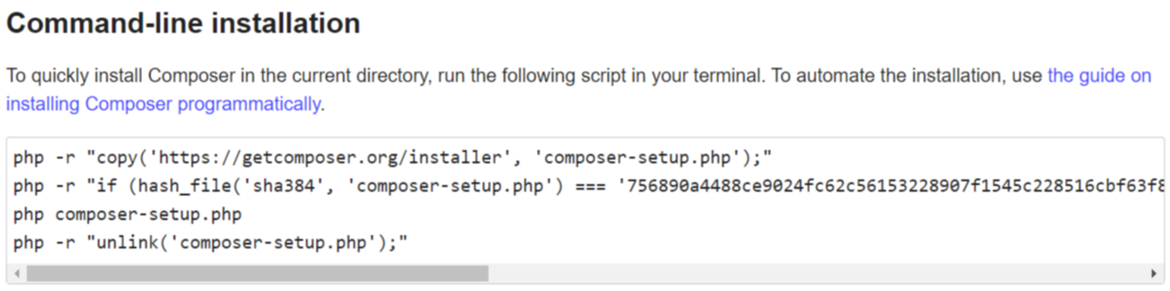
We go to the previously created folder.

In this folder, we will first install Composer, which allows you to easily manage project dependencies. First, the command line runs in the PHP folder. To do this, press the left SHIFT key and right - click somewhere on an empty area of the folder area, and then select 'Open window here' from the menu.

Attention! If there is no such option, you can add appropriate entries to the registry or in the Start Menu, enter cmd and then use the cd command to go to the folder of our choice.

We go to the Composer website (https://getcomposer.org/download/), where we can find commands to enter in the command line to download Composer.

We copy all lines in one go and paste to the command line (use the right mouse button).



Command-line installation

To quickly install Composer in the current directory, run the following script in your terminal. To automate the installation, use the guide on installing Composer programmatically.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === '756890a4488ce9024fc62c56153228907f1545c228516cbf63f8
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Picture 5.15 - Composer installation

If everything went well, you will find composer.phar in the php folder.

To invoke Composer help, type:

```
php composer.phar
```

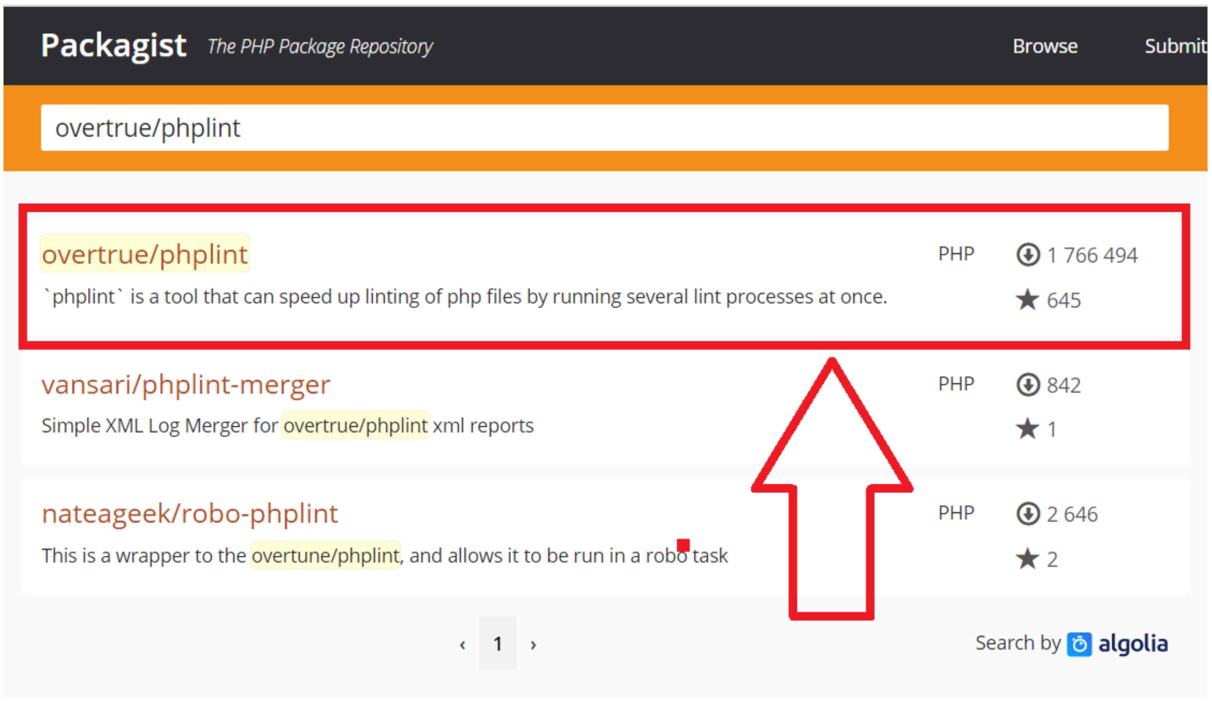
The most important commands are:

- php composer.phar require PackageName - installs the given package,
- php composer.phar remove PackageName - removes a previously installed package,
- php composer.phar update - updating all installed packages,
- php composer.phar selfupdate - updates Composer,
- php composer.phar outdated - shows obsolete versions of installed packages.

You can find the packages at https://packagist.org/, they are simply PHP projects that are hosted on the GIT server. You can download them yourself without Composer, incorporate them into your project, however you will have to keep an eye on whether there are new versions on project pages etc., so Composer just makes the process easier.

First, let's install a package called: overtrue/phplint, which allows you to check the syntax of PHP files. It will be useful for us while working on the heart of the bot in catching errors in PHP syntax that may block the work of our bot.

On the packagist.org website, enter the name of this package and click on the first proposal from the found list.



Packagist The PHP Package Repository Browse Submit

overtrue/phplint

overtrue/phplint PHP 1 766 494
`phplint` is a tool that can speed up linting of php files by running several lint processes at once. ★ 645

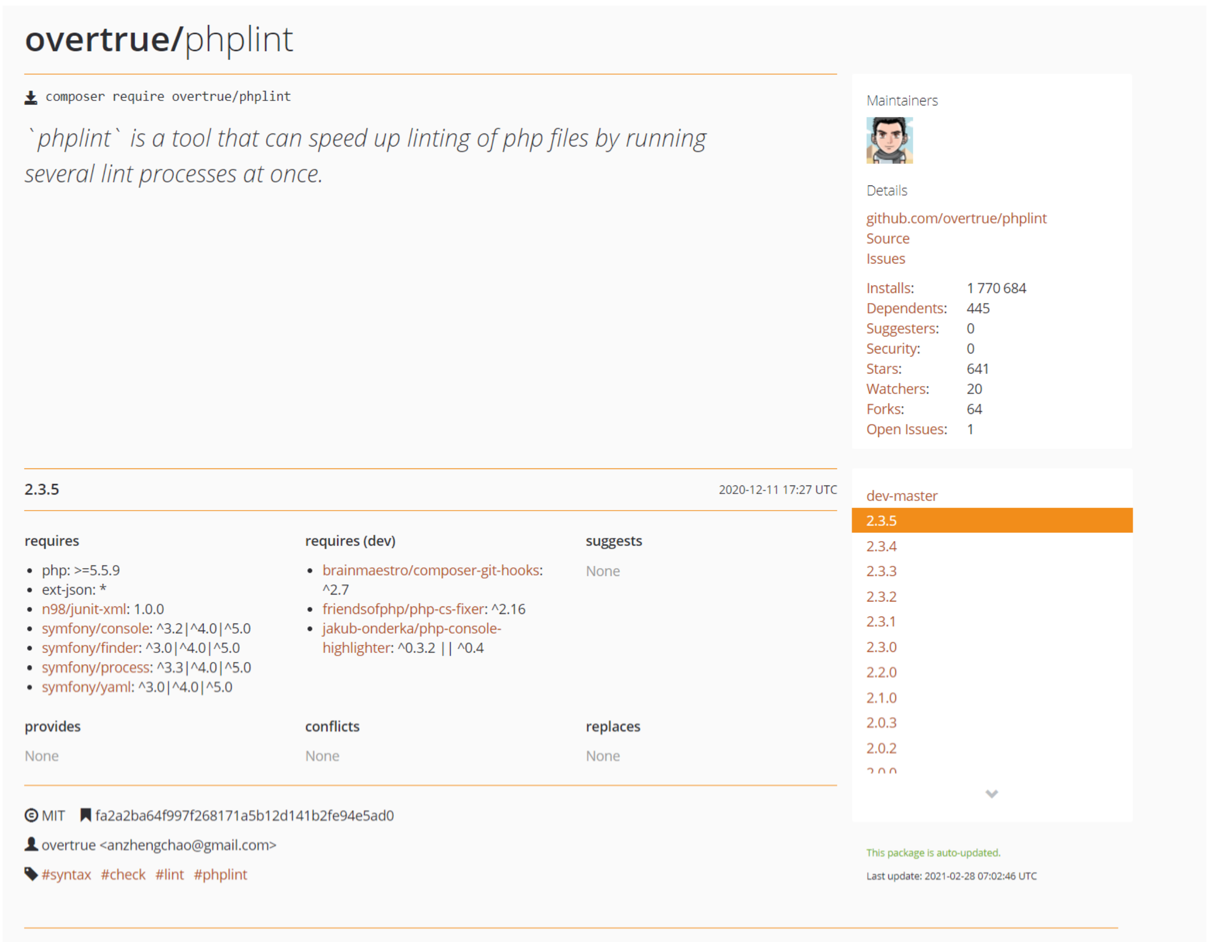
vansari/phplint-merger PHP 842
Simple XML Log Merger for overtrue/phplint xml reports ★ 1

nateageek/robo-phplint PHP 2 646
This is a wrapper to the overtune/phplint, and allows it to be run in a robo task ★ 2

‹ 1 › Search by algolia

Picture 5.16 - Search for a package on the Packagist website

On this page we have the details of the package. We should pay attention to the requirements of a given package. Yes, we do not have to check its requirements for each package, because Composer checks during installation whether the package requirements are compatible with the state of the computer on which the package is to be installed. So if something goes wrong it will cancel the installation and notify you which parts of the requirements do not meet your expectations.


overtrue/phplint

composer require overtrue/phplint

`phplint` is a tool that can speed up linting of php files by running several lint processes at once.

Maintainers

Details

github.com/overtrue/phplint
Source
Issues

Installs:	1 770 684
Dependents:	445
Suggesters:	0
Security:	0
Stars:	641
Watchers:	20
Forks:	64
Open Issues:	1

2.3.5 2020-12-11 17:27 UTC

requires
- php: >=5.5.9
- ext-json: *
- n98/junit-xml: 1.0.0
- symfony/console: ^3.2|^4.0|^5.0
- symfony/finder: ^3.0|^4.0|^5.0
- symfony/process: ^3.3|^4.0|^5.0
- symfony/yaml: ^3.0|^4.0|^5.0

requires (dev)
- brainmaestro/composer-git-hooks: ^2.7
- friendsofphp/php-cs-fixer: ^2.16
- jakub-onderka/php-console-highlighter: ^0.3.2 || ^0.4

suggests
None

provides
None

conflicts
None

replaces
None

MIT fa2a2ba64f997f268171a5b12d141b2fe94e5ad0

overtrue <anzhengchao@gmail.com>

#syntax #check #lint #phplint

dev-master
2.3.5
2.3.4
2.3.3
2.3.2
2.3.1
2.3.0
2.2.0
2.1.0
2.0.3
2.0.2

This package is auto-updated.

Last update: 2021-02-28 07:02:46 UTC

Picture 5.17 - Website overtrue/phplint

We return to the installation of the package.

To do this, enter in the command line:

```
php composer.phar require overtrue/phplint
```

The command will download the package with dependencies.

Comments!

- from time to time (e.g. every 2 weeks or once a month) we check packages for updates via: php composer.phar update,
- obsolete packages - by php composer.phar outdated and, if necessary, we remove them and install them again,
- let's also update the Composer itself.

If everything was successful, then in the folder we will find the vendor folder - our packages are kept here, which we will include in the project, composer.json - a file with information about what packages and what versions have been installed, composer.lock - a file with more information about the installed packages.

Now let's create a phplint.bat file to which we copy the following content:

```
SET mypath=%~dp0
SET mypath=%mypath:~0,-1%
vendor\bin\phplint -j 5 --no-cache -n --exclude=vendor -vvv "%mypath%" -n >
phplint.txt
```

It allows you to use the mentioned phplint with the exclusion of the vendor folder, with full diagnostics, where everything is saved to the phplint.txt file.

Next, we create one of the key PHP files called servebot.php (maybe a different name) to handle requests from Second Life.

We paste the following content into it:

```php
<?php
error_reporting(0);

chdir(__DIR__);

ini_set('memory_limit', '512M');
set_time_limit(300);
date_default_timezone_set('UTC');

require_once('vendor/autoload.php');
require_once('config.php');
require_once('inPolygon.class.php');
require_once('functions.php');

$args = $argv;
$args = array_slice($args, 1);
```

```
$args = implode(' ', $args);
preg_match('/\&\_script\=([^&]+)/', $args, $found);
$scriptNameExternal = (isset($found[1])) ? $found[1] : '';
$args = preg_replace('/(\?|\&)([^=]+)\=([^&]+)/', '', $args);
$body = explode('|', $args);
$body_t = array();
for($i=0;$i<count($body);$i+=2):
      $body_t[ $body[$i] ] = $body[$i+1];
endfor;
$body = $body_t;

switch($body['action']){


}
```

Explanation:

- error_reporting(0) - disable any error reporting,
- chdir() - does everything in the current folder
- ini_set('memory_limit', '512M'); - sets the maximum memory limit to 512 MB for all scripts,
- date_default_timezone_set('UTC'); - sets default time zone to universal,
- require_once('vendor/autoload.php'); - attaches the file vendor/autoload.php from Composer,
- require_once('config.php'); - attaches the configuration file config.php (we will create it later),
- require_once('inPolygon.class.php'); - attaches the inPolygon.class.php file related to checking whether a given point belongs to a polygon (we will create it later),
- require_once('functions.php'); - includes the functions.php file related to functions (we'll create it later).

The following lines allow the parameters sent to this file to be converted into an array.

A bot from Second Life can trigger a character similar to:

```
php\serverbot.php 'parameter1|value1|parameter2|value2 |...'
```

then this part of this file will transform into an array:

```
$body['parameter1'] = 'value1';
$body['parameter2'] = 'value2';
```

The separator is a vertical line '|'.

This part of the script also, just in case, extracts the name of the LSL script that previously called the servebot.php file (if that name was passed).

switch($body['action']) - We preimplemented a switch after the argument values for 'action', which will be responsible for calling the action.

```
php\serverbot.php 'action|action name|parameter2|value2|...'
```

Create the functions.php file with essentially empty content for now:

```
<?php
```

Create an inPolygon.class.php file with the following content:

```
<?php
/*
Description: The point-in-polygon algorithm allows you to check if a point is
inside a polygon or outside of it.
Author: Michaël Niessen (2009)
Website: http://AssemblySys.com

If you find this script useful, you can show your
appreciation by getting Michaël a cup of coffee ;)
donation to Michaël


As long as this notice (including author name and details) is included and
UNALTERED, this code is licensed under the GNU General Public License version 3:
http://www.gnu.org/licenses/gpl.html
*/

class pointLocation {
    var $pointOnVertex = true; // Check if the point sits exactly on one of the
vertices?

    function pointLocation() {
    }

    function pointInPolygon($point, $polygon, $pointOnVertex = true) {
        $this->pointOnVertex = $pointOnVertex;

        // Transform string coordinates into arrays with x and y values
        $point = $this->pointStringToCoordinates($point);
        $vertices = array();
        foreach ($polygon as $vertex) {
            $vertices[] = $this->pointStringToCoordinates($vertex);
        }
        // Check if the point sits exactly on a vertex
        if ($this->pointOnVertex == true and $this->pointOnVertex($point,
$vertices) == true) {
            return "vertex";
        }

        // Check if the point is inside the polygon or on the boundary
        $intersections = 0;
        $vertices_count = count($vertices);

        for ($i=1; $i < $vertices_count; $i++) {
            $vertex1 = $vertices[$i-1];
            $vertex2 = $vertices[$i];
            if ($vertex1['y'] == $vertex2['y'] and $vertex1['y'] == $point['y'] and
$point['x'] > min($vertex1['x'], $vertex2['x']) and $point['x'] <
max($vertex1['x'], $vertex2['x'])) { // Check if point is on an horizontal polygon
boundary
                return "boundary";
            }
            if ($point['y'] > min($vertex1['y'], $vertex2['y']) and $point['y'] <=
max($vertex1['y'], $vertex2['y']) and $point['x'] <= max($vertex1['x'],
$vertex2['x']) and $vertex1['y'] != $vertex2['y']) {
                $xinters = ($point['y'] - $vertex1['y']) * ($vertex2['x'] -
$vertex1['x']) / ($vertex2['y'] - $vertex1['y']) + $vertex1['x'];
                if ($xinters == $point['x']) { // Check if point is on the polygon
boundary (other than horizontal)
                    return "boundary";
                }
                if ($vertex1['x'] == $vertex2['x'] || $point['x'] <= $xinters) {
                    $intersections++;
                }
            }
        }
        // If the number of edges we passed through is odd, then it's in the
polygon.
        if ($intersections % 2 != 0) {
            return "inside";
        } else {
            return "outside";
        }
    }

    function pointOnVertex($point, $vertices) {
        foreach($vertices as $vertex) {
            if ($point == $vertex) {
                return true;
            }
        }

    }

    function pointStringToCoordinates($pointString) {
        $coordinates = explode(" ", $pointString);
        return array("x" => $coordinates[0], "y" => $coordinates[1]);
    }

}
?>
```

We create the config.php file into which we paste the content:

```
<?php
//Bot Connection
define('BOT_GROUP',       'GROUPNAME');
define('BOT_PASSWORD',    'PASSWORD');
define('BOT_URL',         'http://127.0.0.1:8080');

//constants Second Life
define('TEXTURE_DEFAULT',        '89556747-24cb-43ed-920b-47caed15465f');
define('TEXTURE_BLANK',                '5748decc-f629-461c-9a36-a35a221fe21f');
define('TEXTURE_TRANSPARENT',    '8dcd4a48-2d37-4909-9f78-f7a9eb4ef903');
```

- instead of GROUPNAME, we give the name of the group to which the bot belongs,
- instead of PASSWORD we give the password to this group,
- in BOT_URL we provide the URL to the bot HTTP server (127.0.0.1 - scripts are executed locally on the same computer on which the bot will be running, 8080 - the port must be the same as in the configuration),
- constants prefixed with TEXTURE_ define the UUID of textures in Second Life (default, empty, transparent).

Now we should consider where we will store further configuration for our bot. If you have additional materials attached to this ebook, in bot_installer\corrade\bot_ai\php, the configuration is stored in a separate INI file, in this chapter we will keep everything in the config.php file.

When you make changes to this file, call phplint.bat just in case and check for any syntax errors in the phplint.txt file.

Now let's consider what 'actions' our bot is supposed to process.
My suggestions are given below.

5.4.2. Suggested functions for the bot

- **Retrieving a value defined in a configuration file.**

In servebot.php in switch we add:

```
       case 'getConfig':
             echo getConfig($body['name']);
       break;
 ///////////////////
```

It adds that part of the action called getConfig that will allow you to get the value of a constant variable defined in config.php.
Here we have a function calling that takes a constant variable name as a parameter.

Example of calling:

```
php\serverbot.php 'action|getConfig|name|NameOfConstantVariable'
```

In the functions.php file we add:

```
function getConfig($name)
{
  return (defined($name))? constant($name) : '';
}
```

This function checks if a constant variable in config.php exists, if so, takes its value, if not defined, returns an empty string.

- **Cache support**

In my bot project there is cache support, i.e. storing the most frequently used values by the bot without the need to recalculate, process or query Second Life. There are two sources of writing: to the cache.ini file or to the database. I usually choose to save to the cache file.

In servebot.php in switch we add:

```
case 'setCache':
  setCacheValue($body['name'], $body['value']);
break;
//////////////////
case 'getCache':
  echo getCacheValue($body['name']);
break;
//////////////////
case 'deleteCache':
  deleteCacheName($body['name']);
break;
//////////////////
```

Method of calling:

```
php\serverbot.php 'action|setCache|name|ParameterNameInCache|value|ParameterValue'
php\serverbot.php 'action|getCache|name|ParameterNameInCache'
php\serverbot.php 'action|deleteCache|name|ParameterNameInCache'
```

in config.php add:

```
//Database Connection
define('DB_CONNECTION', 'mysql:host=localhost;dbname=corrade');
define('DB_LOGIN', 'root');
define('DB_PASSWORD', '12345');

define('GENERAL_cacheTypeWriteDatabase', '0');
```

in functions.php add:

```
//Insert or Update
function setCacheValue($name, $value)
{
    if(getConfig('GENERAL_cacheTypeWriteDatabase') == '1'):
        try
        {
            $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
            $pdo->beginTransaction();
            $pdo->exec('LOCK TABLE cache WRITE, cache as cacheRead READ');
            $stmt = $pdo->prepare('INSERT INTO 'cache' ('name', 'value') VALUES (:name,
:value) ON DUPLICATE KEY UPDATE value=:value');
            $stmt->execute(array(
                ':name' => $name,
                ':value' => $value,
            ));
            $pdo->commit();
            $pdo->exec('UNLOCK TABLES');
        }
        catch(Exception $e)
        {
            writeDebugFile($e->getTraceAsString());
        }
    else:
        safefilerewriteCache($name, $value);
    endif;
}

function writeDebugFile($msg, $type = 'warn')
{
    @mkdir('logs');
    safefilerewrite('logs/log_'.date('d.m.Y').'.txt', date('d.m.Y H:i:s') . ' ' . $type
. ' ' . $msg . "\n", 'a+');
}

function safefilerewriteCache($name, $value)
{
    $iniCache = parse_ini_file("cache.ini", true);
    $iniCache['cache'][$name] = $value;
    write_php_ini($iniCache, "cache.ini");
}

function safefilerewrite($fileName, $dataToSave, $mode = 'w+')
{
    if ($fp = fopen($fileName, $mode))
    {
        $startTime = microtime(TRUE);
        do
        {
            $canWrite = flock($fp, LOCK_EX);
            // If lock not obtained sleep for 0 - 100 milliseconds, to avoid collision
and CPU load
            if(!$canWrite) usleep(round(rand(0, 100)*1000));
        } while ((!$canWrite)and((microtime(TRUE)-$startTime) < 5));
        //file was locked so now we can store information
        if ($canWrite)
        {
            fwrite($fp, $dataToSave);
            flock($fp, LOCK_UN);
        }
        fclose($fp);
    }
}

function write_php_ini($array, $file)
{
    $res = array();
    foreach($array as $key => $val)
    {
        if(is_array($val))
        {
            $res[] = "[$key]";
            foreach($val as $skey => $sval) $res[] = "$skey = ".(is_numeric($sval) ?
$sval : '"'.$sval.'"');
        }
        else $res[] = "$key = ".(is_numeric($val) ? $val : '"'.$val.'"');
    }
    safefilerewrite($file, implode("\r\n", $res));
}

//Get Cache Value
function getCacheValue($name)
{
    if(getConfig('GENERAL_cacheTypeWriteDatabase') == '1'):
        try
        {
            $value = "";
            $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
            $pdo->beginTransaction();
            $pdo->exec('LOCK TABLE cache WRITE, cache as cacheRead READ');
            $stmt = $pdo->prepare('SELECT 'value' from 'cache' as cacheRead WHERE
name=:name');
            $stmt->execute(array(
                ':name' => $name,
            ));
```

```php
            $data = $stmt->fetch();
            $value = $data['value'];
            $pdo->commit();
            $pdo->exec('UNLOCK TABLES');
            return $value;
        }
        catch(Exception $e)
        {
            writeDebugFile($e->getTraceAsString());
            return "";
        }
    else:
        $iniCache = parse_ini_file("cache.ini", true);
        return $iniCache['cache'][$name];
    endif;
}

//delete cache
function deleteCacheName($name)
{
    if(getConfig('GENERAL_cacheTypeWriteDatabase') == '1'):
        try
        {
            $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
            $pdo->beginTransaction();
            $pdo->exec('LOCK TABLE cache WRITE, cache as cacheRead READ');
            $stmt = $pdo->prepare('DELETE FROM cache WHERE name=:name');
            $stmt->execute(array(
                ':name' => $name,
            ));
            $pdo->commit();
            $pdo->exec('UNLOCK TABLES');
        }
        catch(Exception $e)
        {
            writeDebugFile($e->getTraceAsString());
        }
    else:
        safefilerewriteDeleteCache($name);
    endif;
}
```

```
function safefilerewriteDeleteCache($name)
{
    if ($fp = fopen('cache.ini', 'w+'))
    {
        $startTime = microtime(TRUE);
        do
        {
            $canWrite = flock($fp, LOCK_EX);
            if(!$canWrite) usleep(round(rand(0, 100)*1000));
        } while ((!$canWrite)and((microtime(TRUE) - $startTime) < 5));

        if ($canWrite)
        {
            $iniCache = parse_ini_file("cache.ini", true);
            unset($iniCache['cache'][$name]);

            $res = array();
            foreach($iniCache as $key => $val)
            {
                if(is_array($val))
                {
                    $res[] = "[$key]";
                    foreach($val as $skey => $sval)
                        $res[] = "$skey = ".(is_numeric($sval) ? $sval : '"'.
$sval.'"');
                }else{
                    $res[] = "$key = ".(is_numeric($val) ? $val : '"'.$val.'"');
                }
            }

            fwrite($fp, implode("\r\n", $res));
            flock($fp, LOCK_UN);
        }
        fclose($fp);
    }
}

//delete all from cache
function deleteAllCache()
```

```
{
    if(getConfig('GENERAL_cacheTypeWriteDatabase') == '1'):
        try
        {
            $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
            $pdo->beginTransaction();
            $pdo->exec('LOCK TABLE cache WRITE, cache as cacheRead READ');
            $stmt = $pdo->prepare('DELETE FROM cache');
            $stmt->execute();
            $pdo->commit();
            $pdo->exec('UNLOCK TABLES');
        }
        catch(Exception $e)
        {
            writeDebugFile($e->getTraceAsString());
        }
    else:
        @unlink('cache.ini');
        @unlink('php/cache.ini');
    endif;
}
```

A question of clarification.

We have added to switch the ability to save to the cache, download values from the cache with a given variable, delete in the cache with a given variable, delete all variables in the cache.

In config.php we added database entries and the cache type (GENERAL_cacheTypeWriteDatabase).

If this variable is:

- 0 - save to cache.ini,
- 1 - writes the cache to the database.

If you have selected to save to the database, set the correct values in the previous variables and add a new cache table to the database:

```
CREATE TABLE IF NOT EXISTS 'cache' (
  'name' varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  'value' longtext COLLATE utf8mb4_unicode_ci NOT NULL,
PRIMARY KEY ('name')
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

In functions.php, we've added all the functions that support our cache.
In addition, we find here universal functions for safe data writing to a file.

- **Processing private messages**

As an example, we will do the ability to process private messages sent to the bot.
In servebot.php in switch we add:

```
case 'serveIM':
    $avatar = AvatarUUIDtoName($body['uuid']);
    //check ban
    if(BlackListIMExist($body['uuid'])):
        SendMessageToAvatar($body['uuid'], $avatar.", I don't talk with you. You are 
very rude.");
        exit();
    endif;
    require_once('part.im.php');
break;
///////////////////
```

This part of the code adds an action named 'serveIM'.
At the very beginning, we use the AvatarUUIDtoName() function to replace the sent UUID of the avatar that sent the message to the bot into its name and surname, which will be used when sending a reply (imitation of a personalized message).
Next we have a small part with the possibility of checking if a given avatar is not blacklisted. and if it is, an appropriate message is sent.
In fact, if we do not want to protect against the stream of messages or against the use of administrative commands by unauthorized persons, we can delete this part.
Attention! To use this option, you must configure a database connection and add a table:

```
CREATE TABLE IF NOT EXISTS 'blacklist.im' (
  'agentuuid' varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  'agentname' text COLLATE utf8mb4_unicode_ci NOT NULL,
  'time' text COLLATE utf8mb4_unicode_ci NOT NULL,
  PRIMARY KEY ('agentuuid')
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Next we have attached the part.im.php file in which we will process the messages.

It is best to develop individual parts in the switch in separate PHP files, this will allow you to control the order in the file.

In functions.php we add:

```php
function AvatarUUIDtoName($uuid)
{
    $params = array(
        "command" => "batchavatarkeytoname",
        "group" => BOT_GROUP,
        "password" => BOT_PASSWORD,
        "avatars" => wasArrayToCSV(array($uuid))
    );

    $ret = SendToBot($params);
    $ret = wasCSVToArray(wasKeyValueGet("data", $ret));
    return $ret[1];
}

function BlackListIMExist($uuid)
{
    try {
        $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
        $stmt = $pdo->prepare('SELECT COUNT(*) from 'blacklist.im' WHERE
agentuuid=:uuid');
        $stmt->execute(array(
            ':uuid' => $uuid
        ));
        if($stmt->fetchColumn() == 1):
            return 1;
        endif;
        return 0;
    }catch(Exception $e1){
        return 0;
    }
}

function SendMessageToAvatar($agent, $message)
{
    $params = array(
        'command' => 'tell',
        'group' => BOT_GROUP,
        'password' => BOT_PASSWORD,
        'message' => $message,
        'entity' => 'avatar',
        'agent' => $agent
    );
    SendToBot($params, 0);
}

function wasArrayToCSV($a)
{
    return implode(',',
        array_map(
        function($o)
        {
            $o = str_replace('"', '""', $o);
            switch( (strpos($o, ' ') !== FALSE) || (strpos($o, '"') !== FALSE) ||
(strpos($o, ',') !== FALSE) || (strpos($o, '\r') !== FALSE) || (strpos($o, '\n') !==
FALSE))
            {
                case TRUE:
                    return '"' . $o . '"';
                default:
                    return $o;
            }
        },$a)
    );
}

function SendToBot($param_tab, $ACK = 1, $connectTimeout = 10, $Timeout = 30)
{
    $getTypeProcessingLang = 'was';
    if(file_exists('../Configuration.xml')):
        $xmldata = simplexml_load_file('../Configuration.xml');
        $getTypeProcessingLang = trim(strtolower($xmldata->ScriptLanguage));
    endif;

    if($getTypeProcessingLang == 'json'):
        $postvars = json_encode($param_tab);
    else:
        array_walk(
        $param_tab, function(&$value, $key)
        {
            $value = rawurlencode($key)."=".rawurlencode($value);
        });
        $postvars = implode('&', $param_tab);
    endif;

    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, BOT_URL);
    curl_setopt($curl, CURLOPT_POST, true);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $postvars);
    curl_setopt($curl, CURLOPT_ENCODING, true);
    curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, $connectTimeout);
    curl_setopt($curl, CURLOPT_TIMEOUT, $Timeout);

    if($ACK == 1):
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
        $return = curl_exec($curl);
        return $return;
    else:
        ob_start();
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, 0);
        curl_setopt($curl, CURLOPT_TIMEOUT, 1);
        curl_setopt($curl, CURLOPT_HEADER, 0);
        curl_setopt($curl, CURLOPT_FORBID_REUSE, 1);
        curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, 1);
        curl_setopt($curl, CURLOPT_FRESH_CONNECT, 1);
        curl_exec($curl);
        ob_end_clean();
    endif;

    curl_close($curl);
}

function wasCSVToArray($csv)
{
    $csv = preg_replace('/\"/', '', $csv);
    $csv = preg_replace('/,/', '","', $csv);
    if(substr($csv, 0, 1) != '"') $csv = '"' . $csv;
    if(substr($csv, -1, 1) != '"') $csv .= '"';
    $csv = preg_replace('/"<([\d]+(?:\.[\d]+)?)[",\s]*([\d]+(?:\.[\d]+)?)[",\s]*([\d]+
(?:\.[\d]+)?)>"/U', '"<$1,$2,$3>"', $csv);
    preg_match_all('/"(.*)"/U', $csv, $l);
    return $l[1];
}

function wasKeyValueGet($key, $data, $base64decode = false)
{
    $sreturn = '';
    if($LANG == 'json'):
        $sret_json = json_decode($data);
        $sreturn = $sret_json->$key;
        if(is_array($sreturn)):
            $sreturn = implode('', $sreturn);
        endif;
    else:
        parse_str($data, $response);
        $sreturn = $response[$key];
    endif;

    $sreturn = rawurldecode($sreturn);
    $sreturn = preg_replace('/^\'+/', '', $sreturn);
    $sreturn = preg_replace('/\'+$/', '', $sreturn);
    $sreturn = preg_replace('/\"/', '', $sreturn);
    $sreturn = trim($sreturn);

    if($base64decode):
        $sreturn = base64_decode($sreturn);
    endif;

    if(strtolower($key) == 'success'):
        $sreturn = (strtolower($sreturn) == 'true') ? '1' : '0';
    endif;
    if(($key == 'data') && ($sreturn == 'data')):
        $sreturn = '';
    endif;
    if(($key == 'success') && ($sreturn == 'data')):
        $sreturn = 0;
    endif;
    return $sreturn;
}
```

```
function BlackListIMAdd($uuid)
{
    try {
        $avatar = AvatarUUIDtoName($uuid);
        if(str_word_count($avatar) == 1):
            $avatar.=' resident';
        endif;

        $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
        $stmt = $pdo->prepare('INSERT INTO  'blacklist.im' ('agentuuid', 'agentname', 
'time') VALUES (:agentuuid, :agentname, :time)');
        $stmt->execute(array(
            ':agentuuid' => $uuid,
            ':agentname' => $avatar,
            ':time' => strtotime('now'),
        ));
    }catch(Exception $e1){}
}

function BlackListIMInc($uuid)
{
    try {
        $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
        $stmt = $pdo->prepare('UPDATE 'blacklist.im' SET count=count+1 WHERE 
agentuuid=:agentuuid');
        $stmt->execute(array(
            ':agentuuid' => $uuid,
        ));
    }catch(Exception $e1){}
}

function BlackListIMDel($uuid)
{
    try {
        $pdo = new PDO(DB_CONNECTION, DB_LOGIN, DB_PASSWORD);
        $stmt = $pdo->prepare('DELETE FROM 'blacklist.je' WHERE 'avatar' LIKE "%:avatar
%" OR 'uuid' LIKE "%:uuid%"');
        return (int)$stmt->execute(array(
            ':uuid' => $uuid
        ));
    }catch(Exception $e1){
        return 0;
    }
}

function checkPerm($avatar, $func)
{
    $retValue = 0;
    $params = array(
        'command' => 'getmemberroles',
        'group' => BOT_GROUP,
        'password' => BOT_PASSWORD,
        'agent' => $avatar,
        'target' => BOT_GROUP
    );
    $ret = SendToBot($params);

    if(wasKeyValueGet("success", $ret)):
        $data = wasKeyValueGet("data", $ret);
        $groups = wasCSVToArray($data);
        for($i=0;$i<count($groups);$i++):
            if($groups[$i] == $func):
                $retValue = 1;
                break;
            endif;
        endfor;
    endif;
    return $retValue;
}
```

W config.php dodaj:

```
define('GENERAL_groupFunctionAdminBot', 'botmanager');
```

Here we give the name of the role in the group where the bot is located, where the avatars rewritten for this role will be treated as bot administrators.

Of course, we also play this role in Second Life in this group. It does not need to be assigned any permissions.

In addition to the aforementioned functions, we define function bodies for general sending data to the bot (SendToBot), converting the table to CSV and vice versa, retrieving specific data obtained from the bot.

Create part.im.php

```
<?php
$uuid = $body['uuid'];
$message = trim($body['message']);
$type = $body['type'];
///////////////////
function default_no_command($uuid, $avatar){
    SendMessageToAvatar($uuid, $avatar.', I\'m sorry, but I don\'t understand.');
}
///////////////////
if(!empty($type)):
    switch($type):
        ///////////////////
        case 'error_nosameregion':
            SendMessageToAvatar($uuid, $avatar.", Sorry, you don't in the same region
as me.");
            exit();
        break;
        ///////////////////
        default:
            default_no_command($uuid, $avatar);
            exit();
        break;
    endswitch;
endif;

if( (preg_match('/hi/i', $message)) || (preg_match('/hello/i', $message))):
    $params = array(
        "command" => "displayname",
        "group" => BOT_GROUP,
        "password" => BOT_PASSWORD,
        "action" => 'get',
    );
    $ret = SendToBot($params);
    $avatarDisplayName = wasKeyValueGet("data", $ret);
    SendMessageToAvatar($uuid, "Hello ".$avatar.", My name is ".$avatarDisplayName."
and I'm bot who is living in Second Life.");
    exit();
endif;
///////////////////
if(preg_match('/blim\-add/i', $message)):
    if(!checkPerm($uuid, getConfig('GENERAL_groupFunctionAdminBot'))):
        if(!BlackListIMExist($uuid)):
            BlackListIMAdd($uuid);
        else:
            BlackListIMInc($uuid);
        endif;

        SendMessageToAvatar($uuid, $avatar.', you have not permissions to manage the
bot.');
        exit();
    endif;
    $who = trim(substr($message, stripos($message, "blim-add") + strlen("blim-add")));
    if(BlackListIMAdd($who)):
        SendMessageToAvatar($uuid, $avatar.', avatar has been added to blacklist.');
    else:
        SendMessageToAvatar($uuid, $avatar.', avatar has not added to blacklist.');
    endif;
    exit();
endif;
///////////////////
if(preg_match('/blim\-del/i', $message)):
    if(!checkPerm($uuid, getConfig('GENERAL_groupFunctionAdminBot'))):
        if(!BlackListIMExist($uuid)):
            BlackListIMAdd($uuid);
        else:
            BlackListIMInc($uuid);
        endif;

        SendMessageToAvatar($uuid, $avatar.', you have not permissions to manage the
bot.');
        exit();
    endif;
    $who = trim(substr($message, stripos($message, "blim-del") + strlen("blim-del")));
    BlackListIMDel($who);
    SendMessageToAvatar($uuid, $avatar.', avatar has been delete from blacklist.');
    exit();
endif;
///////////////////
if(preg_match('/\.getbalance/i', $message)):
    if(!checkPerm($uuid, getConfig('GENERAL_groupFunctionAdminBot'))):
        if(!BlackListIMExist($uuid)):
            BlackListIMAdd($uuid);
        else:
            BlackListIMInc($uuid);
        endif;
        SendMessageToAvatar($uuid, $avatar.', you have not permissions to manage the
bot.');
        exit();
    endif;
    $params = array(
        "command" => "getbalance",
        "group" => BOT_GROUP,
        "password" => BOT_PASSWORD,
    );
    $ret = SendToBot($params);
    $money = wasKeyValueGet("data", $ret);
    SendMessageToAvatar($uuid, $avatar.', I have L$ ' . $money . ' on my bank
account.');
    exit();
endif;
///////////////////

default_no_command($uuid, $avatar);
```

Let us explain:

The $uuid variable stores the uploaded avatar UUID.

The $message variable holds the forwarded message.

The $type variable holds the message type.

The default_no_command() function allows you to send a message to a given avatar that does not underst and the command. It will be used in several places.

Next we detect if an empty $type variable was not sent. It stores non - typical commands, e.g. implemented a few lines further error related to the fact that the bot and the interlocutor are not in

the same region. You can also go further here and add a message to be sent when the bot and the caller are too far apart. If the correct type of message is not detected, it sends a general message that it does not underst and the command.
If nothing is detected in the $type variable, we basically go to the main part of message processing.
For example, we took the detection of the words 'hi' or 'hello' in a message sent to the bot. If this happens, it gets the bot's display name and sends a reply back to the caller.
Next we have 3 examples where permissions are checked.

- blim - add - adds an avatar to the banned list,
- blim - del - removes an avatar from the banned list,
- .getbalance - shows how much money our bot has on the account.

The check is done by calling the checkPerm() function, which, on the basis of the avatar's uuid and the previously defined role, checks whether the avatar has been assigned to a given role.
If an avatar does not have the correct role in the group, it is added to the small banned counter, where after three such checks it is added to the permanent banned list. This suggestion is obviously too regoristic, you can skip it and just send a message that the avatar does not have permission to execute the command. If the avatar's authorization was successful, the rest of the code is executed. If, on the other hand, no relevant commands are found, a standard response is sent that the bot does not underst and the interlocutor.
Of course, this is some kind of a proposal, you will decide what to add, correct, etc.

How to develop:

```
php\serverbot.php 'action|serveIM|uuid|uuidOfAvatar|message|message|type|typeOfMessage'
```

We can add other things to our serverbot.php file to be processed by the bot.

5.4.3. Batch files

Now let's create some batch files in the root folder of the bot

start.bat

```
chdir /D %~dp0
@echo off

```

```
break>Cache\AgentCache.xml
break>Cache\CurrentGroupMembers.xml
break>Cache\CurrentGroups.xml
break>Cache\GroupCache.xml
break>Cache\InventoryCache.bin
break>Cache\MapFriend.xml
break>Cache\MuteCache.xml
break>Cache\RegionCache.xml
echo 0 > corrade_cron_inc.txt
echo 0 > corrade_attempt_inc.txt

del /Q State\Notifications.xml

sc.exe config Corrade start= auto
sc.exe start Corrade

exit 0
```

Runs the bot normally, saves empty cache files, deletes the Notifications.xml file with old notifications, sets the Corrade service to runtime, and starts the Corrade service

startWithCleanCache.bat

```
chdir /D %~dp0
@echo off

php -r "include \"php\config.php\"; include \"php\functions.php\"; deleteAllCache();"
START /MIN /WAIT start.bat
```

Deletes the cache file or deletes the cache data from the database (depending on the option) and runs the earlier start.bat file

stop.bat

```
chdir /D %~dp0

sc.exe stop Corrade
sc.exe config Corrade start= disabled
```

Stops the Corrade service, makes the service marked as disabled.

restart.bat

```
chdir /D %~dp0
@echo off

START /MIN /WAIT stop.bat
timeout /T 5 /NOBREAK
START /MIN /WAIT start.bat
```

The file restarts Corrade based on the recall of the two previous files.

If you have additional materials, you can use the bot_installer\corrade\bot_ai\php files by copying them to your bot's bot.

5.4.4. Multitasking in PHP

While working with my own bot, it turned out that I had to implement multitasking. If my artbot exceeded the predefined time, it would return to its former place. Before the implementation, it was difficult because the bot would st and like a salt pillar in the workplace. That's why I implemented multitasking in PHP, where the current script runs a separate script that checks from time to time if the script calling it has not completed its life cycle. This is very simple as the parent script tells the next script its process number (PID) which is used to check if this process from the original script still exists.

A separate task call might look like this:

Windows:

```
pclose( popen('START /B php ' . __DIR__ . '/innyplik.php -p ' . getmypid() . ' inne
parametry' > NUL', 'r' ) );
```

Linux:

```
pclose( popen('nohup php ' . __DIR__ . '/innyplik.php -p ' . getmypid() . ' inne
parametry' < /dev/null &', 'r' ) );
```

Short:

In Windows, a separate PHP file is run with START in the background, where we pass the current PID of the parent script along with other parameters to this file.

popen() and pclose() start the process and close the handle to it, since I am not interested in handling it.
On Linux, we use the nohup command, which runs the command even after the user logs out of the system.

In a separate PHP file, we need to consider how to receive the parameters.
There are a lot of projects to use here - I use ulrichsg/getopt - php which needs to be added to the Composer dependency.
Add after attaching vendor/autoload.php

```
use GetOpt\GetOpt;
use GetOpt\Option;
use GetOpt\Command;
use GetOpt\ArgumentException;
use GetOpt\ArgumentException\Missing;
```

Then we give the code snippet responsible for processing the arguments:

```
//PROCESSING FROM COMMANDLINE
$getOpt = new GetOpt();
$getOpt->addOptions([
  Option::create('p', 'pid', \GetOpt\GetOpt::REQUIRED_ARGUMENT)
->setDefaultValue(-1),
]);

try {
  $getOpt->process();
} catch (Exception $exception) {

}

$processId = $getOpt->getOption('pid');
```

We have yet to rethink how we will execute commands in the system itself.
By default, we can use a PHP command called shell_exec() or something similar.
However, I used symfony/process in my project, which we add to the Composer dependency.

Then we add:

```
use Symfony\Component\Process\Exception\ProcessFailedException;
use Symfony\Component\Process\Process;
use Symfony\Component\Process\PhpProcess;
use Symfony\Component\Process\PhpExecutableFinder;
```

and then the code:

```
while(true):
    if(ISREADY? == "1"):
        //[1] WHAT DO YOU WANT TO DO?
        break;
    else:
        if($processId > -1):
            $os_family = strtolower(PHP_OS_FAMILY);
            switch($os_family):
                case 'windows':
                    $process = Process::fromShellCommandline('powershell -Command "Get-
WmiObject Win32_Process -Filter \"processid = ' . $processId . '\" | measure | %
{ $_.Count }"');
                break;
                case 'linux':
                    $process = Process::fromShellCommandline('ps --no-headers -aux -q '
. $processId . ' | grep -v "grep" | wc -l');
                break;
            endswitch;
            try {
                $process->Run();
                if(trim($process->getOutput()) == "0"):
                    //[1] WHAT DO YOU WANT TO DO?
                    break;
                endif;
            } catch (ProcessFailedException $exception) {

            }
        endif;
    endif;
    sleep(10);
endwhile;
```

We will now look at this code. We loop every 10 seconds to check if the PHP parent script is still running. First, we check if the parent script has set ISREADS? after executing the entire script.

Through ISREADY? I mean here that this script sets a variable in the cache, which is then checked every few seconds by the child script to see if it contains the correct value. If so, it executes the rest of the code - for example, calling a specific function that will be given instead of the line: // [1] WHAT DO YOU WANT TO DO?. This option can of course be deleted if necessary. It is then checked to see if the parent's PID has been passed - a number from 0 upwards. If so, then the system on which the script is running is checked. If the detected system is Windows, the Symfony\Process powershell command is executed with arguments:

```
$process = Process::fromShellCommandline('powershell -Command "Get-WmiObject
Win32_Process -Filter \"processid = ' . $processId . '\" | measure | %{ $_.Count }"');
```

If the detected system is Linux, Symfony\Process executes the command:

```
$process = Process::fromShellCommandline('ps --no-headers -aux -q ' . $processId . ' |
grep -v "grep" | wc -l');
```

Regardless of the system, there is one goal - to check if the process with the given PID is still running in the system. If not, the action covered by the comment // [1] WHAT DO YOU WANT TO DO? Is performed, which is basically the same as the previous one // [1] WHAT DO YOU WANT TO DO?. and that's basically it. Then, after the while() loop, we can give the code that will be executed when the script receives the information that the parent script has finished its operation.

5.4.5. Second Life - creating prim and scripts

On the Second Life side, we create a prim, and in it a note called 'configuration', where we write the following content:

```
BOTID=UUID_BOT
GROUP=GROUP_NAME
PASSWORD=PASSWORD_GROUP
```

In place of:

- UUID_BOT - we put our bot's UUID,
- GROUP_NAME - the name of the group the bot is in,
- PASSWORD_GROUP - group password.

Then we create a new file with any name (preferably IM) with the following content (it depends whether we code with WAS or JSON).

The LSL script using the WAS function looks like this (Listings\mybot\was_IM.txt):

```
float distanceBetweenAvatars = 10.0;
key CORRADE;
string PASSWORD;
string GROUP;
string URL = "";
integer Debug = 1;
string TAG;
string val;
list lmessage;
string sjump = "";
key kAvatar;
string body;

integer line;
list tuples = [];
string type;
string agent;
string message;

key queryid;

list lprepare = [];

setDebug(string msg)
{
  if(Debug == 1)
  {
    llOwnerSay("["+ TAG +"] " + msg);
  }
}

sendPHP(list value){
    lprepare =     [
        "group", wasURLEscape(GROUP),
        "password", wasURLEscape(PASSWORD),
        "command", "execute",
        "file", "php",
        "parameter", "php/servebot.php " + llDumpList2String(value, "|"),
        "_script", TAG
    ];

    llOwnerSay(wasKeyValueEncode(lprepare));
}

integer StringMatch(list lwhatsearch, string sinwhere){
    sinwhere = llToLower(sinwhere);
    sinwhere = strReplace(sinwhere, "!", "");
    sinwhere = strReplace(sinwhere, "?", "");

    list lsinwhere = llParseString2List(sinwhere,[" "],[]);

    integer imatch = llGetListLength(lwhatsearch);
    integer imatched = 0;

    integer i=0;
    for(i=0; i<imatch; i++){

        list lwords = llParseString2List(llList2String(lwhatsearch,i),["|"],[]);

        integer j=0;
        for(j=0; j<llGetListLength(lwords); j++){
            if(llListFindList(lsinwhere, [llList2String(lwords,j)]) >= 0){
                imatched++;
            }
        }
    }
    if(imatch == imatched){
        return 1;
    }else{
        return 0;
    }
}

list wasCSVToList(string csv)
{
    list l = [];
    list s = [];
    string m = "";
    do
    {
        string a = llGetSubString(csv, 0, 0);
        csv = llDeleteSubString(csv, 0, 0);
        if(a == ",")
        {
            if(llList2String(s, -1) != "\"")
            {
                l += m;
                m = "";
                jump continue;
            }
            m += a;
            jump continue;
        }
        if(a == "\"" && llGetSubString(csv, 0, 0) == a)
        {
            m += a;
            csv = llDeleteSubString(csv, 0, 0);
            jump continue;
        }
        if(a == "\"")
        {
            if(llList2String(s, -1) != a)
            {
                s += a;
                jump continue;
            }
            s = llDeleteSubList(s, -1, -1);
            jump continue;
        }
        m += a;
        @continue;
    } while(csv != "");
    return l + m;
}

string strReplace(string str, string search, string replace) {
    return llDumpList2String(llParseStringKeepNulls((str = "") + str, [search], []),
replace);
}

string wasKeyValueGet(string k, string data)
```

```
{
  if(llStringLength(data) == 0) return "";
  if(llStringLength(k) == 0) return "";
  list a = llParseStringKeepNulls(data, ["&", "="], []);
  integer i = llListFindList(llList2ListStrided(a, 0, -1, 2), [ k ]);
  if(i != -1){
    string ret = llList2String(wasCSVToList(wasURLUnescape(llList2String(a, 2*i+1))),
0);
    ret = strReplace(ret, "\\r", "");
    ret = strReplace(ret, "\\n", "");
    ret = llStringTrim(ret, STRING_TRIM);

    return ret;
  }
  return "";
}

string wasKeyValueEncode(list data)
{
  list k = llList2ListStrided(data, 0, -1, 2);
  list v = llList2ListStrided(llDeleteSubList(data, 0, 0), 0, -1, 2);
  data = [];
  do
  {
    data += llList2String(k, 0) + "=" + llList2String(v, 0);
    k = llDeleteSubList(k, 0, 0);
    v = llDeleteSubList(v, 0, 0);
  } while(llGetListLength(k) != 0);
  return llDumpList2String(data, "&");
}

string wasURLEscape(string i)
{
  string o = "";
  do
  {
    string c = llGetSubString(i, 0, 0);
    i = llDeleteSubString(i, 0, 0);
    if(c == "") jump continue;
    if(c == " ")
    {
      o += "+";
      jump continue;
    }
    if(c == "\n")
    {
      o += "%0D" + llEscapeURL(c);
      jump continue;
    }
    o += llEscapeURL(c);
    @continue;
  } while(i != "");
  return o;
}

string wasURLUnescape(string i)
{
    return llUnescapeURL( llDumpList2String( llParseStringKeepNulls( llDumpList2String(
llParseStringKeepNulls( i, ["+"], [] ), " " ), ["%0D%0A"], [] ), "\n" ) );
}

default
{
    state_entry()
    {
        TAG = llGetScriptName();
        setDebug("Free memory: " + (string)llGetFreeMemory());
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
```

```
}

state ReadConfigurationNotecard
{
    state_entry()
    {
        if(llGetInventoryType("configuration") != INVENTORY_NOTECARD)
        {
            setDebug("Sorry, could not find a configuration inventory notecard.");
            return;
        }
        setDebug("Reading configuration file...");
        line = 0;
        llGetNotecardLine("configuration", line);
    }

    dataserver(key id, string data)
    {
        if(data == EOF)
        {
            if(llGetListLength(tuples) % 2 != 0)
            {
                setDebug("Error in configuration notecard.");
                return;
            }

            CORRADE = llList2Key(tuples, llListFindList(tuples, ["BOTID"])+1);

            if(CORRADE == NULL_KEY)
            {
                setDebug("Error in configuration notecard: BOT ID KEY");
                return;
            }

            GROUP = llList2String(tuples, llListFindList(tuples,["GROUP"])+1);

            if(GROUP == "")
            {
                setDebug("Error in configuration notecard: GROUP");
                return;
            }
            PASSWORD = llList2String(tuples, llListFindList(tuples, ["PASSWORD"])+1);

            if(PASSWORD == "")
            {
                setDebug("Error in configuration notecard: PASSWORD");
                return;
            }

            state url;
        }
        if(data == "") jump continue;
        integer i = llSubStringIndex(data, "#");
        if(i != -1) data = llDeleteSubString(data, i, -1);
        list o = llParseStringKeepNulls(data, ["="], []);
        string k = llDumpList2String( llParseStringKeepNulls( llStringTrim(
llList2String(o,0), STRING_TRIM), ["\""], []), "\"");
        string v = llDumpList2String( llParseStringKeepNulls( llStringTrim(
llList2String( o, 1 ), STRING_TRIM), ["\""], [] ), "\"");
        if(k == "" || v == "") jump continue;
        tuples += k;
        tuples += v;
        @continue;
        llGetNotecardLine("configuration", ++line);
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state url
```

```
{
    state_entry()
    {
        if ((GROUP == "BOTID") || (GROUP == "GROUP") || (GROUP == "PASSWORD")){
            llResetScript();
            return;
        }

        if ((PASSWORD == "BOTID") || (PASSWORD == "GROUP") || (PASSWORD == "PASSWORD"))
{
            llResetScript();
            return;
        }

        setDebug("Requesting URL...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        llRequestURL();
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(method != URL_REQUEST_GRANTED) return;
        URL = body;
        setDebug("Got URL...");
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state NotifyInstall {
    state_entry() {
        llOwnerSay("Binding to the IM notification...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        lprepare = [
            "group", wasURLEscape(GROUP),
            "password", wasURLEscape(PASSWORD),
            "callback", wasURLEscape(URL),
            "URL", wasURLEscape(URL),
            "command", "notify",
            "action", "set",
            "type", "message",
            "_script", TAG
        ];

        llOwnerSay(wasKeyValueEncode(lprepare));
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");

        if(wasKeyValueGet("success", body) != "True") {
                llOwnerSay("Failed to bind to the IM notification...");
                state preNotifyInstall;
        }
        llOwnerSay("IM notification installed...");
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state preNotifyInstall
{
    state_entry()
    {
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state sense {
    state_entry() {
        llOwnerSay("Listen IM...");
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        type = wasKeyValueGet("type", body);
        agent = wasKeyValueGet("agent", body);
        message = wasKeyValueGet("message", body);

        if(agent == NULL_KEY){
            return;
        }

        if(llStringLength(message) > 255){
            return;
        }

        lmessage = [] + llParseString2List(message,[" "],[]);

        if(llGetListLength(lmessage) == 0){
            return;
        }

        state CheckAvatar;
        return;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state CheckAvatar {
    state_entry() {
        queryid = llRequestAgentData(agent, DATA_ONLINE);
    }
    dataserver(key DataserverQueryId, string data)
    {
        if(DataserverQueryId == queryid)
        {
            if((integer)data == 1){
                state stateIM;
                return;
            }
        }
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state stateIM {
    state_entry() {
        list lbot = llGetObjectDetails(CORRADE, [OBJECT_POS]);
        list lagent = llGetObjectDetails(agent, [OBJECT_POS]);

        if(StringMatch(["where", "are", "you"], message)){
            sendPHP(["action", "serveIM", "type", "whereareyou", "uuid", agent]);
            state sense;
            return;

        if(llGetListLength(lagent) == 0){
            sendPHP(["action", "serveIM", "type", "error_nosameregion", "uuid",
agent]);
            state sense;
            return;
        }

//        vector vAgentPos = llList2Vector(lagent, 0);
//        float distance = llVecDist(vBotPos, vAgentPos);

//        if(distance > distanceBetweenAvatars){
//            sendPHP(["action", "serveIM", "type", "error_distance", "uuid", agent]);
//            state sense;
//            return;
//        }

        sendPHP(["action", "serveIM", "uuid", agent, "message", message]);

        state sense;
        return;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state preSense {
    state_entry() {
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

Script file using JSON functions (Listings\mybot\json_IM.txt):

```
float distanceBetweenAvatars = 10.0;
key CORRADE;
string PASSWORD;
string GROUP;
string URL = "";
integer Debug = 1;
string TAG;
string val;
list lmessage;
string sjump = "";
key kAvatar;
string body;

integer line;
list tuples = [];

string type;
string agent;
string message;

key queryid;

list lprepare = [];

setDebug(string msg)
{
  if(Debug == 1)
  {
    llOwnerSay("["+ TAG +"] " + msg);
  }
```

```
}

integer StringMatch(list lwhatsearch, string sinwhere){
    sinwhere = llToLower(sinwhere);
    sinwhere = strReplace(sinwhere, "!", "");
    sinwhere = strReplace(sinwhere, "?", "");

    list lsinwhere = llParseString2List(sinwhere,[" "],[]);

    integer imatch = llGetListLength(lwhatsearch);
    integer imatched = 0;

    integer i=0;
    for(i=0; i<imatch; i++){

        list lwords = llParseString2List(llList2String(lwhatsearch,i),["|"],[]);

        integer j=0;
        for(j=0; j<llGetListLength(lwords); j++){
            if(llListFindList(lsinwhere, [llList2String(lwords,j)]) >= 0){
                imatched++;
            }
        }
    }
    if(imatch == imatched){
        return 1;
    }else{
        return 0;
    }
}

sendPHP(list value){
    lprepare =     [
        "group", GROUP,
        "password", PASSWORD,
        "command", "execute",
        "file", "php",
        "parameter", "php/servebot.php " + llDumpList2String(value, "|"),
        "_script", TAG
    ];
    llOwnerSay(wasKeyValueToJSON(wasKeyValueEncode(lprepare), JSON_OBJECT));
}

list wasCSVToList(string csv)
{
    list l = [];
    list s = [];
    string m = "";
    do
    {
        string a = llGetSubString(csv, 0, 0);
        csv = llDeleteSubString(csv, 0, 0);
        if(a == ",")
        {
            if(llList2String(s, -1) != "\"")
            {
                l += m;
                m = "";
                jump continue;
            }
            m += a;
            jump continue;
        }
        if(a == "\"" && llGetSubString(csv, 0, 0) == a)
        {
            m += a;
            csv = llDeleteSubString(csv, 0, 0);
            jump continue;
        }
        if(a == "\"")
        {
            if(llList2String(s, -1) != a)
            {
                s += a;
                jump continue;
            }
            s = llDeleteSubList(s, -1, -1);
            jump continue;
        }
        m += a;
        @continue;
    } while(csv != "");
    return l + m;
}

string strReplace(string str, string search, string replace) {
    return llDumpList2String(llParseStringKeepNulls((str = "") + str, [search], []),
replace);
}

string wasKeyValueGet(string k, string data)
{
  if(llStringLength(data) == 0) return "";
  if(llStringLength(k) == 0) return "";
  list a = llParseStringKeepNulls(data, ["&", "="], []);
  integer i = llListFindList(llList2ListStrided(a, 0, -1, 2), [ k ]);
  if(i != -1){
    string ret = llList2String(wasCSVToList(wasURLUnescape(llList2String(a, 2*i+1))),
0);
    ret = strReplace(ret, "\\r", "");
    ret = strReplace(ret, "\\n", "");
    ret = llStringTrim(ret, STRING_TRIM);

    return ret;
  }
  return "";
}

string wasKeyValueToJSON(string kvp, string T) {
    list data = llParseString2List(kvp, ["&", "="], []);
    list temp = [];
    do {
        string k = llList2String(data, 0);
        string v = llList2String(data, 1);
        string o = "";
        o += "\"" + k + "\"";
        if(T == JSON_ARRAY) {
            o += ",";
            jump type;
        }
        o += ":";
        @type;
        if((float)v != 0 || v == "0") {
            o += v;
            jump dump;
        }
        o += "\"" + v + "\"";
        @dump;
        temp += o;
        data = llDeleteSubList(data, 0, 1);
    } while(llGetListLength(data) != 0);
    if(T == JSON_ARRAY)
        return "[" + llDumpList2String(temp, ",") + "]";
    if(T == JSON_OBJECT)
        return "{" + llDumpList2String(temp, ",") + "}";
    return JSON_INVALID;
}

string wasKeyValueEncode(list data)
{
  list k = llList2ListStrided(data, 0, -1, 2);
  list v = llList2ListStrided(llDeleteSubList(data, 0, 0), 0, -1, 2);
  data = [];
  do
  {
    data += llList2String(k, 0) + "=" + llList2String(v, 0);
    k = llDeleteSubList(k, 0, 0);
    v = llDeleteSubList(v, 0, 0);
  } while(llGetListLength(k) != 0);
  return llDumpList2String(data, "&");
}

string wasURLUnescape(string i)
{
    return llUnescapeURL( llDumpList2String( llParseStringKeepNulls( llDumpList2String(
llParseStringKeepNulls( i, ["+"], [] ), " " ), ["%0D%0A"], [] ), "\n" ) );
}

string wasJSONToKeyValueData(string JSON) {
    list output = [];
    list symbols = [];
    string data = "";
    integer level = 1;
    // parse JSON
    do {
        string c = llGetSubString(JSON, 0, 0);
        // we are at an object start
        if(c == "{") {
            if(level > 1) {
                data += c;
            }
            // increment the level
            ++level;
            // add the symbol
            symbols += c;
            // and continue
            jump continue;
        }
        // we are at an object end
        if(c == "}") {
            // decrement the level
            --level;
            if(level > 1) {
                data += c;
                output += data;
                data = "";
            }
            // pop symbols
            string s = llList2String(symbols, -1);
            symbols = llDeleteSubList(symbols, -1, -1);
            // got an object start
            if(s == "{") {
                // so continue
                jump continue;
            }
            // error
            return JSON_INVALID;
        }
        // we are at data end or data start
        if(c == "\"") {
            if(level > 2) {
                data += "\"";
                jump continue;
            }
            // pop symbols
            string s = llList2String(symbols, -1);
            symbols = llDeleteSubList(symbols, -1, -1);
            // if we are at data end
            if(s == "\"") {
                // add the data to the output
                output += data;
                // flush data
                data = "";
                // and continue
                jump continue;
            }
            // we are not a the end of data
            // add the symbol back
            symbols += s;
            // add the current character
            symbols += c;
            // and continue
            jump continue;
        }
        if(level > 2) {
            data += c;
            jump continue;
        }
        // pop symbols
        string s = llList2String(symbols, -1);
        symbols = llDeleteSubList(symbols, -1, -1);
        if(s == "\"") {
            data += c;
            symbols += s;
            jump continue;
        }
        symbols += s;
@continue;
        JSON = llDeleteSubString(JSON, 0, 0);
    } while(llStringLength(JSON) != 0);
    // now encode to key-value data
    list k = llList2ListStrided(output, 0, -1, 2);
    list v = llList2ListStrided(llDeleteSubList(output, 0, 0), 0, -1, 2);
    output = [];
    do {
        output += llList2String(k, 0) + "=" + llList2String(v, 0);
        k = llDeleteSubList(k, 0, 0);
        v = llDeleteSubList(v, 0, 0);
    } while(llGetListLength(k) != 0);
    return llDumpList2String(output, "&");
}

default
{
    state_entry()
    {
        TAG = llGetScriptName();
        setDebug("Free memory: " + (string)llGetFreeMemory());
        state ReadConfigurationNotecard;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state ReadConfigurationNotecard
{
    state_entry()
    {
        if(llGetInventoryType("configuration") != INVENTORY_NOTECARD)
        {
            setDebug("Sorry, could not find a configuration inventory notecard.");
            return;
        }
        setDebug("Reading configuration file...");
        line = 0;
        llGetNotecardLine("configuration", line);
    }

    dataserver(key id, string data)
    {
        if(data == EOF)
        {
            if(llGetListLength(tuples) % 2 != 0)
            {
                setDebug("Error in configuration notecard.");
                return;
            }

            CORRADE = llList2Key(tuples, llListFindList(tuples, ["BOTID"])+1);

            if(CORRADE == NULL_KEY)
            {
                setDebug("Error in configuration notecard: BOT ID KEY");
                return;
            }

            GROUP = llList2String(tuples, llListFindList(tuples,["GROUP"])+1);

            if(GROUP == "")
            {
                setDebug("Error in configuration notecard: GROUP");
                return;
            }

            PASSWORD = llList2String(tuples, llListFindList(tuples, ["PASSWORD"])+1);

            if(PASSWORD == "")
            {
                setDebug("Error in configuration notecard: PASSWORD");
                return;
            }

            state url;
        }
        if(data == "") jump continue;
        integer i = llSubStringIndex(data, "#");
        if(i != -1) data = llDeleteSubString(data, i, -1);
        list o = llParseStringKeepNulls(data, ["="], []);
        string k = llDumpList2String( llParseStringKeepNulls( llStringTrim(
llList2String(o,0), STRING_TRIM), ["\""], []), "\"");
        string v = llDumpList2String( llParseStringKeepNulls( llStringTrim(
llList2String( o, 1 ), STRING_TRIM), ["\""], [] ), "\"");
        if(k == "" || v == "") jump continue;
        tuples += k;
        tuples += v;
        @continue;
        llGetNotecardLine("configuration", ++line);
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state url
{
    state_entry()
    {
        if ((GROUP == "BOTID") || (GROUP == "GROUP") || (GROUP == "PASSWORD")){
            llResetScript();
            return;
        }

        if ((PASSWORD == "BOTID") || (PASSWORD == "GROUP") || (PASSWORD == "PASSWORD"))
{
            llResetScript();
            return;
        }
        setDebug("Requesting URL...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        llRequestURL();
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");
        if(method != URL_REQUEST_GRANTED) return;
        URL = body;
        setDebug("Got URL...");
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state NotifyInstall {
    state_entry() {
        llOwnerSay("Binding to the IM notification...");
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        lprepare = [
            "group", GROUP,
            "password", PASSWORD,
            "callback", URL,
            "URL", URL,
            "command", "notify",
            "action", "set",
            "type", "message",
            "_script", TAG
        ];

        llOwnerSay(wasKeyValueToJSON(wasKeyValueEncode(lprepare), JSON_OBJECT));
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");

        body = wasJSONToKeyValueData(body);

        if(wasKeyValueGet("success", body) != "True") {
                llOwnerSay("Failed to bind to the IM notification...");
                state preNotifyInstall;
        }
        llOwnerSay("IM notification installed...");
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

```
state preNotifyInstall
{
    state_entry()
    {
        state NotifyInstall;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state sense {
    state_entry() {
        llOwnerSay("Listen IM...");
    }

    http_request(key id, string method, string body)
    {
        llHTTPResponse(id, 200, "OK");

        body = wasJSONToKeyValueData(body);

        type = wasKeyValueGet("type", body);
        agent = wasKeyValueGet("agent", body);
        message = wasKeyValueGet("message", body);

        if(agent == NULL_KEY){
            return;
        }
        if(llStringLength(message) > 255){
            return;
        }

        lmessage = [] + llParseString2List(message,[" "],[]);

        if(llGetListLength(lmessage) == 0){
            return;
        }

        state CheckAvatar;
        return;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state CheckAvatar {
    state_entry() {
        queryid = llRequestAgentData(agent, DATA_ONLINE);
    }

    dataserver(key DataserverQueryId, string data)
    {
        if(DataserverQueryId == queryid)
        {
            if((integer)data == 1){
                state stateIM;
                return;
            }
        }
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state stateIM {
    state_entry() {

        list lbot = llGetObjectDetails(CORRADE, [OBJECT_POS]);
        list lagent = llGetObjectDetails(agent, [OBJECT_POS]);

        if(StringMatch(["where", "are", "you"], message)){
            sendPHP(["action", "serveIM", "type", "whereareyou", "uuid", agent]);
            state sense;
            return;
        }

        if(llGetListLength(lagent) == 0){
            sendPHP(["action", "serveIM", "type", "error_nosameregion", "uuid",
agent]);
            state sense;
            return;
        }

//         vector vBotPos = llList2Vector(lbot, 0);
//         vector vAgentPos = llList2Vector(lagent, 0);
//        float distance = llVecDist(vBotPos, vAgentPos);
//         if(distance > distanceBetweenAvatars){
```

```
//            sendPHP(["action", "serveIM", "type", "error_distance", "uuid", agent]);
//            state sense;
//            return;
//        }

        sendPHP(["action", "serveIM", "uuid", agent, "message", message]);

        state sense;
        return;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}

state preSense {
    state_entry() {
        state sense;
    }

    changed(integer change)
    {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START))
        {
            llResetScript();
        }
    }

    on_rez(integer start_param)
    {
        llResetScript();
    }
}
```

Now let's explain the different parts:

- sendPHP() - sends a command set to php/serverbot.php via bot, arguments are separated by a vertical bar.
- StringMatch() - looks for one text string in the second text string.
- strReplace() - zamends one string into another string in the given text.
- TAG = llGetScriptName(); - assigns a script name to a variable, for analytical purposes.
- llSetTimerEvent() i timer() - the first sets the time after which the timer fires (I use a short period here), while the second is a time block that executes every specified number of seconds - the default is every 60 seconds, because Corrade takes the maximum time to execute the command at home on average . If the code is successful, we can exit this block to the next state.
- llOwnerSay() - we send everything directly to the bot as prim is owned by the bot and we edit prim and its scripts as a bot; if the prim is placed on the plot, it is recommended to use llInstantMessage(), and before sending the command, it is better to put the status checking if the bot is available (using a timer) in Second Life (check the CheckAvatar status) and until it is there, we do not send any commands.

```
state CheckBotAvatar {
    state_entry() {
        llSetTimerEvent(0.5);
    }

    timer(){
        llSetTimerEvent(60);
        queryid = llRequestAgentData(CORRADE, DATA_ONLINE);
    }

    dataserver(key DataserverQueryId, string data) {
        if(DataserverQueryId == queryid) {
            if((integer)data == 1){
                state NastepnyStan;
            }
        }
    }
    changed(integer change) {
        if(change & (CHANGED_OWNER | CHANGED_REGION | CHANGED_REGION_START)) {
            llResetScript();
        }
    }

    on_rez(integer start_param) {
        llResetScript();
    }
}
```

Next, of course, is reading the 'configuration' note, where you can find information about: our bot's UUID, group name, group password. After loading the configuration, you go to the 'url' state, where an attempt is made to download the URL from Second Life to which notifications will be sent.

Another very important step is to install the message notification itself, which allows you to receive private messages sent to the bot. Everything is sent to the prim owner, the bot, by the llOwnerSay command. If it is not possible to set the notification (an error is returned), the request to the bot to install the notification is sent again.

When all this is successful, it is listened to - if the bot receives any private message, it will be forwarded to the script via a notification that contains the return URL.

If that's the case, my sample script first checks the following:

- whether the agent is not a null value, ie it does not accept only zeros,
- that the length of the message is less than 255 characters,
- if the message converted to a list has no zero items.

If each question is false, the following goes to check if the sender avatar is still available on Second Life, because if there is no sender in the virtual world, it becomes superfluous to reply to him when he is offline.

If our interlocutor is available, the main part of the code begins:

- downloading bot coordinates,
- downloading avatar - sender coordinates,
- checking if the message contains the following keywords: where, are and you, if so, it sends a command to the PHP file of the bot service by the bot as an intermediary with the action: serveIM, type: whereareyou and uuid as the UUID of the sender's avatar,

- checking if the sender's avatar is in the same region as the bot (non - zero list element),
- checking if the distance between the bot and the avatar is not greater than a given number (commented out option),
- sending an avatar message to PHP support along with the sender avatar UUID.

The sample code is divided into two parts upon receipt of the message.
The first part checks the message globally, e.g. when the sender asks for a bot location. Most of the conditions mentioned above are not checked in this section.
The second part is checking certain conditions and, if everything is correct, send it to PHP for further processing.

If we run the script, after a few minutes we will be able to talk to our bot via private messages.

6. Puzzle Grid for Arranging Objects (PosGrid)

Manually computing coordinates for objects can be laborious, and with a complex script, you may later encounter problems with its code analysis.

In Second Life, you can set prim rows with unique names, but this idea quickly fails, because the limit of prims to be used on a plot or in a region will expire even faster.

A much better method I invented is the logical division of a given area (e.g. an object) into vertical and horizontal lines (hereinafter referred to as a grid), and then determining the position of a given point as the intersection of these two lines. The more vertical and horizontal lines, the more precise the position of the selected point.

A detailed explanation based on examples is provided below.

Data needed to calculate the coordinate:

- O - X, Y coordinates of a point that is the center of an object in Second Life - usually these are the coordinates of that object,
- S - object size,
- R - rotation factor (X, Y) depending on the direction of the compass rose,
  - North (N) - 0,1,
  - East (E) - 1,0,
  - South (S) - 0, - 1,
  - West (W) - - 1,0.
- L - the number of vertical and horizontal lines (X, Y) that divide the area into identical pieces,
- E' - the (X, Y) coordinates of the end point where the object should be positioned. Coordinates are counted from 0.

Assumptions:

- X is a floating point number with 5 decimal places, values: 0...255,
- Y is a floating point number with 5 decimal places, values: 0...255,
- Z is a floating point number with 5 decimal places, values: 0...4,096,
- O is a floating point number with 5 decimal places,
- S is a floating point number with 5 decimal places,
- R is an integer with the following values: - 1, 0, 1,
- L is an integer that ranges from 1 to infinity,
- If L is divisible by 2, then increase L by 1,
- E' is an integer that ranges from 0 to infinity,
- E' cannot be greater than the number of L lines,
- If E' is less than 0 then E' = 0,
- If E' is greater than the number of lines L, then E' = number of lines - 1,
- If L = 1 then E' = 0,
- In case of incorrect or incomplete data, E' = O,

To obtain a universal formula for calculating the coordinates of the ending point E, let us consider 4 situations compatible with the compass rose, taking only the main directions: North, South, East and West.

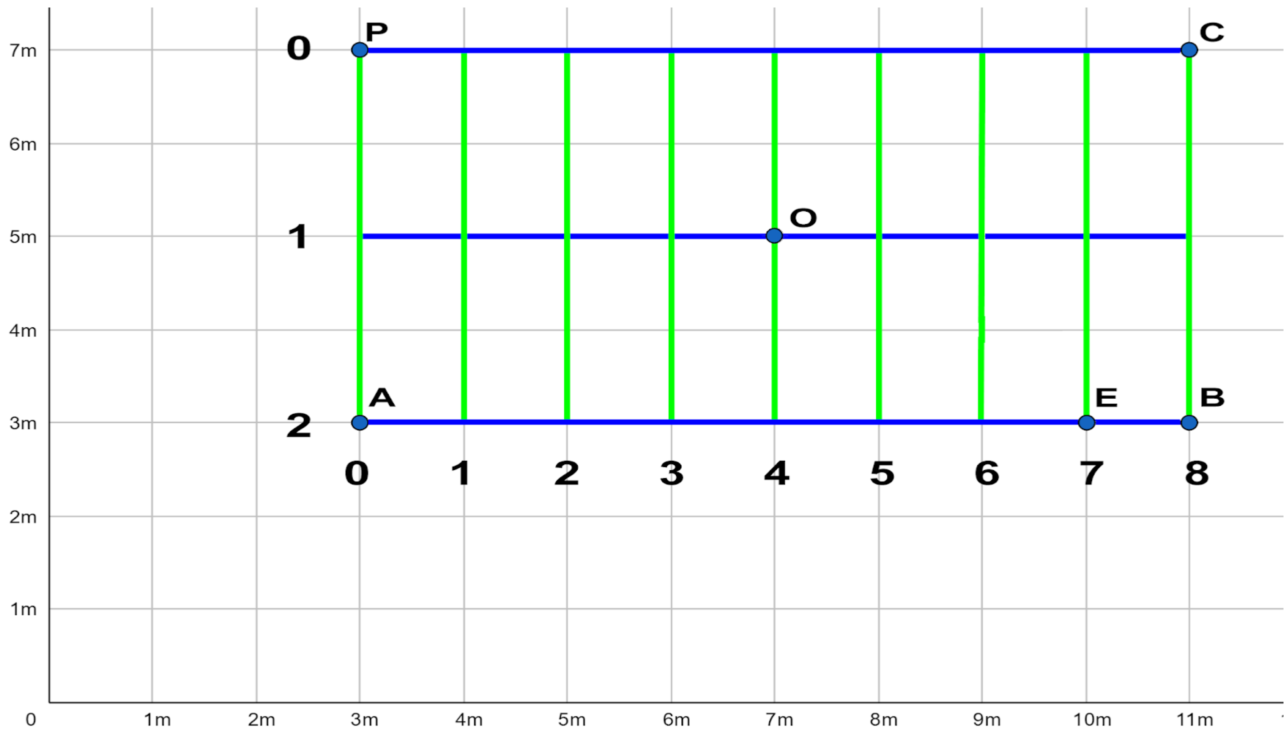
Data common to each picture below:

- O = (7,5),
- S = (8,4),
- R - depends on the rotation of the object that we share with virtual lines, it is given in front of the image,
- L = (9,3),
- E' = (7,2).

We have to calculate the point P, which is the upper left corner of the object, which would also be the starting point for the calculation of the end point E, and the width and length of a single piece.

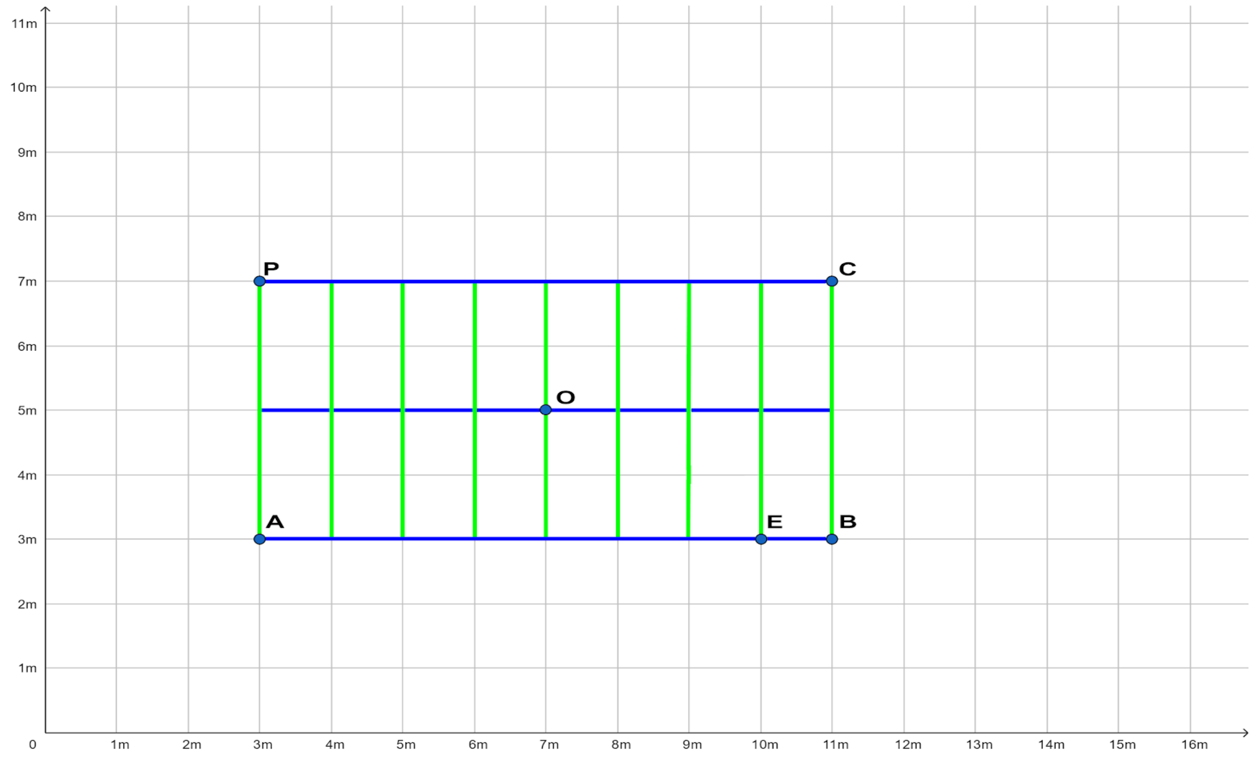
The method of calculating the E' coordinates is shown in the drawing below.

The index is counted from 0, e.g. for point E = (7.3) E' is (7.2), for point O = (4.5) E' it is (4.1), etc.



Picture 6.1 - Way of counting point E'

North (N), R = (0,1), object rotation by 0°



Picture 6.2 - Way of calculating point E for the north direction

First, we calculate the width and length of a single piece in this part. For the calculation you will need the size of the object S and the number of vertical and horizontal lines L.

If we divide the object vertically (green line) from 3 to 11 m, every 1 m, we get 16 pieces (8 pieces × 2 rows).

If we divide the object horizontally (blue line) from 3 to 7 m every 2 m, we get 16 pieces (2 pieces × 8 rows).

It can be concluded that the number of pieces is equal when the number of lines is less by 1.

$$S` = \frac{S}{L-1}$$

The above pattern is universal and determines the size of a single piece.

Check:

$$S`_x = \frac{S_x}{L_x - 1} = \frac{8}{9-1} = \frac{8}{8} = 1$$

$$S`_y = \frac{S_y}{L_y - 1} = \frac{4}{3-1} = \frac{4}{2} = 2$$

Now let's calculate the coordinates of the point P, i.e. the point from which we will count the coordinates of the point E.

$$P_x = O_x - \frac{S_x}{2}$$

$$P_y = O_y + \frac{S_y}{2}$$

Check:

$$P_x = O_x - \frac{S_x}{2} = 7 - \frac{8}{2} = 7 - 4 = 3$$

$$P_y = O_y + \frac{S_y}{2} = 5 + \frac{4}{2} = 5 + 2 = 7$$

Let us now calculate the coordinates of the point E we are interested in. To calculate E we will need P, previously calculated lengths and widths of a single piece, S ' and E.

$$E_x = P_x + S`_x \cdot E`_x$$

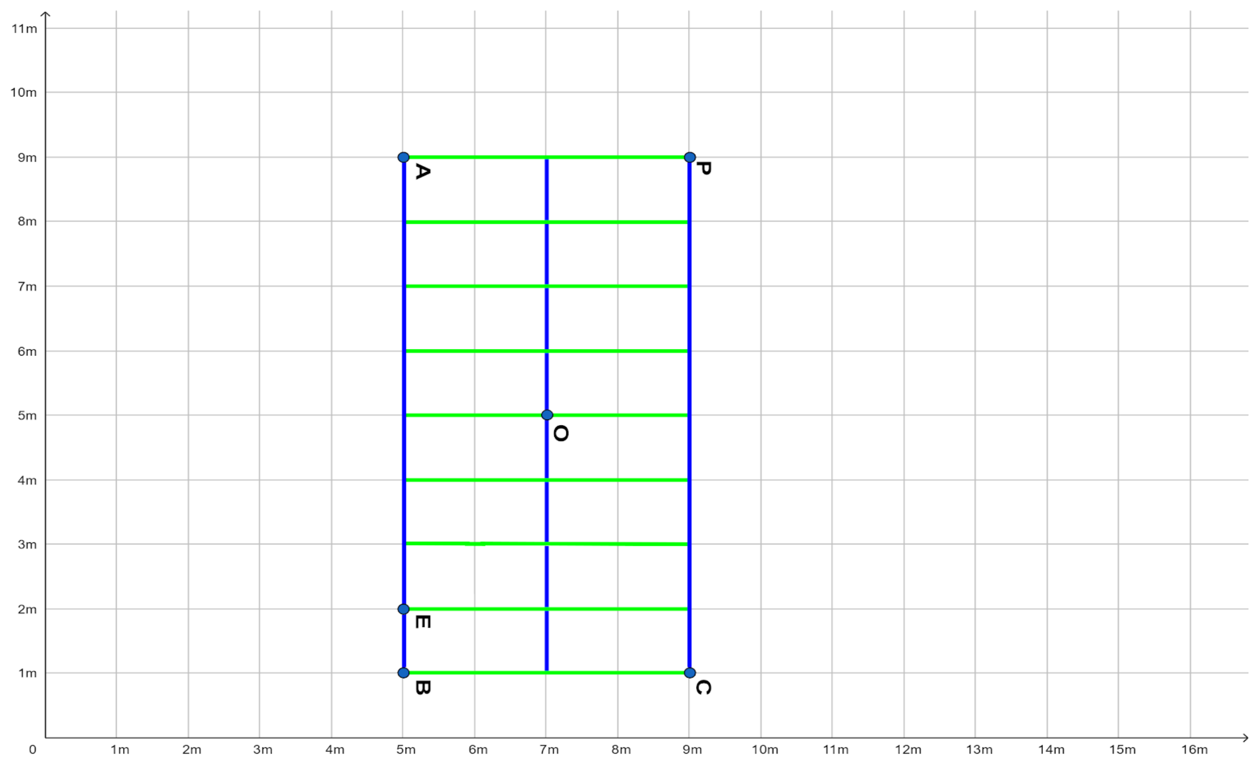
$$E_y = P_y - S`_y \cdot E`_y$$

Check:

$$E_x = P_x + S`_x \cdot E`_x = 3 + 1 \cdot 7 = 3 + 7 = 10$$

$$E_y = P_y - S`_y \cdot E`_y = 7 - 2 \cdot 2 = 7 - 4 = 3$$

East (E), R = (1.0), object rotation 90°



Picture 6.3 - The way of calculating the point E for the eastern direction

Now let's calculate the coordinates of the point P for the eastward rotation.

$$P_x=O_x+\frac{S_y}{2}$$

$$P_y=O_y+\frac{S_x}{2}$$

Check:

$$P_x=O_x+\frac{S_y}{2}=7+\frac{4}{2}=7+2=9$$

$$P_y=O_y+\frac{S_x}{2}=5+\frac{8}{2}=5+4=9$$

Now let's recalculate point E for the east direction.

$$E_x=P_x-S`_y\cdot E`_y$$

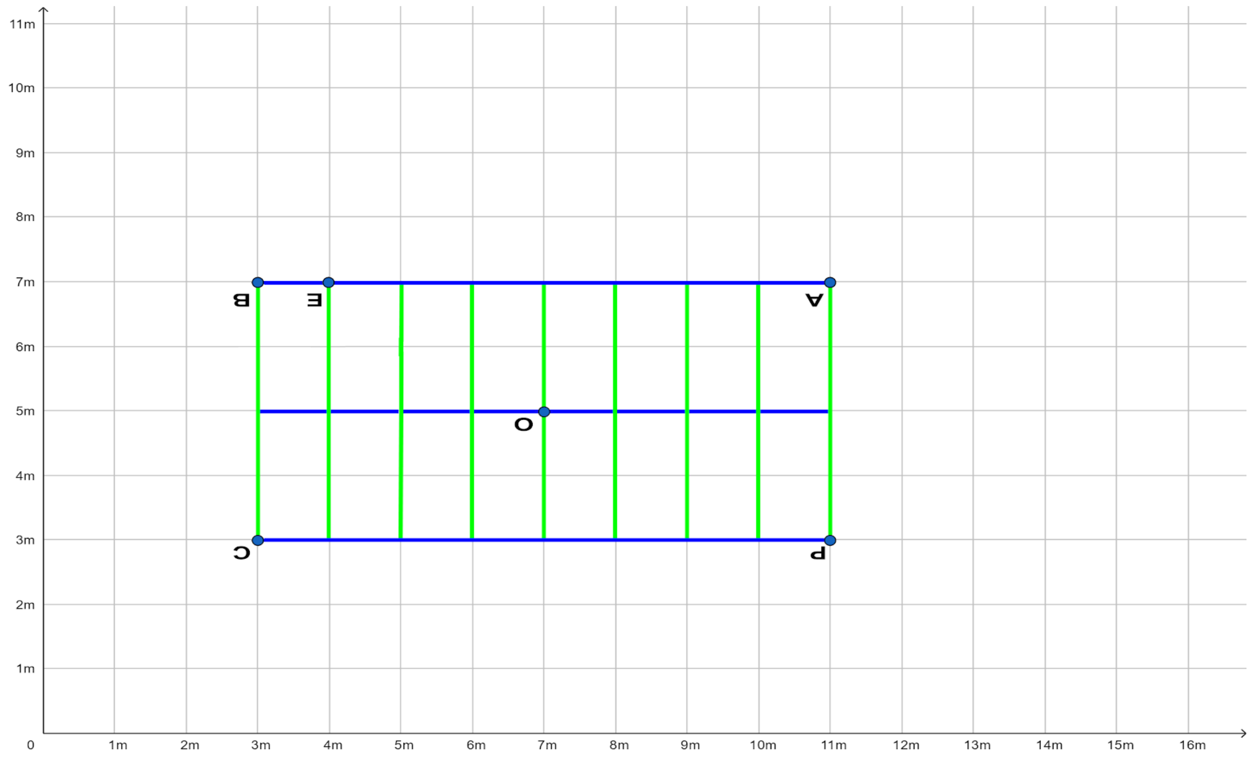
$$E_y=P_y-S`_x\cdot E`_x$$

Check:

$$E_x=P_x-S`_y\cdot E`_y=9-2\cdot 2=9-4=5$$

$$E_y = P_y - S`_x \cdot E`_x = 9 - 1 \cdot 7 = 9 - 7 = 2$$

South (S), R = (0, - 1), rotate the object 180°



Picture 6.4 - The way of calculating the point E for the south direction

Now let's calculate the coordinates of the point P for the south rotation.

$$P_x = O_x + \frac{S_x}{2}$$

$$P_y = O_y - \frac{S_y}{2}$$

Check:

$$P_x = O_x + \frac{S_x}{2} = 7 + \frac{8}{2} = 7 + 4 = 11$$

$$P_y = O_y - \frac{S_y}{2} = 5 - \frac{4}{2} = 5 - 2 = 3$$

Now let's calculate the coordinates of E.

$E_x = P_x - S`_x \cdot E`_x$

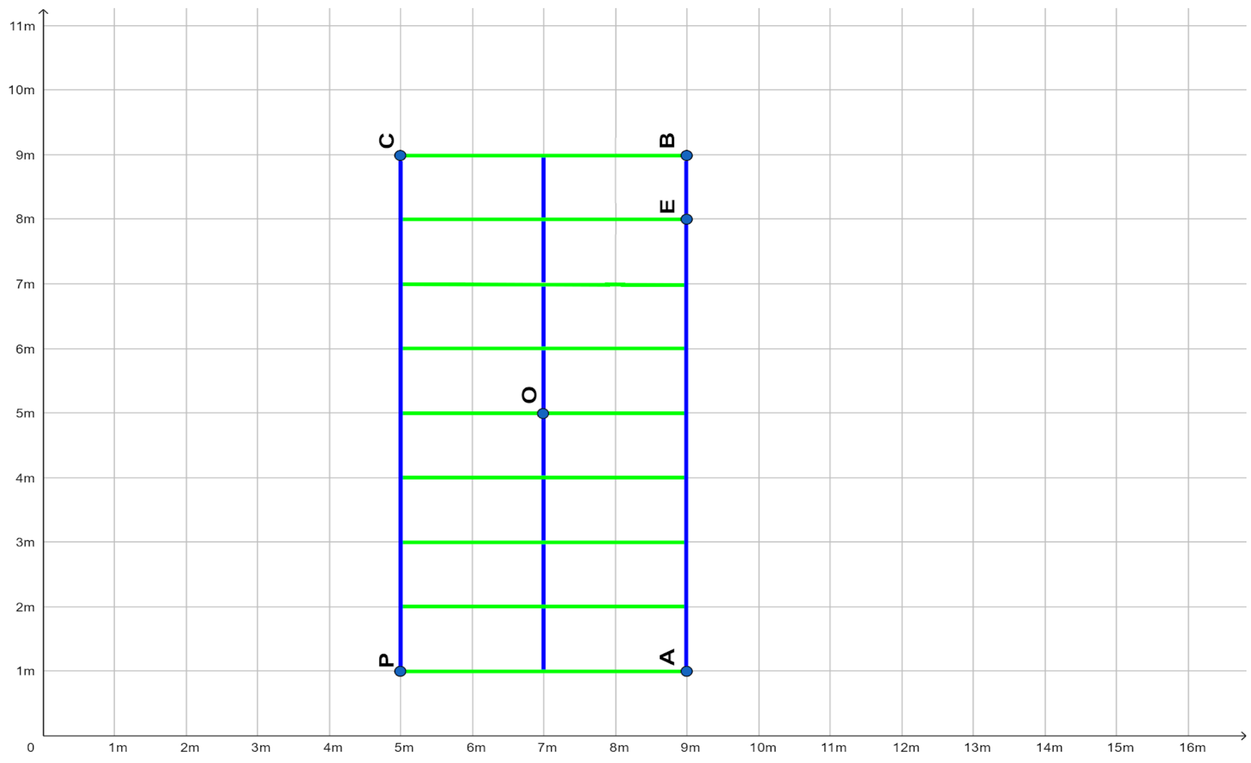
$E_x = P_x - S`_x \cdot E`_x$

Check:

$E_x = P_x - S`_x \cdot E`_x = 11 - 1 \cdot 7 = 11 - 7 = 4$

$E_x = P_x - S`_x \cdot E`_x = 3 + 2 \cdot 2 = 3 + 4 = 7$

West (W), R = (- 1.0), rotation of the object by 270°



Picture 6.5 - The way of calculating the point E for the south direction

Now let's calculate the coordinates of the point P for the west rotation.

$$P_x = O_x - \frac{S_y}{2}$$

$$P_y = O_y - \frac{S_x}{2}$$

Check:

$$P_x = O_x - \frac{S_y}{2} = 7 - \frac{4}{2} = 7 - 2 = 5$$

$$P_y=O_y-\frac{S_x}{2}=5-\frac{8}{2}=5-4=1$$

Now let's recalculate point E for the west direction.

$$E_x=P_x+S`_y\cdot E`_y$$

$$E_y=P_y+S`_x\cdot E`_x$$

Check:

$$E_x=P_x+S`_y\cdot E`_y=5+2\cdot 2=5+4=9$$

$$E_y=P_y+S`_x\cdot E`_x=1+1\cdot 7=1+7=8$$

From the above partial equations, taking into account the rotation factors, we obtain a universal formula:

$$E_x=O_x-(\frac{S_x}{2}\cdot R_y)+(\frac{S_x}{L_x-1}\cdot E`_x\cdot R_y)+(\frac{S_y}{2}\cdot R_x)-(\frac{S_y}{L_y-1}\cdot E`_y\cdot R_x)$$

$$E_y=O_y+(\frac{S_y}{2}\cdot R_y)-(\frac{S_y}{L_y-1}\cdot E`_y\cdot R_y)+(\frac{S_x}{2}\cdot R_x)-(\frac{S_x}{L_x-1}\cdot E`_x\cdot R_x)$$

Check:

Common data (rewritten specially from the previous page):

- O = (7,5),
- S = (8,4),
- R - object rotation factor,
- L = (9,3),
- E' = (7,2).

N	$E_x=7-(\frac{8}{2}\cdot 1)+(\frac{8}{9-1}\cdot 7\cdot 1)+(\frac{4}{2}\cdot 0)-(\frac{4}{3-1}\cdot 2\cdot 0)=7-4+7+0+0=3+7=10$ $E_y=5+(\frac{4}{2}\cdot 1)-(\frac{4}{3-1}\cdot 2\cdot 1)+(\frac{8}{2}\cdot 0)-(\frac{8}{9-1}\cdot 7\cdot 0)=5+2-4+0-0=7-4=3$
S	$E_x=7-(\frac{8}{2}\cdot -1)+(\frac{8}{9-1}\cdot 7\cdot -1)+(\frac{4}{2}\cdot 0)-(\frac{4}{3-1}\cdot 2\cdot 0)=7+4-7+0-0=11-7=4$ $E_y=5+(\frac{4}{2}\cdot -1)-(\frac{4}{3-1}\cdot 2\cdot -1)+(\frac{8}{2}\cdot 0)-(\frac{8}{9-1}\cdot 7\cdot 0)=5-2+4+0-0=3+4=7$
E	$E_x=7-(\frac{8}{2}\cdot 0)+(\frac{8}{9-1}\cdot 7\cdot 0)+(\frac{4}{2}\cdot 1)-(\frac{4}{3-1}\cdot 2\cdot 1)=7-0+0+2-4=7+2-4=9-4=5$

	$E_y=5+(\frac{4}{2}\cdot 0)-(\frac{4}{3-1}\cdot 2\cdot 0)+(\frac{8}{2}\cdot 1)-(\frac{8}{9-1}\cdot 7\cdot 1)=5+0-0+4-7=5+4-7=9-7=2$
W	$E_x=7-(\frac{8}{2}\cdot 0)+(\frac{8}{9-1}\cdot 7\cdot 0)+(\frac{4}{2}\cdot -1)-(\frac{4}{3-1}\cdot 2\cdot -1)=7-0+0-2+4=7-2+4=5+4=9$ $E_y=5+(\frac{4}{2}\cdot 0)-(\frac{4}{3-1}\cdot 2\cdot 0)+(\frac{8}{2}\cdot -1)-(\frac{8}{9-1}\cdot 7\cdot -1)=5+0-0-4+7=5-4+7=1+7=8$

Table 6 - listing of all endpoint calculations

The PHP function implementing the above formula along with the assumptions is presented below.

```php
function posGrid($beginPoint, $size, $ratio, $lines, $endPoint){
	$returnPoint = new classPoint();

	if($lines->getPointX() % 2 == 0):
		$lines->setPointX($lines->getPointX() + 1);
	endif;

	if($lines->getPointY() % 2 == 0):
		$lines->setPointY($lines->getPointY() + 1);
	endif;

	if($endPoint->getPointX() < 0):
		$endPoint->setPointX(0);
	endif;

	if($endPoint->getPointY() < 0):
		$endPoint->setPointY(0);
	endif;

	if($endPoint->getPointX() > $lines->getPointX() - 1):
		$endPoint->setPointX($lines->getPointX() - 1);
	endif;

	if($endPoint->getPointY() > $lines->getPointY() - 1):
		$endPoint->setPointY($lines->getPointY() - 1);
	endif;

	$x = $beginPoint->getPointX() - (($size->getPointX() / 2) * $ratio->getPointY())
+ (($size->getPointX() / ($lines->getPointX() - 1)) * $endPoint->getPointX() * $ratio-
>getPointY()) + (($size->getPointY() / 2) * $ratio->getPointX()) - (($size->getPointY()
/ ($lines->getPointY() - 1)) * $endPoint->getPointY() * $ratio->getPointX());
$y = $beginPoint->getPointY() + (($size->getPointY() / 2) * $ratio->getPointY()) -
(($size->getPointY() / ($lines->getPointY() - 1)) * $endPoint->getPointY() * $ratio-
>getPointY()) + (($size->getPointX() / 2) * $ratio->getPointX()) - (($size->getPointX()
/ ($lines->getPointX() - 1)) * $endPoint->getPointX() * $ratio->getPointX());

      if($x > 255):
            $x = 255;
      endif;

      if($y > 255):
            $y = 255;
      endif;

      if(!in_array($ratio->getPointX(), array(-1, 0, 1))):
            $x = $beginPoint->getPointX();
      endif;

      if(!in_array($ratio->getPointY(), array(-1, 0, 1))):
            $y = $beginPoint->getPointY();
      endif;

      if(($lines->getPointX() <= 1) && ($lines->getPointY() <= 1)):
            $x = $beginPoint->getPointX();
            $y = $beginPoint->getPointY();
      endif;

      $returnPoint->setPointX(number_format($x, 5, '.', ''));
      $returnPoint->setPointY(number_format($y, 5, '.', ''));
      $returnPoint->setPointZ(number_format($beginPoint->getPointZ(), 5, '.', ''));
      return $returnPoint;
}
```

Text 10 - function that returns the coordinates of a point based on the object alignment logic grid

To use this function, you must attach the class.point.php file beforehand.

```
<?php
class classPoint {
      private $X;
      private $Y;
      private $Z;

      public function setPoint($x, $y, $z) {
            $this->X = $x;
            $this->Y = $y;
            $this->Z = $z;
      }

      public function setPointX($x){
            $this->X = $x;
      }

      public function getPointX(){
            return $this->X;
      }
}
```

Text 11 - source code of class.point.php [bot_installer\corrade\bot_ai\php\class.point.php]

Now we define our code for the calculation, we use the sample data mentioned in this chapter:

- $beginPoint - the coordinates of the center of the object (O) - example: (7,5),
- $size - object size (S) - example: (8,4),
- $ratio - turnover factor (R) - example: (1.0),
- $lines - number of lines (L) - example: (9,3),
- $endPoint - end point (E') - example: (7,1).

```
$beginPoint = new classPoint();
$beginPoint->setPoint(7, 5, 100);

$size = new classPoint();
$size->setPoint(8, 4, 0);

$ratio = new classPoint();
$ratio->setPoint(1, 0, 0);

$lines = new classPoint();
$lines->setPoint(9, 3, 0);

$endPoint = new classPoint();
```

```
$endPoint->setPoint(7, 2, 0);

$grid = posGrid($beginPoint, $size, $ratio, $lines, $endPoint);

echo 'Result: X:' . $grid->getPointX() . ', Y:' . $grid->getPointY() . ',Z:' . $grid-
>getPointZ() . PHP_EOL;
```

Text 12 - attempting to calculate the endpoint using PosGrid [bot_installer\corrade\bot_ai\php\functions.php]

What will give us as a result:

```
Result: X:5.00000, Y:2.00000,Z:100.00000
```

Text 13 - the result of the PosGrid function based on previously defined data

7. Frequently Asked Questions

Most of the questions and answers below can be found on the software manufacturer's website.

Please contact me, if you have a question about Corrade.

1. Does Corrade require hosting?

Yes, Corrade requires hosting as a program. You can rent a VPS or allocate a free computer to it so that you don't have to pay for the rental every month.

2. What can I do with Corrade?

Corrade comes pre-programmed with all browser capabilities for Second Life. With Corrade, you can, for example, automatically accept payments for renting lands or sell items.

Corrade's possibilities are wide, you just need to read the documentation.

3. Is Corrade free?

Yes, Corrade is free. It is licensed under its own license which is not restrictive. In fact, the manufacturer asks you to tag it when using it in other projects and not reverse engineer it on the program's binaries.

For more information on the license, visit grimore.org[21] and in the chapter: **License**.

4. If Corrade is free, is I a "product"?

I can't say it. The manufacturer ensures that it does not collect any data about the user for monetization purposes. If you want to make a small contribution to the development of the project,

21 License WAS PC & OD 1.0 - https://grimore.org/licenses/was - pc - od

you can, for example, enable the sending of diagnostic data in Nucleus, advertise the project, write scripts and make them available to a wider audience.

5. **Is Corrade OpenSource?**

No, the project's source code is closed to the public. However, templates, PHP or LSL scripts are open and you can safely modify them.

6. **Which platforms can Corrade run on?**

Currently Corrade has been built on the basis of .NET Core and can be run on Linux, Windows, MacOSX, platforms (x64, ARM) that allow .NET Core to be installed.

7. **How to start working and programming with Corrade?**

You need to download Corrade and configure it. When it comes to programming, you need to know the basics of Second Life LSL. It is not a difficult language to learn. On the Second Life website you will find a rich source of knowledge about this language[22].

On the Corrade website and in the manufacturer's store Corrade (Second Life Marketplace)[23] you will find free examples of using Corrade.

8. **How does Corrade manage access control?**

Corrade manages the access control assigned to the group. In other words, you specify groups with passwords that will be allowed to execute commands on the Corrade. Corrade doesn't care what avatar triggers commands. This has its advantage, because you can create a script in Second Life with the appropriate permissions (e.g. no access to the file preview), sell it, then the buyer will only be able to modify the bot configuration note.

9. **Why does Corrade use groups instead of avatars for access or permissions?**

22 LSL Portal - http://wiki.secondlife.com/wiki/LSL_Portal
23 Corrade Second Life Marketplace - https://marketplace.secondlife.com/stores/165275

The groups in Second Life are at the lowest level in terms of entitlement.

Grid operators have the greatest powers in Second Life, wealth managers have only l and management powers, and the powers of ordinary avatars are limited to their resources. Most of the actions in Second Life, i.e. sharing the l and with someone else, streaming music on the device, allowing the object rezz and other simple operations i.e. setting the landmark are limited by the permissions of the group that is assigned to the land.

To ensure successful execution of commands, all commands must ask the grid for permissions because in the absence of a GUI, Corrade will not be able to report errors and attempts to see if the command was successful. Most of the checks performed by Corrade end up asking about the group permission.

10. Why is Corrade unable to restart itself?

Because there are tools that better cope with restarting a given program, taking into account conditions such as network condition and load, CPU load, traffic, etc., and on this basis decides whether to restart the program.

11. I downloaded Corrade, set it up, but in the console I see a message that there is a problem with logging in.

Usually there is a problem with the wrong password provided when setting up the bot.

```
login failed : key
```

Open the Configuration.xml file and find 'Password'.

Also open some MD5 coding tool (google: md5 online).

```
<Password>$1$842e4818f8ede223c9b920d4f7425c9b</Password>
```

Delete the content between <Password> and </Password>.

Add $1$ followed by the MD5 encoded password.

If your password is more than 16 characters long, change it to 6 - 16 characters long, as only such passwords are accepted by Second Life.

12. What does "presence" mean?

In the console and program logs, you can see the error:

```
login failed : presence
```

This means that the avatar is already logged in to Second Life, you may have logged in manually using a browser, or Corrade has previously logged out of the world incorrectly.

Attempting to log in to an account where the avatar is already logged in will cause the previous logon to be disconnected, and the user will see a message that they have logged out of Second Life.

Corrade will try to reconnect with this error.

However, if you see an error:

```
login failed : ban
```

It means there is a temporary login lock, in which case wait a few minutes and try to login again.

13. Does Corrade work on OpenSim?

Corrade is basically compatible with Second Life protocols as they are well described. Corrade can work on the OpenSim mesh, but the manufacturer does not guarantee that all commands will work properly, because OpenSim is not compatible with Second Life protocols and is heavily burdened with bugs.

14. I am sending a URL through Corrade which contains spaces, but Corrade breaks it

This happens most often for SLURL, e.g. secondlife://Some%20Simulator/32/13/34, where Corrade converts the space to% 20 such that the link is no longer recognizable by the browser as a link.

Send link as: secondlife://Some%2520Simulator/32/13/34, which will already run (%2520 will be converted to %20).

15. I am trying to add a group to a Corrade configuration that contains special characters.

In XML entries, some characters are special and must be replaced with their equivalents.

Sign	Equivalent
"	"
'	&apos;
<	<
>	>
&	&

Table 7: Special characters in HTML

For example, a group named Wizardry & Steamworks must be converted into the configuration file: Wizardry &amp; Steamworks

16. When I send commands from LSL to a hidden group, I always get the message that access is denied or that the group was not found.

Instead of assigning a group name to a group parameter, assign a group UUID to it. The reason is that Corrade cannot find the group displayed in Directory Services.

17. Corrade and Proxy

Corrade has little experience with proxy or Tor servers.

It is best not to set up a proxy on a computer running Corrade.

The most common symptoms of using a proxy are:

- the wardrobe does not fully load, and an attempt to point to an item in the wardrobe returns an error that the item does not exist,
- Corrade does not know what groups it is in,
- some notifications often fail.

So as you can see, it's best not to use proxy and turn it off on your system.

18. My bot has a problem with rezzing (or appears as a cloud)

There are two known possibilities for the bot to appear as clouds:

- Corrade carries more than one attachment per slot; Corrade supports one attachment per slot,
- Corrade does not wear: skin, eyes, hair, or shape; in Second Life, one body part must have some accessory; Browsers tend to ban avatars that do not wear body parts.

19. Autopilot with altitude

Second Life contains a long-standing bug that causes the agent to behave arbitrarily at the height specified for the autopilot command. If you want the bot to fly to a location instead of walking around, consider using the 'flyto' command, which is much more versatile, or a combination of both:

- Use the autopilot to navigate on surfaces/ground,
- Use flyto to make Corrade fly to the target.

20. Why are some wearables not reported by Corrade?

This is a problem in the libopenmetaverse library that Corrade uses where wearing some accessories (e.g. a skirt) is not shown on the bot. This bug has been reported several times, fixed and reported again.

This causes the commands to carry items to behave inconsistently. When these errors are fixed in the library, the commands should now work properly.

21. The bot drops off vehicles when crossing regions

This error occurs in all known bot softwares. Nobody knows what causes this error. Most likely it is caused in libopenmetaverse library, it has been reported several times with no remedial effects.

22. Do i need to update?

Corrade receives frequent updates to fix the reported bugs.

Before downloading, you can read the text file, which gives a short description of what has changed in a given version.

From my personal experience, you should be happy with the development of the program, etc., however, I recommend that you be careful with new versions of the program, as it may turn out that the new version breaks something in the bot functionality. In this case, it is best to take a screenshot, report the error to the manufacturer on the website.

I personally have two bots - one operating - works on a stable version of the program, the other test - on which the new version of the program is tested and if everything works here, then the operating bot is updated to a newer version.

23. Which ports are used by Corrade?

Corrade uses the same ports as Second Life

TCP - 53, 80, 443, 12043, 12046, 21002

UDP - 53, 3478, 3479, 5060, 5062, 12000 - 29999

24. Do I have to restart Corrade after configuration changes?

No, you don't have to, Corrade should detect these changes itself and apply them if they were made by Nucleus.

25. Are Wizardry and Steamworks or libraries getting any private data?

No, they don't download any private data.

26. Why shouldn't you use a proxy?

Corrade and Second Life hardly like having your computer access the Internet through a proxy server. This causes a problem on the side of Corrade with obtaining or sending results, downloading data from Second Life, especially regarding the avatar's wardrobe, handling notifications, information in which groups it is located. Therefore, it is recommended that the computer on which Corrade is running connects to the Internet directly without the use of a proxy server.

27. I have a problem with notifications or feedback.

Are you using llRequestURL()? If so, replace it with llRequestURL() which is not a secure URL, however this is because Second Life uses self - signed certificates for servers and Corrade may have a problem with them.

8. Dictionary

A

API (Application Programming Interface) – it is a set of rules that govern how a program or programs communicate with each other.

ATC (Air Traffic Control) – air traffic controller.

B

Bot - a program that performs activities faster in an automated manner that mimics a human being.

C

Co - ordinates - they define the position of an object, an avatar.

Compass rose - indicates the directions: north, south, east and west.

I

IM (Instant Message) – private message sent between two users.

L

LSL (Language Second Life) – pseudo language created by Liden Lab., which is a combination of C #, C ++, Java, which allows for basic animation of objects and interaction with the avatar.

M

Mesh - a type of graphic in Second Life that was created in an external graphics program, such as Blender, and imported into Second Life.

O

Object - primes connected with each other (or primes with meshes) into one whole.

P

PHP - language for creating dynamic websites.

Plot - a virtual place in Second Life, which is the result of dividing the region into smaller parts of land.

Prim - basic building block in Second Life.

Proxy - a server that is an intermediary between the user's computer and the target server from which the content is to be downloaded.

R

Region - virtual l and in Second Life.

Rezz - can mean, for example, placing an object from a closet, loading clothes, loading a texture on an object.

S

Second Life - a virtual world accessible via the Internet with its own community, montage system, etc.

Second Life Marketplace - Second Life store offering various user - created virtual goods.

U

URL - a web address that points to a specific resource on the Internet, such as a web page or file.

V

VPS - (Virtual Private Server) - virtual dedicated server, which is designated as a certain area on the server.

9. Images used

- **Cover** - image downloaded from mepixels.com[24] under the Public Domain Dedication (CC0) license,
- **Image of the assigned DOI** - downloaded from zenodo.org[25],
- **Picture 4.1 - Corrade logo** - image downloaded from grimore.org[26] under the 'free for compatibility' license,
- **Picture 4.2 - The way of information flow in Corrade** - obraz pobrany ze strony grimore.org[27],
- ℹ - image taken from wikimedia.org[28] under public domain license,
- 💿 - image taken from wikimedia.org[29]; author: derex99; under the CC BY - SA 3.0 license,
- 🔗 - image taken from wikimedia.org[30]; author: Fabrice Florin; under the CC BY - SA 3.0 license.

24 Website - https://www.mepixels.com/photo/bot - is - working - in - pc - 51
25 Zenodo website - https://zenodo.org/
26 Grimore.org website with logo - https://grimore.org/_detail/secondlife/scripted_agents/secondlife_scripted_agents_corrade_banner.png?id=secondlife%3Ascripted_agents%3Acorrade%3Alogos
27 The website grimore.org - Corrade - https://grimore.org/secondlife/scripted_agents/corrade
28 Wikimedia.org website - https://commons.wikimedia.org/wiki/File:Info_icon - 72a7cf.svg
29 Wikimedia.org website - https://commons.wikimedia.org/wiki/File:CD_icon_test.svg
30 Wikimedia.org website - https://commons.wikimedia.org/wiki/File:Media_Viewer_Icon_ - _Link_Hover.svg

10. Additional materials

Additional materials are available with this e-book or on the website https://ko-fi.com/nitropl/shop for the author's support.

10.1. VPS bot installer (Powershell scripts, Windows)

My proprietary Corrade installer, which will install Corrade with additional programs, e.g. on VPS. Script operation:

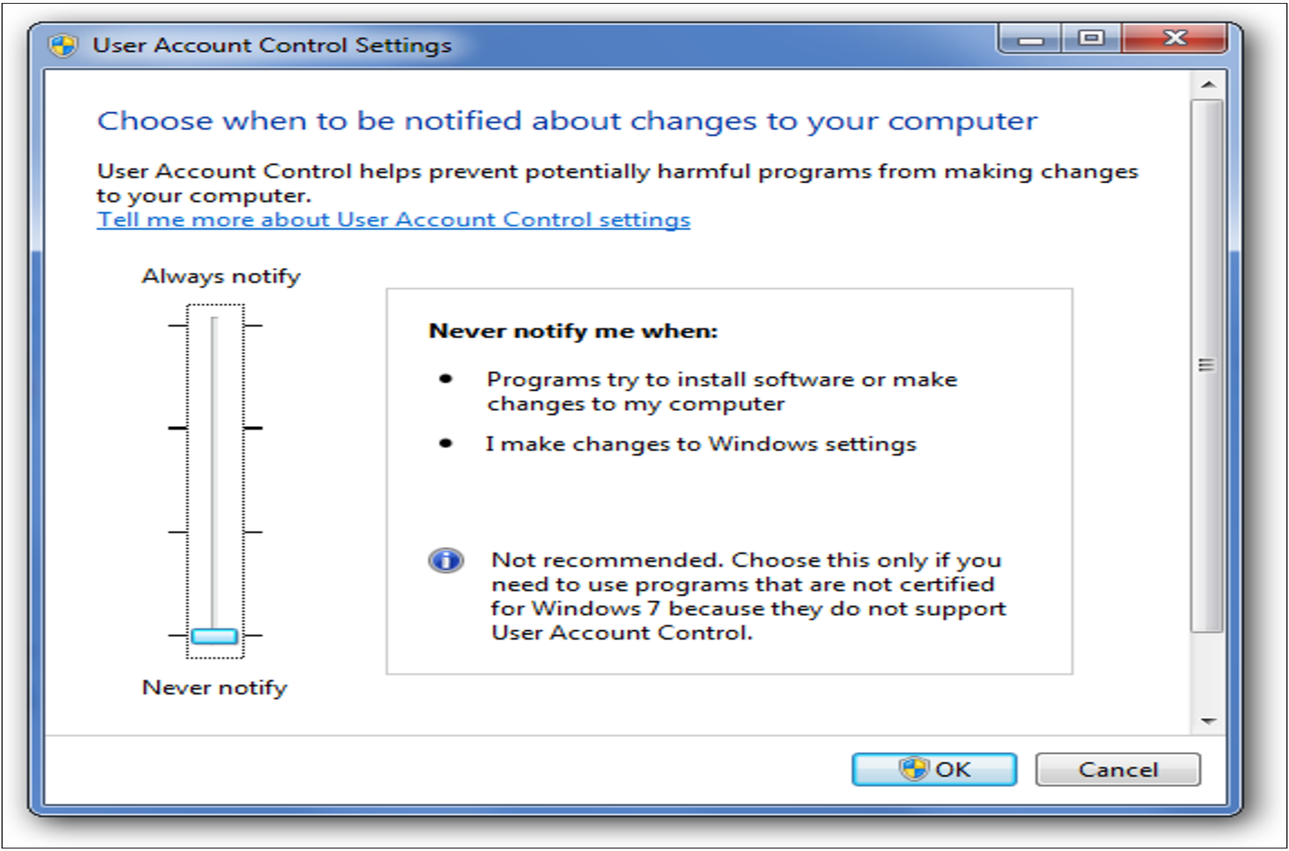
- adds appropriate ports to the network firewall,
- installs a set of libraries that allow many applications created with Microsoft's programming tools to run in Windows (Visual C++ Redistributable Package for Visual Studio 2019),
- installs the .NET Framework,
- installs a number of updates to upgrade Powershell to version 5.1,
- installs .NET Core,
- installs 7 - ZIP,
- installs XAMPP (optional)
  - adding a rule of ports used by XAMPP to the firewall,
  - extract XAMPP to drive C:\to path C:\xampp,
  - copies Apache configuration,
  - copies useful Apache modules,
  - deletes the htdocs folder and copies the previously prepared folder,
  - copies configuration files - my.ini and mysql.ini,
  - installs services for Apache and MySQL,
  - set the root password for MySQL,
  - loads SQL queries from a file,
  - adds PHP to the system path,
  - adds a 'corrade' user with a predefined password in the MySQL permissions table and grants permissions to the 'corrade' database which it also creates.
- If XAMPP is not installed:

  - unpacks PHP to C:\php,
  - adds PHP to the system path,
  - copies the php.ini file.
- installs Corrade
- copies bot AI to folder,
- copies bot and Nucleus configuration,
- creates shortcuts to the bot on the desktop, installs the service in the system, adds a task to the task scheduler that checks the availability of the bot (if there is a mcorrade.ps1 file in the bot folder).
- installs Firefox (optional),
- installs Notepad++ and copies default settings (optional),
- configures the computer (optional),
- disables unnecessary services (print spooler, disk defragmentation, fax, topics, program compatibility assistant, Windows Update),
- turn off the animation when starting the computer,
- deletes current volume backups,
- disables the system restore option,
- disables hibernation of the system,
- enable last access timestamp update on every directory that is listed on an NTFS volume,
- enables encryption of files and directories on an NTFS volume,
- disables memory paging file encryption in Windows,
- disable 8.3 - character file name creation on NTFS and FAT volumes,
- turns off DEP,
- ignores errors during system startup,
- allows remote connection to the computer,
- disables patches for Meltdown and Specter,
- installs Rainmeter (optional),
- installs HWINFO in the system and in the Task Scheduler,
- installs 'better' settings for the command line
- adds the Task Scheduler as a shortcut on the Desktop,
- adds and sets another power plan (optional).

The installer is located in the bot_installer folder. All script actions will be written to install.log.txt. To use the installer you need, among others 5 GB of free disk space (no, the installer will not use it, it is only as a reservation).

I present you step by step what you need to do:

1. Disable UAC (User Access Control) - lower the slider level, click OK, restart the target computer.



Choose when to be notified about changes to your computer

User Account Control helps prevent potentially harmful programs from making changes to your computer.
Tell me more about User Account Control settings

Never notify me when:

- Programs try to install software or make changes to my computer
- I make changes to Windows settings

Not recommended. Choose this only if you need to use programs that are not certified for Windows 7 because they do not support User Account Control.

Picture 10.1 - Disabling User Access Control in Windows

2. Now we are preparing to install Corrade on another computer.
3. Go to additional materials, to the bot_installer folder.
4. You have a lot of files there.
5. Open configs\installer.config.ps1.

```
#SET XAMPP INSTALL
$xampp_install=0
```

```
#SET MYSQL ROOT PASSWORD
$mysqlRootPassword="12345"
#SET MYSQL CORRADE PASSWORD
$mysqlCorradePassword="qazwsxedc"
#INSTALL FIREFOX
$firefox_install=0
#INSTALL NOTEPAD++
$notepadpp_install=0
#CONFIGURE COMPUTER
$configurePC=0
#SET RAINMETER INSTALL
$rainmeter_install=0
#ADD  and SET POWER PLAN
$powerplan=0
```

Here we decide what components we want to install:

1. `$xampp_install=0`

   1 - installs and configures XAMPP[31]

   0 - bypasses the XAMPP related process

   Below is a more detailed description for this option.

   - xampp_config\apache - in this folder you can define the initial Apache configuration ('conf' folder) and additional modules for it ('modules'). These folders mirror the tree of folders and files in XAMPP. Copy your configuration here, remembering that on the target computer the server is installed by default to the 'C:\xampp' folder. In the 'modules' folder, add the modules you want to be copied to the Apache modules folder on the target computer. Make sure these modules are according to the Apache version of the XAMPP package you want to use and that they are enabled in the target configuration.
   - Initially you'll find the modules that I use - transfer speed limitation (mod_bw module), DDoS attack protection - Slowloris (mod_antiloris) and too many connections from one IP address (mod_evasive, mod_limitipconn) and I hope they will be useful, otherwise you can use them delete from folder.
   - xampp_config\htdocs - contains a predefined public directory for XAMPP. Place your scripts, pages, etc. here. It will be copied after installing XAMPP. From me you have

31 Website project: https://www.apachefriends.org/

browsing the MySQL database (PHPMyAdmin[32]), displaying system information (phpsysinfo[33]) and a script about the current PHP settings (script: phpsysinfo.php).

- xampp_config\php.ini - previously prepared php.ini file that will be copied after XAMPP installation.
- xampp_config\my.ini - previously prepared my.ini file (for MYSQL), which will be copied after XAMPP installation.
- xampp_config\sql.sql - SQL file with SQL queries that will be executed after XAMPP installation - here I just added corrade database creation.

2. `$mysqlRootPassword="12345"`

Sets the initial password for MySQL root from the XAMPP package, rather use some simple password to start with (use letters, numbers, but no special characters - you can change it later), because the script has a problem with setting such complicated passwords and you will have to manually regain access to Database.

3. $mysqlCorradePassword="qazwsxedc"

   Sets the initial password for the user 'corrade' when creating a database from the XAMPP package. The method of setting the password is described as above, i.e. letters and numbers at the beginning without special characters.

4. $firefox_install=0

   Copies a previously defined folder with Firefox.

   In programs\FirefoxPortable you will find a portable version of the program that will be copied to the target computer. You can run it normally and set profile, settings, add bookmarks, etc. - everything will be saved in its folder.

   From me you have tabs for Apache modules, faster access to Nucleus, PHPMyAdmin, PHPSysInfo, Corrade documentation online.

5. $notepadpp_install=0

   Installs Notepad++, after installation it copies the profile from configs\Notepad++, which can be previously modified.

6. $configurePC=0

   Configures the computer (recommended for Windows VPS), e.g. disables Meltdown and Specter fixes, disables some features, disables themes, etc.

7. $rainmeter_install=0

32 Website project: https://www.phpmyadmin.net/
33 Website project: https://phpsysinfo.github.io/phpsysinfo/

Installs Rainmeter (desktop gadgets) by copying the programs\Rainmeter folder, which can be customized.
Along with Rainmeter, HWINFO (programs\hwinfo) is installed to collect temperature from the system and the task to Task Scheduler (tasks\hwinfo.xml).

8. $powerplan=0
   Installs and sets a new power plan for the system (others\server.pow file) that prevents the hard drives from turning off; turns off the screen after a minute, etc.

After setting up the script installer configuration file and possibly additional folders for each of the options described above, we move on.

- corrade\bot_ai - this folder contains files and folders that will be copied to the bot folder. You can put the bot's brain and heart here.
- In this folder you can also put files already configured: Configuration.xml and Nucleus.xml from Corrade.

In this folder there are files from me (which is basically nothing to configure):

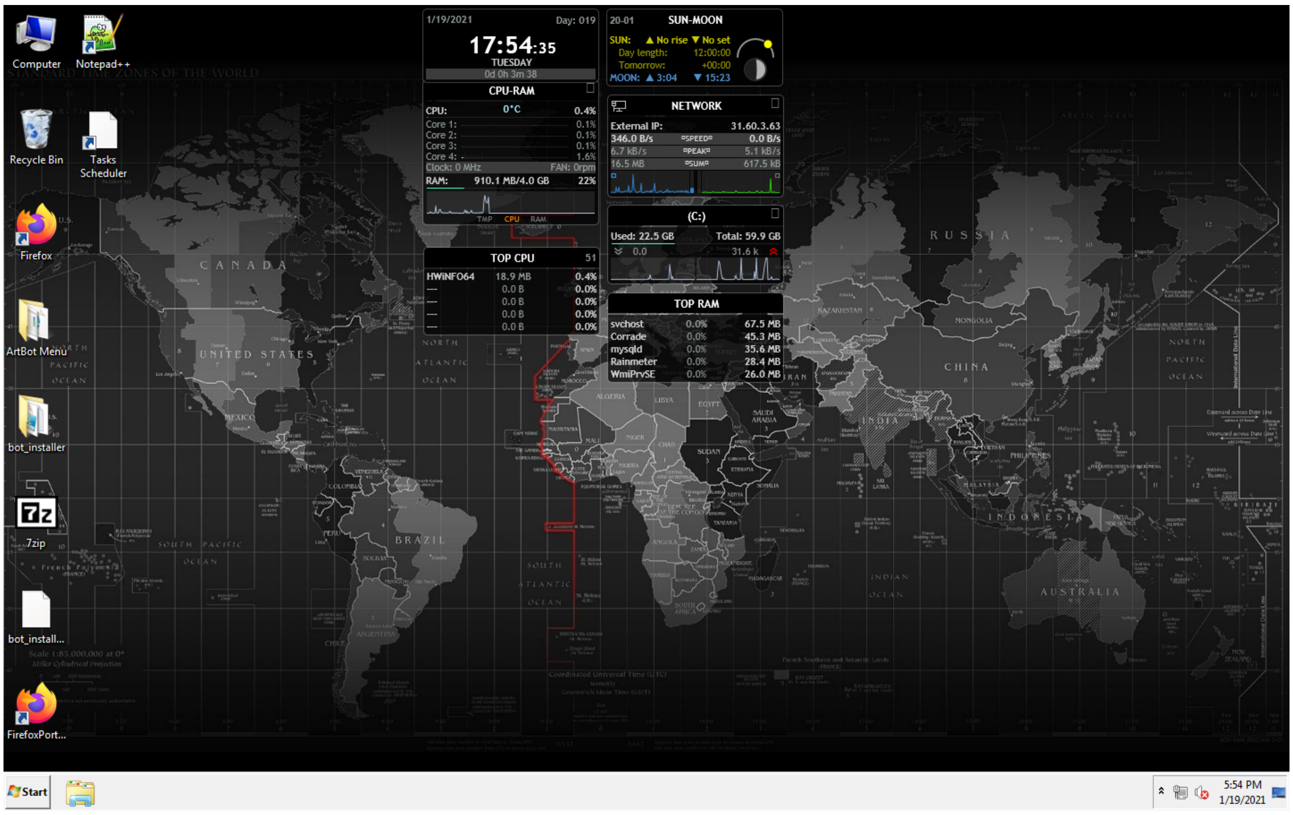
- corrade-kill.bat - kills the bot process,
- Create ShortCuts.bat - creates shortcuts on the desktop in the 'ArtBot Menu' folder,
- mcorrade.ps1 - Powershell file that allows you to manage the bot,
- nssm.exe - allows you to install the program as a service in the system,
- restart.bat - restarts the boot,
- start.bat - boot starts,
- stop.bat - stops the bot,
- update.bat - updates the boot.

Other files in the installer folder:

- installer.ps1 - the main part of the installer,
- installer.bat - a batch program that runs the installer.ps1 file with the appropriate parameters,
- functions.ps1 - functions created for the installer,
- configs\php.ini - php.ini file, which will be copied in case of not installing XAMPP,
- programs\7zip.exe - program 7 - ZIP,
- programs\php.zip - archiwum zip z PHP,

- programs\Root_Certificate_Updater - program for updating certificates in the system,
- programs\VC_redist.x64.exe - Visual C++ Redistributable Package for Visual Studio 2019 x64,
- programs\VC_redist.x86.exe - Visual C++ Redistributable Package for Visual Studio 2019 x86,
- registry\console - restore - settings.reg - settings for the command line
- System updates by updating Powershell to version 5.1:
  - patches\kb4457144 - x64.msu,
  - patches\kb4457144 - x86.msu,
  - patches\W2K12 - KB3191565 - x64.msu,
  - patches\Win7 - KB3191566 - x86.msu,
  - patches\Win7AndW2K8R2 - KB3191566 - x64.msu,
  - patches\Win8.1 - KB3191564 - x86.msu,
  - patches\Win8.1AndW2K12R2 - KB3191564 - x64.msu.

6. If you want, you can modify the installer.ps1 file, add your programs to the folder and the installation script.
7. If everything is ready, pack the bot_installer folder, preferably with 7 - ZIP.
8. Transfer the ready archive to the target computer.
9. Unpack the folder.
10. Open the bot_installer folder.
11. Click 2x on installer.bat.
12. Wait for the script to install everything and start the computer.
13. The installer logs the entire installation process to the install.log.txt file in the bot_installer directory, so you can then check this file for errors.
14. Once launched, you will see the following image:



Picture 10.2 - Desktop after installation via script

15. This means everything is ready to go.

FAQ

1. Corrade does not work after installation

It does not work because the Corrade service is disabled by default.

To run Corrade, you have an 'ArtBot Menu' folder on your desktop and there is a shortcut called 'Artbot Start', double - click on it and Corrade should run in the background.

2. XAMPP server failed to start (no access to address http://127.0.0.1)

- Run XAMPP panel C:\xampp\xampp - control,
- View on program events,
- Perhaps the services were installed incorrectly
  - Then stop Apache and/or MySQL by clicking on the STOP button,

- Next, next to the program, click on ✔ to uninstall the service,
- Then click again in the same place on ✕ to install the program as a service,
- Give to START next to each program.

- If Apache did not start, there are many reasons for this
  - To see what happened, in the C:\xampp directory, run apache_start.bat,
  - You will see a message from Apache what happened,
  - There may be something wrong with the configuration files.
- I encourage you to use Google here.

3. **I cannot log in to the server as 'root' on the MySQL server (password reset)**
   - Launch the XAMPP panel,
   - stop the MySQL server,
   - Open the file in notepad: C:\xampp\mysql\bin\my.ini,
     - At the beginning of the [mysqld] section, add:

```
skip-grant-tables
```

- Save file.

- In the XAMPP panel with MySQL, select START,
- In the folder C:\xampp\mysql\bin run the command prompt,
- Enter and confirm: C:\xampp\mysql\bin\mysql.exe - - user = root,
- After logging in as root without a password, we execute further SQL commands:

```
FLUSH PRIVILEGES;
ALTER USER 'root'@'localhost' IDENTIFIED BY '12345';
ALTER USER 'root'@'127.0.0.1' IDENTIFIED BY '12345';
exit
```

- Now we delete the previously added skip - grant - tables in the my.ini file and save the file.
- In the XAMPP panel, we stop and start MySQL.
- Now we can login to MYSQL as root with the password 12345.

10.2. Updating programs in bot_installer

Before packing the Corrade installer along with additional programs, think about updating programs, as those available in additional materials may be outdated a long time ago.
For a list of programs that require attention, see the bot_installer\README.txt file.

It is enough to download the program from the given URL to the indicated location given by the PATH path, where we agree to replace the files. Please note that if the program was a ZIP archive then download ZIP, if EXE is EXE, MSI is MSI, etc.

10.3. mcorrade.ps1 - bot management (Powershell, Windows)

This is my Powershell script that allows me to manage Corrade.

You will find it in additional materials: bot_installer\corrade\bot_ai.

To use it, copy all files and folders from this folder to the bot folder.

Run the command line (not from PowerShell) to call the program with parameters type:

```
powershell.exe -ExecutionPolicy Bypass -File .\mcorrade.ps1 -PARAMETR
```

where PARAMETER can be:

- -Latest - downloads and updates Corrade to the latest version,
- -Status - shows the current status of the Corrade service,
- -Start - launches Corrade,
- -Stop - stops Corrade
- -Restart - reboots Corrade,
- -InstallService - installs Corrade as a service,
- -DeleteService - removes Corrade as a service,
- -EnableService - enables the Corrade service,
- -DisableService - turns off Corrade as a service,
- -GetVersion - downloads the current version of Corrade,
- -CreateShortCuts - creates bot shortcuts.

10.4. Description of the files in bot_installer\corrade\bot_ai

- **bot_ai\php\botconfig.ini**

```
[general]
MonitorServices = 1
MonitorServicesAttempts = 3
MonitorServicesCheckEvery = 1
MonitorServiceCreatePrim = 1
MonitorServiceCreatePrimAxisZ = 10
botTitle = "Bot Manager"
botTitleTarget = "Bot Group"
cacheTypeWriteDatabase = 0
[homeback]
enabled = 0
slurl = "http://maps.secondlife.com/secondlife/XXXXXX/112/112/23"
polygon = "<127,97,21>|<97,97,21>|<97,127,21>|<127,127,28>"
```

This file is a corrupted version of the configuration file for my bot.

general.MonitorServices	enables Corrade monitoring
general.MonitorServicesAttempts	defines the maximum number of attempts after which the bot will be restarted
general.MonitorServicesCheckEvery	how much does it mean trying to check if the bot works
general.MonitorServiceCreatePrim	or when checking the bot, try to create a temporary prime in Second Life
general.MonitorServiceCreatePrimAxisZ	at what distance to place the provisional prime
general.botTitle	what bot title to set (from the group)
general.botTitleTarget	target group for setting the title
general.cacheTypeWriteDatabase	cache write type: 0 - file, 1 - database
homeback.enabled	whether the procedure for checking whether the bot is on the right plot is on
homeback.slurl	SLURL region and plot where the bot

	should be
homeback.polygon	plot boundaries in which the bot should move; the limit is set by the Nitro Position Recorder tool[34].

- **composer.phar (with files: composer.json, composer.lock, folder: vendor)**

Dependency management for a PHP project (I gave PHPLint here for syntax checking) using Composer[35].

- **class.point.php**

My point definition script

$point = new classPoint();	defines a new point
$point - >setPoint(1,3,4);	sets the coordinates of the point: X - 1, Y - 3, Z - 4
$point - >setPointX(1);	sets the X coordinate
$point - >setPointY(3);	sets the Y coordinate
$point - >setPointZ(4);	sets the Z coordinate
$point - >getPointX();	gets the value of the X coordinate
$point - >getPointY();	gets the value of the Y coordinate
$point - >getPointZ();	gets the value of the Z coordinate
$point - >getPoint();	gets a point to an array
$point - >setPointFromArray(array(1,3,4));	sets the coordinates of a point from an array

- **corrademonitorservice.php**

PHP script that checks the availability of the bot.

The corresponding task must be installed in the Windows task scheduler.

- **cron.bat/cron.php**

A cron written in PHP that can periodically call bot functions (e.g. to do something in Second Life).

The corresponding task must be installed in the Windows task scheduler.

34 Second Life Marketplace - https://marketplace.secondlife.com/p/Nitro - Position - Recorder/20479529 or the sources folder in the additional materials

35 Website - https://getcomposer.org/

Here he can check if the bot is at home on the plot.

- **inPolygon.class.php**

Script that checks if a point is in a polygon.

- **logs**

Folder that contains text files with events logged by the writeDebugFile() function.

- **phplint.bat**

A batch program that checks the syntax of PHP files in the current directory along with subdirectories.

- **config.php**

PHP configuration file used by other PHP files that are part of my bot's brain.

```php
<?php
error_reporting(0);
//Bot Connection
define('BOT_GROUP', 'Bot Group');
define('BOT_PASSWORD', 'PASSWORD');
define('BOT_URL', 'http://127.0.0.1:9199');

//Database Connection
define('DB_CONNECTION', 'mysql:host=localhost;dbname=corrade');
define('DB_LOGIN', 'corrade');
define('DB_PASSWORD', '');

//constants Second Life
define('TEXTURE_DEFAULT', '89556747-24cb-43ed-920b-47caed15465f');
define('TEXTURE_BLANK', '5748decc-f629-461c-9a36-a35a221fe21f');
define('TEXTURE_TRANSPARENT', '8dcd4a48-2d37-4909-9f78-f7a9eb4ef903');
```

BOT_GROUP	name of the group to which the bot belongs
BOT_PASSWORD	password to authenticate the command in the group
BOT_URL	URL to the bot (same as in the HTTP server settings in the bot configuration file)
DB_CONNECTION	database connection (if you use)
DB_LOGIN	database login
DB_PASSWORD	database password

TEXTURE_DEFAULT	UUID of the default texture (leave Second Life as is)
TEXTURE_BLANK	Empty texture UUID (leave Second Life as is)
TEXTURE_TRANSPARENT	UUID of transparent texture (leave Second Life as is)

- **functions.php**

Functions that I use with my bot.

writeDebugFile($msg, $type = 'warn')	logs $msg events to file; default type - 'warn'
write_php_ini($array, $file)	writes data to an INI file
safefilerewrite($fileName, $dataToSave, $mode = 'w+')	saves data to a file in a safe manner - arguments: file name, data to write, write mode (overwrites by default)
safefilerewriteCache($name, $value)	saves data to the cache file in a safe manner
safefilerewriteDeleteCache($name)	delete everything in the cache file safely
getConfig($nameval)	gets a value from an INI configuration file named $nameval
clearConfig($section, $name)	deletes the $name value from $section in the INI configuration file
setConfig($section, $name, $value)	writes $value for the $name argument in $section
setCacheValue($name, $value)	writes the value of $value for the $name argument to the cache file
getCacheValue($name)	gets the value for the $name argument from the cache file
deleteCacheName($name)	clears the value for the $name argument from the cache file
deleteAllCache()	delete the cache file
wasURLEscape($param_tab)	a function from WAS that encodes the URL
SendToBot($param_tab, $ACK = 1, $connectTimeout = 10, $Timeout = 30)	sends the $param_tab parameter array to the bot; $ACK - request for a reply (value: 1) or send and forget (0); $connectTimeout - time to connect to the bot in seconds; $Timeout -

	waiting time for a bot response in seconds
SendMessageToAvatar($agent, $message)	sends $message to avatar with given UUID $agent in Second Life
wasKeyValueGet($key, $data, $base64decode = false)	WAS function - gets the value from the $data request from the $key key; if the data is base64 encoded, it can decode $base64decode immediately
CSV2Array($csv)	converts CSV to an array
wasCSVToArray($csv)	WAS function - converts CSV to an array
wasArrayToCSV($a)	WAS function - converts an array to CSV
vectorStringToArray($vector)	converts a vector as a character, e.g. <1,5,4> to array (1,5,4)
vectorArrayToString($vector)	converts a vector array to character form
vectorDistance($v1, $v2)	computes the distance between vectors that are arrays
AvatarNameToUUID($name)	tries to get the UUID for the avatar with the specific name $name
randomPassword($length, $keyspace = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')	randomizes a password of the given length $length from the given $keyspace character set
checkPerm($avatar, $func)	checks if a given avatar with UUID $avatar is assigned to a specific $func role
AvatarUUIDtoName($uuid)	tries to get the avatar name based on its UUID $uuid
getBOTUUID()	gets bot UUID
BlackListIMAdd($uuid)	adds $uuid's avatar to the blacklist; the avatar will not be able to talk to the bot privately (as long as the notification is installed)
BlackListIMExist($uuid)	checks if a given $uuid avatar has been added to the blacklist
BlackListIMDel($uuid)	removes an avatar from the blacklist
isValidUuid($uuid)	checks if the UUID is valid

getRegionANDPos($slurl)	gets a region and position from a given SLURL and gives them as an array
getCurrentRegion()	gets the current region that the avatar is in
RandomNumber($min, $max, $step = 1, $float = false, $places = 5)	randomizes a number from minimum $min to maximum $max; with the option of specifying the $step; whether to draw as an integer or floating point $float; with the given number of decimal places $places
calcObjectBorderPoints($point, $size, $ratio)	gives the boundary points for a given object in the table; $point - the center of the object; $size - the size of the object; $ratio - object rotation (see PosGrid topic)
BotIsActive()	checks if the bot is active
getBotPosition()	gives the current position of the bot
setTitle()	sets the title of the bot
posGrid($beginPoint, $size, $ratio, $lines, $endPoint)	see topic: PosGrid
isInParcel($point = '', $polygon = '')	checks if a given point $point is in a polygon

- **corradeMonitorService.bat**

A batch file that runs the bot availability monitor.

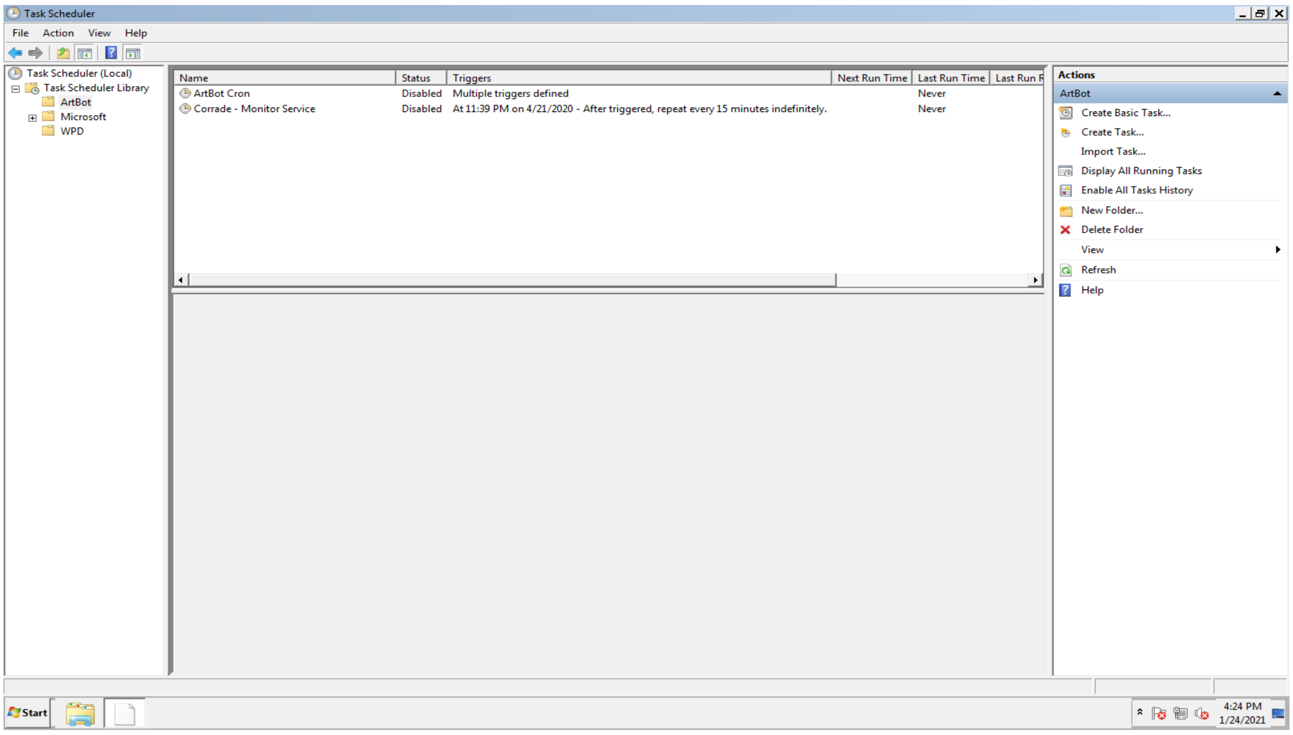
10.5. Task Scheduler

If you used, for example, my original script to install the bot on your computer, 2 tasks will be added to the system task schedule:

- ArtBot Cron - tasks that run for the bot every, e.g. 5 - 10 minutes,
- Corrade - Monitor Service - a job that monitors the Corrade process.

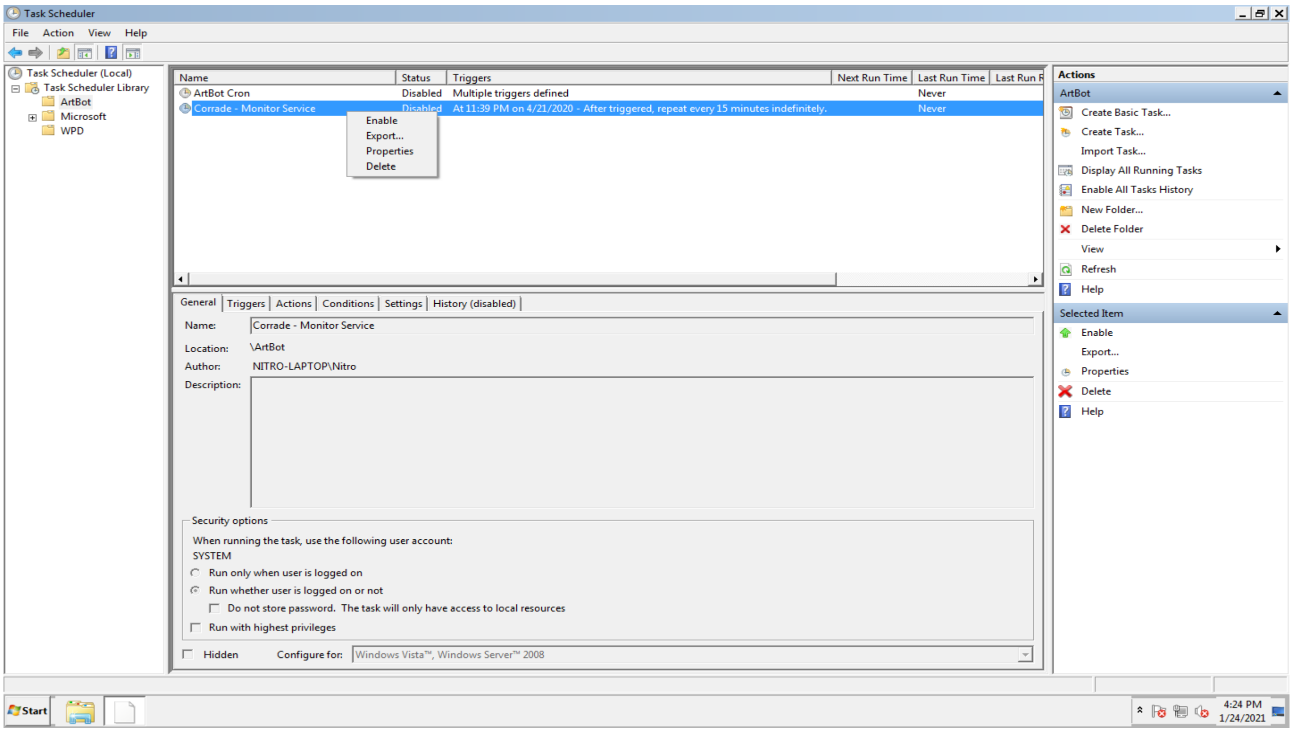
If you want, you can modify them for yourself.

Make sure that the task is assigned to the SYSTEM user permissions.



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run R
ArtBot Cron	Disabled	Multiple triggers defined		Never	
Corrade - Monitor Service	Disabled	At 11:39 PM on 4/21/2020 - After triggered, repeat every 15 minutes indefinitely.		Never	

Picture 10.3 - Bot tasks in the Windows Task Scheduler



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run R
ArtBot Cron	Disabled	Multiple triggers defined		Never	
Corrade - Monitor Service	Disabled	At 11:39 PM on 4/21/2020 - After triggered, repeat every 15 minutes indefinitely.		Never	

Name: Corrade - Monitor Service
Location: \ArtBot
Author: NITRO-LAPTOP\Nitro
Description:

Security options
When running the task, use the following user account:
SYSTEM
Run only when user is logged on
[✓ Run whether user is logged on or not]
Do not store password. The task will only have access to local resources
Run with highest privileges
Hidden Configure for: Windows Vista™, Windows Server™ 2008

Picture 10.4 - Options for the task