
Re-mars-tered Script Loader

Nov 25, 2020

Contents

1	Installation	3
1.1	Requirements	3
1.2	Steps	3
1.3	Known issues	4
1.4	General Troubleshooting:	4
2	Overlay Guide	5
2.1	Keyboard Shortcuts	5
2.2	Menus	5
3	Changelog	9
3.1	v0.5.0	9
4	Contributing	11
4.1	Basics	11
4.2	Contributing to the docs	12
4.3	Contributing to the Lua Core library	13
4.4	Contributing the to C++ Core	13
5	API	15
5.1	Events	15
5.2	rfg	17
5.3	rsl	99
6	Guides	103
6.1	Autorun scripts	103
6.2	Script events	104
6.3	Introduction	105
6.4	Lua overrides	107
7	Examples	109
7.1	Explosions	109
7.2	Logging	111

Important: Development for RSL1 has ended. A rewrite has started development: [RSL2](#).

The Re-mars-tered Script Loader (RSL) adds a lua scripting API to Red Faction Guerrilla Re-mars-tered Edition (RFGR). The goal of this is to allow for more creative and dynamic mods through scripting, but also to bypass many of the current limitations of modding the game. You should start with the [introduction](#) page.

Special thanks is given to [MWSE v2](#) and it's developers. It's been used as a design template/example and source of ideas for RSL. You may also notice many structural similarities with these docs and theirs.

Important: The latest version of RSL can be found on the [github releases page](#). For discussion about scripting, troubleshooting, or modding try the mod_makers_talk channel in the [Official Red Faction discord server](#).

1.1 Requirements

- Windows 7, 8, or 10. Windows XP support is unconfirmed.
- The [Visual Studio 2019 x86 Redistributable](#) is required for RSL. You might need to fully restart your PC for the install to properly complete.
- A copy of Red Faction Guerrilla Re-Mars-tered from steam. Support for the GOG version of re-mars-tered is planned for the future. The RSL **does not** work for any version of RFG other than re-mars-tered.

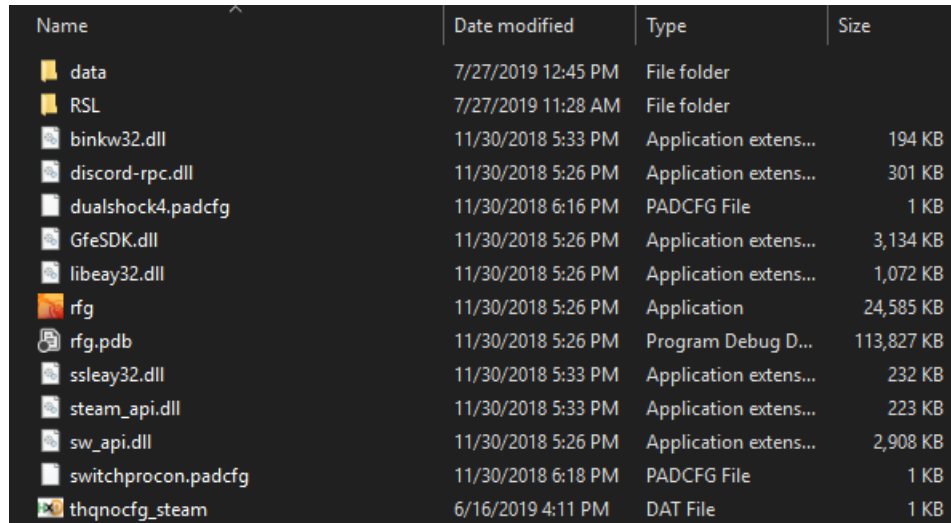
1.2 Steps

Note: If you have a version of RSL prior to 0.5.0 installed, you should delete it before installing this version to minimize the chance of issues.

- Download the latest release from the [releases page](#). Just select the latest release, click the artifacts tab, and download the zip file listed there.
- Unzip the release and copy the folder it contains into your Red Faction Guerrilla Re-mars-tered folder. In steam you can find that by right clicking the game in your library and then clicking `Properties > Local files > Browse local files`. Don't worry about overwriting files, the RSL **does not** modify any files included with the game. After copying the files your RFG folder should look something like this:

Important: Note that RSL.dll and the Scripts and Fonts folders are not in the same folder as rfg.exe. If they are, you've made a mistake.

- Run re-mars-tered through steam as normal, you should see the RSL launcher pop up, here you can pick whether or not to play with the RSL. Choose to play with the RSL enabled.



Name	Date modified	Type	Size
data	7/27/2019 12:45 PM	File folder	
RSL	7/27/2019 11:28 AM	File folder	
binkw32.dll	11/30/2018 5:33 PM	Application extens...	194 KB
discord-rpc.dll	11/30/2018 5:26 PM	Application extens...	301 KB
dualshock4.padcfg	11/30/2018 6:16 PM	PADCFG File	1 KB
GfeSDK.dll	11/30/2018 5:26 PM	Application extens...	3,134 KB
libeay32.dll	11/30/2018 5:26 PM	Application extens...	1,072 KB
rfg	11/30/2018 5:26 PM	Application	24,585 KB
rfg.pdb	11/30/2018 5:26 PM	Program Debug D...	113,827 KB
ssleay32.dll	11/30/2018 5:33 PM	Application extens...	232 KB
steam_api.dll	11/30/2018 5:33 PM	Application extens...	223 KB
sw_api.dll	11/30/2018 5:26 PM	Application extens...	2,908 KB
switchprocon.padcfg	11/30/2018 6:18 PM	PADCFG File	1 KB
thqnocfg_steam	6/16/2019 4:11 PM	DAT File	1 KB

Fig. 1: How your RFGR folder should look after a proper install. You might see a few other folders, which is fine.

- Wait until you hear 3 beeps. This means that the RSL has successfully activated. It might take 5-10 seconds before occurring, and you'll need to load a save first before it's fully active.
- The RSL is now ready for use. You can use F1 to toggle the overlay where you'll see a welcome menu with more shortcuts. You should read the usage guide and scripting guides for more info.

1.3 Known issues

1.3.1 Problem: “The game freezes at startup, or doesn’t show the launcher”

1.3.2 Solution:

The launcher sometimes causes the game to lockup when it starts. This isn't easily reproducible and has no fix yet. Instead there are ways to bypass the launcher. Find your RFGR game folder, it contains `rfg.exe`. To bypass the launcher and play the game with the RSL loaded, create an empty text file in that folder called `RSL_No_Launcher.txt`. To bypass the launcher and play the game with the RSL disabled (lets you use MP features), create a text file called `Vanilla_No_Launcher.txt`.

1.4 General Troubleshooting:

- Make sure to restart your computer if you haven't already after installing the Visual Studio 2019 x86 Redistributable. There have been a few instances where the installation wasn't properly completed until the user fully restarted. For best results you should fully power down and start your PC again after several seconds.
- If that doesn't solve your issue, please [create a new github issue](#) or contact us on the [Official Red Faction Discord Server](#) in the `mod_makers_talk` channel. When you report a bug, please include a zipped copy of your RSL logs folder `/RSL/Logs`, the version you are using (available in the RSL about menu, or just as the name of the zip file you downloaded), and if possible, steps for reproducing your problem and a description of what is happening.

This guide will provide you with the basic info you'll need to use the RSL and it's different GUIs and controls. This guide assumes that you've already followed the installation guide and have it working.

2.1 Keyboard Shortcuts

- F1 - Toggle RSL overlay
- F2 - Toggle script editor
- Hold F3 - Deactivate RSL
- F4 - Toggle lua console
- F5 - Run current script editor script
- Numpad 1 - Toggle game hud
- Numpad 2 - Toggle fog
- Numpad 3 - Toggle free cam
- Numpad 4 - Lock object in crosshair to introspection menu
- Numpad 5 - Ragdoll player

2.2 Menus

When opening the overlay for the first time you'll see the Welcome menu which includes buttons for several often used menus, and a list of keyboard shortcuts. These menus and others not listed there can be opened through the menu bar at the top of the screen. Below is a short list of each menu and what they do, sorted by their menu bar category.

2.2.1 System

Deactivate script loader

Deactivates the script loader and any running scripts. Can also be accomplished by holding F3. Main use is to re-inject the script loader to attempt to fix a bug.

Reset core lua state

Disables all scripts and fully resets the script loader lua state. This means that any scripts, event, variables, etc, will be completely reset as if the script loader was just started. Also reloads the core library in the Core folder.

2.2.2 Tweaks

General Tweaks

Contains a large amount of useful tweaks for the player such as invincibility, infinite jetpack, ignored by ai, no ragdoll, custom move speed, custom jump height, custom explosion spawning, and more.

Teleport

Has a preset list of teleport locations and lets you create your own to teleport the player anywhere on the map. Preset locations include places like the tutorial area, each safehouse, and mount vogel.

Logger

Lists logging system output. Has options to filter by message type. This is the same output that can be found in the script loader Logs folder.

Theme editor

Lets you customize and save the script loader overlay color scheme and style. Saved to Settings/GUI Config.json. You can share this with anyone and have them put it in their settings folder so they can use you custom theme.

Camera settings

Includes settings for the free cam, experimental first person cam, and other camera settings like the far clip distance which is the distance at which nothing is rendered past.

Physics settings

Has general settings like gravity, frametime multiplier (pseudo slo-mo at values less than 1.0), and settings for the physics solver.

2.2.3 Scripting

Scripts

A list of scripts detected in the script loader scripts folder. Allows you to individually run or edit each of the scripts listed.

Lua console

Directly executes an input as a lua script. You can call lua functions and set lua variables here. So, for example, you might set the far clip distance to 1200 with `rfg.SetFarClipDistance(1200)`. You can also add new functions to be called in the console by calling a script with a non local function declaration.

Script editor

Built in script editor. Has all the common text editor shortcuts like copy, paste, select all, etc. This is mainly intended for quick edits. I recommend that you use an external editor like VSCode for larger scripts to take advantage of better syntax highlighting and error checking. Eventually people could also write there own extensions for external editors that highlight RSL specific lua functions and types. See the starbound lua VSCode extension as an example of this.

2.2.4 Help

Welcome

The welcome menu that shows up each time the script loader overlay is opened. Has buttons for commonly used menus and a shortcuts list.

Metrics

Lists some debug metrics about the RSL overlay.

About

Lists info about the current RSL version and info about some of the libraries (such as Dear ImGUI) used by RSL.

CHAPTER 3

Changelog

This page contains a list of changes, fixes, and additions made from version 0.4.0 onwards. Changelogs for older version can be found on the [github releases page](#). More details about any changes to scripting or new functions and types can be found in the scripting section of the docs.

3.1 v0.5.0

3.1.1 Changes

- Several dozen new game structs and functions bound to lua. See the scripting section of the docs for more information and example of scripting. This was the primary feature and goal of this release. Writing these docs took up about half of the dev time. Even so, they're incomplete in many places. Contributions are welcome and appreciated. See the [contributing guide](#) for info on how to contribute to the docs.
- Changed the input detection code to something less hacky which should have less issues. The only thing still on the old input system is the script editor shortcuts.
- Added an experimental first person camera, available in the camera settings menu. This is mainly as a proof of concept and to figure out any issues that might come up when making it. The free cam will be bound to lua in a later release so that anybody can perfect it.
- Several additions to the general tweaks menu: AI ignore player, disable player ragdoll, salvage count, ore count, supply crate count.
- Added "High LOD far clip distance" setting to camera menu. For best results set both this and the normal far clip distance setting to higher values (anything above 1200 makes no difference). Increases distance you can see terrain shadows and makes distant terrain look slightly better.
- Added several more error checks when starting the script loader. It will detect some common installation mistakes, warn you about them and shutdown until you've fixed it.
- Inline error markers when you run scripts in the editor
- Syntax highlighting and type tooltip info support in the script editor. Loads type definitions from `/Settings/LanguageConfig.json`. It has some limitations but will still be useful. I've only added a few types to save

time. I'd greatly appreciate if anyone could go through the documentation and add entries for each type and function to this file which I'd include in the next release.

- Added information bar at the top of the script editor that shows info such as line and column number, and what typing mode you're in (insert, caps lock, etc).
- When running scripts through the editor it will now check for syntax errors before running your script, then run it if none are detected.
- The script editor is now more performant with larger scripts. The colorization code was made more performant.
- Added a basic file browser sidebar to the script editor.
- Moved explosion spawner to it's own menu. Now lets you copy values from the games pre-existing explosions (rpg, mine, tank shell, etc), and has a list of the games visual effects and their IDs so you no longer need to guess them.
- Changed the lua console shortcut to F4 since the old shortcut stopped the use of tilde in the script editor and is a pain to access on some keyboards. See the welcome window for an updated keybinds list.
- The overlay now supports tab and window docking
- Added experimental autoloader dll. Just place dinput8.dll in your rfgr folder (contains rfg.exe) and install the RSL normally. This will automatically load the RSL when starting the game, avoiding the need for the injector. The only change is that you'd need to move dinput8.dll into another folder before starting the game to play MP. The injector is still included until the autoloader is more stable.
- Added support for [autorun scripts](#).
- Added an object introspection menu. To use it, aim at an object and press numpad 4. This will lock that object. Then go to the menu in the overlay and you'll see some basic info about the object like it's type, it's object list index, and it's handle.
- Added [lua overrides](#) for internal rfg lua scripts.

3.1.2 Fixes

- Fixed weapon lockers and crates freezing player movement until reload.
- Most issues with explosion spawning should be fixed with the new explosions spawner menu and explosion presets.
- Fixed an issue with the script editor not properly adding the ".lua" extension when saving a file for the first time, and added a few checks to prevent accidental overwrites.
- Updated the script editor to fix a bug causing improper cursor placement.

While the core code of the RSL is written with c++, you'd don't need to be able to write c++ to contribute. There are areas of contribution for all skill levels, such as improving or expanding the documentation or the lua core library.

4.1 Basics

If you're familiar with git, and know how to fork a repo, push your changes, and make a pull request you can skip or skim over this first section. Otherwise, this section will explain those things and show you the basic process of contributing. The later sections give more details on editing specific parts of the RSL like the docs, the lua core lib, or the c++ core.

4.1.1 Requirements

- A [github](#) account (they're free!). You can also get free benefits if you're a student via the [github student pack](#).
- A git client. Any can be used, but [Gitkraken](#) will be used in the examples shown in this guide.

4.1.2 Forking

Once you've fulfilled the requirements, you can head over to the [RSL github repo](#), where we'll start. The first thing you'll want to do is click the `Fork` button in the top right corner of the repo. After pressing the button you'll be brought to your fork of the RSL repo. A fork is a copy created of the original repo at the time you forked it. This is where you'll apply your changes to the code before getting them added to the main codebase.

4.1.3 Cloning

The next thing you'll want to do is clone your fork to your desktop. If you log into gitkraken with your github account, this is quite easy to do. You go to `File > Clone Repo > GitHub.com` and then select your fork of RSL and choose the local folder is should be stored in. Gitkraken will clone the repo to your PC for you. This is also explained [here](#) in the GitKraken docs.

4.1.4 Pushing changes

It's at this point that you would make your changes to the codebase. More details on how to make changes and other tools required for that are explained in the later sections of this page, but for now we'll explain how to push your changes back to github. First thing you should do is stage and commit any changes you've made through Gitkraken, and write a brief description of what the commit does. After clicking `Stage files/changes` to commit the changes are still local to your PC. To send your changes to GitHub, click the `Push` button at the top of the GitKraken window.

4.1.5 Making a pull request

Finally, you'll want to create a pull request to request that your changes get applied to the main RSL repo. If you go to your fork after pushing your changes, Github should automatically show a button asking if you'd like to make a new pull request. Click that button, select your changes, and type a description and name for your pull request explaining what it does and any issues it might resolve. If you don't see the button, you can also go to the [RSL PR page](#), and select `New pull request`. From here, the RSL code reviewer(s) might request edits to your changes, or if they have no change requests, merge your PR into the main repo. Once a PR is merged, it will automatically be built by appveyor with 30min, [here](#).

4.2 Contributing to the docs

This section explains how to edit the docs and contribute changes to them. The docs are on a separate repo than the rest of the code, located [here](#).

4.2.1 Docs requirements

- [Python](#) for building the docs locally. The latest version should be fine.
- [Sphinx](#) for generating the docs. If you have python and pip installed you can just run `pip install -U Sphinx` on the command line.
- (Optional) [Visual Studio Code](#) with the [reStructuredText extension](#) which will allow you to see a live preview of the page you are editing. This makes editing much smoother instead of needing to regenerate the docs on each edit and open them in your browser.

4.2.2 Editing the docs

Once you've installed Python and Sphinx you can start making edits to the docs. You'll be editing the rst files in the Docs folder. These files are used by sphinx to generate the docs that you are reading right now. This guide will not go over the syntax of reStructuredText, as there are many existing guides on that. You should check out the [reStructuredText Primer](#) and also look at the existing rst files for examples of how it works. This is where the VSCode extension mentioned earlier comes into play as it lets you see the effect of your edits live, which makes learning the syntax much easier.

Once you are done editing you should generate the docs locally and look for any errors before pushing your changes. This can be done by running `make.bat` on the command line. If you shift + right click the docs folder you should see "open a powershell window here". Once the window is open you can run `.\make.bat` to see a list of build options. Running `.\make.bat html` will build the docs as html files. After running that command you can find the built docs in `_build/html/index.html`. When you've fixed any mistakes, and you're ready to move your changes to the main repo, continue to the next step.

4.3 Contributing to the Lua Core library

The lua core library is a set of lua scripts that are run during RSL startup. Any functions or values provided by the lua core lib are available to all other lua scripts run by the RSL. Currently the core lib is quite barebones, only containing some useful enums such as `rfg.ObjectTypes`, but there's much opportunity for expansion. Good candidates for addition to the lua core lib are helper functions which people are repeatedly writing themselves in scripts anyways, or wrappers around certain behavior, like for example, providing preset, named teleport locations. Once the overlay guis are bound to lua, some of them will likely be moved into the core lua lib where possible, as more people will be able to edit them in lua form than in c++ form.

Editing the Lua core lib does not require any special tools. Just a text editor. Visual studio code is an excellent free text editor that is good for this, and offers many expansions which can enhance your editing experience. You can test your edits while the game is running. Simply edit the core lib files in your installation of the RSL and then in the overlay menu (opened with F1), select `System > Reset core lua state`. This will stop all currently running lua scripts, reload the core lib, and reload autorun scripts, as if you were running the scripts for the first time in this game session.

4.4 Contributing the to C++ Core

The c++ portion of the RSL is what provides all the primary functionality of the RSL such as the overlay system and scripting system. It manipulates rfg by hooking it's game functions to change or track their inputs, and locating different data structures in the games memory to modify them or provide lua scripts access to them. This part of the codebase is by far the most complex, and will have to be explained more in other guides. This section will give a basic rundown of how to compile the code yourself and edit it.

Editing the c++ side of RSL requires a copy of [Visual Studio 2019](#) with the following features enabled:

- **Workloads:**
 - Desktop development with C++
- **Individual components:**
 - VC++ 2019 v142 toolset (x86, x64)
 - Windows 10 SDK for Desktop C++ (x86 and x64, any/latest version)

To compile the project, open `RSL.sln` with visual studio 2019, and once open select `Build > Build solution`. Depending on if you selected to build it as Debug or Release, it the resulting `rsl.dll`, `injector.exe`, and `dinput8.dll`, will be in either the `Releases` or `Debug` folder of the local repo. You can then copy these files over to your RSL install and run the game as normal to see your changes in action. Usually you'll only have to copy over `RSL.dll` as it's where most of the code is.

Note: Debug builds have a much larger final dll since they include debug information, but also build much more quickly as they have minimum amounts of optimisation. You should generally use debug builds for development since using the debugger will show you more information with them. Release builds are good for sharing with others to test since they are small enough to share on discord.

This is only a very basic guide to the c++ codebase of the RSL, and how it works. Please see the additional guides linked below for more information on important concepts such as function hooking, rfg function calls, and lua binding.

Note: These other c++ related guides have not yet been written, and should be up over the course of the next few days. The docs are very much still a work in progress.

These are the lua namespaces used by the scripting api. They're really just tables containing functions and types which are used for organization.

5.1 Events

This table lists all the lua event types available for use.

5.1.1 FrameUpdate

This event is triggered every single frame. This makes it the most performance heavy event. Prefer using other events when possible since they will likely be more selectively triggered instead of being every frame. It's `RegisterId` is "FrameUpdate".

EventData

Frametime (float) The time since the last frame.

5.1.2 Initialized

This event is triggered the first time that a save is loaded. It's only triggered once per session of the game. For it to trigger again you must restart the game. It's a useful way to activate your mods/scripts as once this event is triggered all game values and state should fully be initialized and ready for use. If you run your script immediately in your `main.lua` the values it relies on may not be initialized yet. It's `RegisterId` is "Initialized".

EventData

This event has no data.

Examples

Below is a simple example of simple mod that disables fog when the game runs and which utilizes this event. This script would be placed in a file named `main.lua` so its automatically loaded when the RSL starts.

```
--This function will be called once when the Initialized event is triggered
--This happens when the first save is loaded. By then, all lua values and
--functions are safe to use.
function StartScript(EventData)
    rfg.HideFog()
end

rfg.RegisterEvent("Initialized", StartScript, "Hide fog script initializer")
```

5.1.3 Keypress

This event is triggered whenever a key is pressed or released. It's `RegisterId` is "Keypress".

EventData

KeyDown (bool) True if a key is being held down.

KeyUp (bool) True if a key was just released.

KeyCode (KeyCodes) The code of the key being pressed. Used to check which specifically which key is being pressed. See `RSL/Core/rfg/KeyCodes.lua` for a full list of keycodes.

Control (bool) True if the ctrl button is being held down.

ShiftDown (bool) True if the shift button is being held down.

AltDown (bool) True if the alt button is being held down.

WindowsDown (bool) True if the windows button is being held down.

5.1.4 Load

This event is triggered every time a save is loaded. Note that every time the player dies a save is loaded, so this can also be used as a "AfterPlayerDies" style event. Eventually an explicit event for the player dying may be added too if this isn't specific enough. It's `RegisterId` is "Load".

Note the difference between this and `Initialized`. This event is triggered every time a save is loaded, while `Initialized` is triggered only when the first save is done loading. `Initialized` is useful for one time script/mod activation and other things that should only happen once while this is useful for something that should be updated each time a save is loaded.

If you're using user messages to display data you'll need to have one of these events to regenerate them, as the game removes them each time a save is loaded.

EventData

This event has no data.

5.1.5 Mouse

This event is triggered whenever the mouse is moved or the scroll wheel is scrolled. It's `RegisterId` is "Mouse".

EventData

Scrolled (bool) True if the mouse wheel has been scrolled.

ScrollDelta (int) If the mouse wheel has been scrolled this is how much it's been scrolled and in what direction. Otherwise this is 0.

MouseMove (bool) True if the mouse has been moved.

MouseX (int) The x position of the mouse in screen coordinates.

MouseY (int) The y position of the mouse in screen coordinates.

Control (bool) True if the ctrl button is being held down.

ShiftDown (bool) True if the shift button is being held down.

AltDown (bool) True if the alt button is being held down.

WindowsDown (bool) True if the windows button is being held down.

5.2 rfg

This table contains the functions and types used internally by rfg, and some helper functions which make scripting easier.

5.2.1 Types

Lists all types functions in the rfg namespace.

AlertLevels

Alert level.

Access Variable	Value
<code>rfg.AlertLevels.Green</code>	0
<code>rfg.AlertLevels.Yellow</code>	1
<code>rfg.AlertLevels.Orange</code>	2
<code>rfg.AlertLevels.Red</code>	3
<code>rfg.AlertLevels.NumAlertLevels</code>	4
<code>rfg.AlertLevels.Invalid</code>	4294967295

AmmoTypes

A weapons ammo type, used in `WeaponInfo` instances. These are defined in `./Core/rfg/AmmoTypes.lua`.

Access Variable	Value
rfg.AmmoTypes.Bullet	0
rfg.AmmoTypes.Projectile	1
rfg.AmmoTypes.Thrown	2
rfg.AmmoTypes.Melee	3
rfg.AmmoTypes.Electricity	4
rfg.AmmoTypes.Repair	5
rfg.AmmoTypes.Blanks	6
rfg.AmmoTypes.Flame	7

AnimationActions

The current animation action of a [Human](#). Defined in `./Core/rfg/AnimationActions.lua`. This enum is absolutely massive (hundreds of values) so for the sake of time it hasn't been redefined here yet. Check the lua file for the values it contains.

AnimationGroups

A weapon animation group, used in [WeaponInfo](#) instances. These are defined in `./Core/rfg/AnimationGroups.lua`.

Access Variable	Value
rfg.AnimationGroups.None	4294967295
rfg.AnimationGroups.Default	0
rfg.AnimationGroups.AK	1
rfg.AnimationGroups.Melee	2
rfg.AnimationGroups.Pistol	3
rfg.AnimationGroups.Revolver	4
rfg.AnimationGroups.Rifle	5
rfg.AnimationGroups.Rpg	6
rfg.AnimationGroups.Mortar	7
rfg.AnimationGroups.Mine	8
rfg.AnimationGroups.Hammer	9
rfg.AnimationGroups.RemoteCharge	10
rfg.AnimationGroups.TurretMachinegun	11
rfg.AnimationGroups.TurretRocket	12
rfg.AnimationGroups.TurretRailgun	13
rfg.AnimationGroups.TwoHandCarry	14
rfg.AnimationGroups.Gutter	15
rfg.AnimationGroups.Shotgun	16
rfg.AnimationGroups.Sniper	17
rfg.AnimationGroups.Arc	18
rfg.AnimationGroups.Railgun	19
rfg.AnimationGroups.Nano	20
rfg.AnimationGroups.Thermobaric	21
rfg.AnimationGroups.Pipe	22
rfg.AnimationGroups.Shield	23
rfg.AnimationGroups.Grinder	24
rfg.AnimationGroups.Enforcer	25

AttachInfoData

Contains attachment data for an object if it is attached to another object.

Variables

ParentHandle (int) The handle of [Object](#) that this object is attached to.

ParentPropPoint (int) The parents prop point that the child is attached to.

ChildPropPoint (int) The prop point on the child which the child is attached to the parent by.

RelativeTransform (Matrix43) Rotation and translation of the attached object (the child) relative to the parent.

UseRelativeTransform (Bitfield) Uses the relative transform if set to 1. Range: 0-1

UpdatePhysics (Bitfield) Set to 1 if the game needs to update it's physics. Range: 0 to 1

Updated (Bitfield) Set to 1 if the game has updated it's physics. Range: 0 to 1

AudiolibCuePriorities

Exact purpose unknown. Seems to be a priority system for different types of audio (music, dialogue, etc). Defined in `./Core/rfg/AudiolibCuePriorities.lua`.

Access Variable	Value
<code>rfg.AudiolibCuePriorities.Invalid</code>	4294967295
<code>rfg.AudiolibCuePriorities.Low</code>	0
<code>rfg.AudiolibCuePriorities.VoiceAnimation</code>	1
<code>rfg.AudiolibCuePriorities.DialogueAnimation</code>	2
<code>rfg.AudiolibCuePriorities.VoiceAmbient</code>	3
<code>rfg.AudiolibCuePriorities.Medium</code>	4
<code>rfg.AudiolibCuePriorities.DialogueRadio</code>	5
<code>rfg.AudiolibCuePriorities.DialogueNormal</code>	6
<code>rfg.AudiolibCuePriorities.High</code>	7
<code>rfg.AudiolibCuePriorities.DialogueHigh</code>	8
<code>rfg.AudiolibCuePriorities.DialogueBriefing</code>	9
<code>rfg.AudiolibCuePriorities.VeryHigh</code>	10
<code>rfg.AudiolibCuePriorities.Music</code>	11
<code>rfg.AudiolibCuePriorities.NumAudiolibCuePriorities</code>	12

BaseArray

A generic array used by the game to store different types. It can store any type, such as [Objects](#) or [Humans](#), or literally anything else. You can access it's elements just like any other array by using brackets and specifying the index of the element you wish to access, like so:

```
-- Set `Object` equal to the first element in world object array.
-- Note that this array, unlike lua tables, is 0 based. That is, the first element_
↪has an index of 0.
Object = rfg.ActiveWorld.AllObjects[0]
```

The most common usage of this type is to iterate through the world object list, which contains all of the games objects, and perform some action on them. In this example the a for loop runs through the world object and counts the number of human objects currently in the game.

```
HumanCount = 0
ObjectList = rfg.ActiveWorld.AllObjects -- Make an alias to the list for convenience
for i=0, rfg.ActiveWorld.AllObjects.Size(), 1 do -- Loop through all valid indices of
    the list
    Object = ObjectList[i] -- Get the object at this index for convenience.

    if Object.Type == rfg.ObjectTypes.Human then -- Check if the object is a human.
        HumanCount = HumanCount + 1
    end
end

rsl.Log("Human count: " .. tostring(HumanCount))
```

Functions

Size (BaseArray Self) Returns the number of elements in this array as an **int**.

Length (BaseArray Self) Alternative name for Size.

Capacity (BaseArray Self) Returns the maximum number of elements this array can contain as an **int**.

BaseVehicleTypes

Base vehicle types. Defined in Core/rfg/BaseVehicleTypes.lua. Access through rfg.BaseVehicleTypes.

Access Variable	Value
rfg.BaseVehicleTypes.Invalid	4294967295
rfg.BaseVehicleTypes.None	0
rfg.BaseVehicleTypes.Courier	1
rfg.BaseVehicleTypes.Ambulance	2
rfg.BaseVehicleTypes.Tank	3
rfg.BaseVehicleTypes.ArtTank	4
rfg.BaseVehicleTypes.Bomber	5
rfg.BaseVehicleTypes.Special;	6

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

Bbox

Bounding box. Data represents a cube in the game world.

Variables

Min (Vector) Minimum point of the bounding box.

Max (Vector) Maximum point of the bounding box.

BlockMovementTypes

Used in [Humans](#). Defined in `./Core/rfg/BlockMovementTypes.lua`.

Access Variable	Value
<code>rfg.BlockMovementTypes.Pos</code>	0
<code>rfg.BlockMovementTypes.Orient</code>	1
<code>rfg.BlockMovementTypes.PosAndOrient</code>	2

CharDefHead

Contains info about a characters head such as it's mesh and animation rig.

Variables

Name (char*) Character head name.

LodName (char*) Lod file name.

MorphMatSRID (int) Unique ID for the heads material.

HeadMorph (int) Unknown value. Likely related to animation.

BaseHeadMorph (int) Unknown value. Likely related to animation.

NumSkinShaderParams (int) Number of skin shader params. Actual skin shader params not bound to lua yet.

CharacterInstance

Contains info about a characters head such as it's mesh and animation rig.

Variables

AimHandle (int) Unknown value.

OrigCharacterScale (float) The character instances original scale.

CharacterScale (float) The current scale of the character instance.

CiFlags (unsigned int16) Unknown value.

RenderAlpha (char) The alpha value of the character instance.

CacheLod (char) Unknown value.

BufVertsShadowOffset (int) Unknown value.

BufFrameVerts (char*) Unknown value.

Next (CharacterInstance) The next character instance. May be `nil`

Prev (CharacterInstance) The previous character instance. May be `nil`

VariantIndex (unsigned int) Unknown value.

BboxMaxDimension (float) Unknown value.

ChecksumStri

Used to store checksum info for various strings the game uses. Currently this just wraps an unsigned int but later on some of the helper functions the game has for this type will be included. They've not been added yet due to time constraints.

Variables

Checksum (unsigned int) Checksum value

ContactNode

Node on a doubly linked list of contact nodes. Contains info about the contacted object for this node, and number of other contacts. The list can be traversed by looping through `Previous` and `Next` until a `nil` value is reached.

Variables

ContactObject (unsigned int) The handle of the contact `Object` for this node.

NumberOfContacts (int16) The number of contacted objects.

Previous (ContactNode) The previous contact node. May be `nil`.

Next (ContactNode) The next contact node. May be `nil`.

DamageScalingInfo

Damage scaling info for weapons. Used in `WeaponInfo`.

Variables

ScaleValue (float) Unknown value.

Damage (float[4]) Unknown values. Array of 4 floats, 0 indexed.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

DofStateBlock

Depth of field (DOF) state values. Access this through `rfg.Dof`. So, for example, if you wanted to set the value of `FocusStartA`, you'd do that like so: `rfg.Dof.FocusStartA = 2.3`.

Variables

FocusStartA (float) Needs description. Default value: 1.0

FocusStartB (float) Needs description. Default value: 3.0

FocusEndA (float) Needs description. Default value: 50.0

FocusEndB (float) Needs description. Default value: 100.0

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

ExplosionInfo

Information about an explosion. Used when spawning an explosion. You can use `rfg.GetExplosionInfo()` to get an explosion of the specified name, or you can iterate the games explosion preset list directly. If you'd like to make a custom explosion, using an existing one as a template you can make a copy of it and modify it from there. See the examples section for a more detailed explanation of this. Note that many of these variable descriptions were pulled from `explosions.tbl` and slightly modified.

Variables

Name (String) The explosions name. Probably shouldn't change this in case the game expects certain named explosions to exist. Note: is really a `char[32]`, behaves like a string.

UniqueId (int) A unique Id used to identify an explosion preset.

NameCrcStr (unsigned int) CRC hash on the explosions name string.

Flags (unsigned int) Flags representing behavior about the explosion. The purpose of each bit/value is currently unknown. Should be easy enough to determine these by looking at the flags var description in `explosions.tbl`.

Radius (float) The explosion radius.

SecondaryRadius (float) The secondary explosion radius. Equivalent to `<CrumbleRadius>` in `explosions.tbl`.

KnockdownRadius (float) The knockdown radius in meters for humans. Also the distance at which the damage to humans falls off to 0 (from `explosions.tbl`).

FlinchRadius (float) The flinch radius in meters for humans.

AI_SoundRadius (float) The maximum distance in meters, that humans/ai can hear the sound of this explosion.

HumanMinDamageHitpoints (unsigned int) Amount of hitpoints to do at the Blast Radius of the explosion to humans.

HumanMaxDamageHitpoints (unsigned int) Amount of hitpoints to do at the exact center of the explosion to humans.

VehicleMinDamageHitpoints (unsigned int) Minimum amount of hitpoints to do at the Secondary Radius to vehicles and ordinary objects (not human or destroyable). Often will be much less than Max Vehicle Damage.

VehicleMaxDamageHitpoints (unsigned int) Amount of hitpoints to do at the exact center of the explosion to vehicles and ordinary objects (not human or destroyable).

PlayerDamageMult (float) Multiplier applied to damage calculation for players.

PlayerVehDamageMult (float) This is a multiplier applied to the damage calculation to the player's vehicle. It is used instead of the Player multiplier - they are not multiplied together.

PlayerVehicleImpulseMult (float) Impulse multiplier for when the player's vehicle is hit with this explosion.

ImpulseMagnitude (float) Maximum impulse to apply at the center of the explosion in kg * m/s. This is interpolated to 0 at the explosion edge.

StructuralDamage (int) Damage amount to be applied to destroyable objects. Damage to destroyable objects is computed entirely different than normal objects, so this value has little relevance compared to human or vehicle damage.

ExpandingExplosionDuration (unsigned int) This is used to control how long (in milliseconds) it takes an expanding explosion to reach the full radius.

ExpandingExplosionDelay (unsigned int) This controls how long to delay (in milliseconds) before starting to delete pieces, the effect(s) start immediately

NumEffects (unsigned int) The number of effects the explosion plays.

Effects (unsigned int[4]) An array of effects that the explosion plays. Up to 4 effects can be used. Zero based array.

NumMaterialEffects (unsigned int) The number of material effects the explosion has.

MaterialEffects (unsigned int[8]) An array of material effects that the explosion plays. Up to 8 material effects can be used. Zero based array.

MaterialReference (char[8]) Unknown value. Zero based array with 8 elements.

SalvageMaterial (SalvageMaterialTypes) Salvage material used by the explosion.

SalvageMaxPieces (int) Max number of salvage pieces dropped by the explosion.

SalvageProbability (float) The probability that the explosion will drop salvage. 1.0 is a 100% probability and 0.0 is a 0% probability.

TimeBetweenBreaks (int) Time between each destruction frame of breaking links. This might only be valid if another variable is set. Explosions.xtbl defines a parent variable to this one called `Break_off_pieces` with the following description: *"Set this to make the explosion behave like a nano-rifle (breaking off pieces from objects)"*.

HalfAngleDot (float) When specified it defines half angle (in degrees) for a cone around the fvec of the explosion orient, things outside that cone are not damaged.

BlastDecalRadius (float) The radius of the sphere used to place blast decals.

CameraShakeType (String) The camera shake type. Can be one of several preset values: `explosion_small`, `explosion_large`, `explosion_quake`, `explosion_very_large`, `explosion_smash`.

CameraShakeMultiplier (float) Base multiplier of the camera shake. Values greater than 1.0 will make the shaking effect more intense than usual.

CameraShakeFalloff (float) The distance from the explosion at which the camera doesn't shake at all.

IgnoreOrientation (bool) Whether the orientation of the explosion should be ignored. Unknown what it does when this is true. Perhaps uses some default orientation, like directly upwards in the y direction.

AlwaysRagdoll (bool) Always ragdoll instead of playing knockdown animations.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

Flyer

Flyer object. Access these by iterating `rfg.World.Objects` and filtering by type. Flyer objects have an `ObjectType` equal to `rfg.ObjectTypes.Vehicle` and an `ObjectSubType` equal to `rfg.ObjectSubTypes.VehicleFlyer`.

Inherits `vehicle`

Variables

ReqUrgency (float) Unknown value.

ReqMaxVel (float) Unknown value.

ReqPoint (Vector) Unknown value.

ReqVel (Vector) Unknown value.

ReqLookAt (Vector) Unknown value.

ReqPointIsValid (bool) Unknown value.

ReqVelsIsValid (bool) Unknown value.

ReqLookatIsValid (bool) Unknown value.

ReqStopAtPoint (bool) Unknown value.

ReqHalt (bool) Unknown value.

WingIsDetached (bool) Unknown value.

VehicleMass (float) Unknown value.

ChassisComCs (Vector) Unknown value.

ChassisComWs (Vector) Unknown value.

ThrustMatCs (Matrix) Unknown value.

ThrustMatWs (Matrix) Unknown value.

ThrustUpTarget (Vector) Unknown value.

HoverNoiseVec (Vector) Unknown value.

HoverNoiseVecNew (Vector) Unknown value.

HoverNoiseVecOld (Vector) Unknown value.

HoverNoiseTimer (Timestamp) Unknown value.

HoverNoiseDur (float) Unknown value.

EngineOffSuspensionPeriod (Timestamp) Unknown value.

TurbineRpmTarget (float) Unknown value.

TurbineRpm (float) Unknown value.

TurbineLoad (float) Unknown value.

LWing (int) Unknown value.

RWing (int) Unknown value.

LCanard (int) Unknown value.

RCanard (int) Unknown value.

WingtipEffect (unsigned int[2]) Unknown value.

FrontThruster (unsigned int[2]) Unknown value.

CenterThruster (unsigned int[2]) Unknown value.

MainEngines (unsigned int[2]) Unknown value.

Jetwash (unsigned int) Unknown value.

EngineForce (float) Unknown value.

StartupTimer (Timestamp) Unknown value.

LastPlayerDamage (Timestamp) Unknown value.

CatchFireTimer (Timestamp) Unknown value.

FlyerFlags (FlyerFlags) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

FlyerFlags

Flags used by [Flyer](#) objects.

Variables

LandingGearUp (bool) Unknown value.

LandingGearCollisionOff (bool) Unknown value.

NeedsToTakeoff (bool) Unknown value.

HasLift (bool) Unknown value.

StayAtConstantHeight (bool) Unknown value.

OnGround (bool) Unknown value.

FootGroundEffects

Has variables for a humans footsteps and walking.

Variables

Name (char[20]) Name of the effect.

WalkFoley (int) Walk sound effect id.

RunFoley (int) Run sound effect id.

JumpFoley (int) Jump sound effect id.

LandFoley (int) Landing / ground impact sound effect id.

```
struct human_info_flags //19 {
```

```

__int8 female : 1; __int8 coin : 1; __int8 preload : 1; __int8 officer : 1; __int8 unarmed : 1; __int8 vip
: 1; __int8 use_tech_level : 1; __int8 sniper : 1; __int8 elite_dodge : 1; __int8 riot_shield : 1; __int8
body_armor : 1; __int8 nonflammable : 1; __int8 no_attaching_projectiles : 1; __int8 guerrilla_tech : 1;
__int8 guerrilla_rfc : 1; __int8 old_coot : 1; __int8 dan : 1; __int8 jenkins : 1; bool driverless_exit_only;
};

```

FootPlant

Describes which foot/feet are planted on the ground. Defined in `./Core/rfg/FootPlant.lua`.

Access Variable	Value
<code>rfg.FootPlant.LeftFoot</code>	0
<code>rfg.FootPlant.RightFoot</code>	1
<code>rfg.FootPlant.NeitherFoot</code>	2
<code>rfg.FootPlant.BothFeet</code>	3

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

GameSaveInfo

Additional info about the current save game.

Variables

NumMissionsCompleted (int) The number of missions completed in this save.

NumActivitiesCompleted (int) The number of activities completed in this save game.

DistrictsLiberated (int) The number of districts liberated in this save game.

TerritoryIndex (int) The current territory index of the player in this save game.

DistrictIndex (int) The current district index of the player in this save game.

Day (int) The real world day at the time this game was saved.

Month (int) The real world month at the time this game was saved.

Year (int) The real world year at the time this game was saved.

Hours (int) The number of hours played in this save.

Minutes (int) The number of minutes played in this save.

Seconds (int) The number of seconds played in this save.

AutoSave (bool) Whether or not autosave is enabled.

UsedSlot (bool) Unknown value.

DlcId (int) Unknown value.

SaveSlotIndex (unsigned int) The index of this save game. The game supports 10 saves and an autosave.

HammersUnlocked (char) Determines whether MP hammers are unlocked.

GoldenHammerDesired (bool) Unknown value.

NewData (GameSaveInfoNewData) More save data that is for some reason placed in it's own type.

Padding (char[117]) Padding. Likely placed in by the compiler or to allow future expansion of the save file format without changing it.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

GameSaveInfoNewData

More values stored in saves.

Variables

HammersUnlockedLarge (char[8]) An array of values which represent hammers unlocked. Exact purpose of each index untested.

BackpacksUnlocked (char[2]) Array representing whether or not MP backpacks are unlocked. Exact purpose of each index untested.

JetpackUnlockLevel (char) The jetpack unlock level. Exact values untested.

GeneralActionTypes

Action type of a useable object. Defined in `./Core/rfg/AnimationActions.lua`. The enum needs to be redefined on this page.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

HavokBPO

Contains data about the objects Havok state if the object is physically simulated.

Variables

Flags (char) Description needed.

State (char) Description needed.

BpoIndex (int16) Description needed.

StateIndex (int16) Description needed.

Owner (unsigned int16) Description needed.

Next (unsigned int16) Description needed.

Prev (unsigned int16) Description needed.

HdrStateBlock

Hdr state values. Access this through `rfg.Hdr`. So, for example, if you wanted to set the value of `BloomSoft`, you'd do that like so: `rfg.Hdr.BloomSoft = false`. Another example, if you wanted to set the value of `IrisRate`, you'd do `rfg.Hdr.IrisRate = 0.33`.

Variables

Override (bool) Needs description. Default value: `true`

Enable (bool) Needs description. Default value: `true`

BloomSoft (bool) Needs description. Default value: `true`

BloomAlternate (bool) Needs description. Default value: `true`

ToneMappedBloom (bool) Needs description. Default value: `true`

LuminanceRange (float) Needs description. Default value: `3.0`

LuminanceOffset (float) Needs description. Default value: `10.0`

HdrLevel (float) Needs description. Default value: `0.6`

IrisRate (float) Needs description. Default value: `3.0`

LuminanceMin (float) Needs description. Default value: `0.001`

LuminanceMax (float) Needs description. Default value: `4.0`

LuminanceMaskMax (float) Needs description. Default value: `4.0`

BrightpassThreshold (float) Needs description. Default value: `3.0`

BrightpassOffset (float) Needs description. Default value: `6.0`

UseHdrLevel (bool) Needs description. Default value: `true`

BloomNew (bool) Needs description. Default value: `true`

EyeAdaptionBase (float) Needs description. Default value: `0.2`

EyeAdaptionAmount (float) Needs description. Default value: `0.5`

EyeFadeMin (float) Needs description. Default value: `1.0`

EyeFadeMax (float) Needs description. Default value: `1.0`

BloomAmount (float) Needs description. Default value: `2.5`

BloomTheta (float) Needs description. Default value: `1.2`

BloomSlopeA (float) Needs description. Default value: `0.0`

BloomSlopeB (float) Needs description. Default value: `0.08`

LuminanceConversion (Vector) Needs description. Note that being a vector, it has `x`, `y`, and `z` values. So for example, you'd set it's `y` value like this: `rfg.Hdr.LuminanceConversion.y = 0.23`. Default value:
`x: 0.213, y: 0.715, z: 0.072`.

BloomSuperSoft (bool) Needs description. Default value: `false`

EyeOverride (bool) Needs description. Default value: `false`

HudMessageHandle

Usage unknown. Used in [Player](#). Defined in `./Core/rfg/HudMessageHandle.lua`.

Access Variable	Value
<code>rfg.HudMessageHandle.Invalid</code>	4294967295
<code>rfg.HudMessageHandle.HudMessageHandleForceTo32Bit</code>	2147483647

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

Human

Has all the data representing a human such as health, jump height, move speed, and more.

Inherits [Object](#)

- [Variables](#)
- [Functions](#)

Variables

Flags ([HumanFlags](#)) A set of [bool](#) values which determine the behavior of the human.

LastTriggerDownFrame ([int](#)) The last frame the humans weapon was fired.

Info ([HumanInfo](#)) More human specific data. May be `nil`.

RenderDistance ([ObjectRenderDistance](#)) The humans render distance.

XRayMaterial ([char](#)) The xray material of this human. They may or may not be visible via xray depending on their material.

XrayRenderAlpha ([char](#)) The transparency of this human in xray mode.

Rank ([int](#)) Unknown value.

ApproxLocalBmin ([Vector](#)) The minimum point of this humans bounding box.

ApproxLocalBmax ([Vector](#)) The maximum point of this humans bounding box.

LastPosition ([Vector](#)) The last x, y, z position.

TurnToTarget ([Vector](#)) Point that the human is turning towards.

NavCellDetourRequestHandle ([unsigned int](#)) Unknown value.

RaycastHitInfo ([HumanRaycastHitInfo](#)) Info about a raycast starting point and result. Unknown if this is just cast from the humans position or if it's from the weapon if they possess one.

Velocity ([Vector](#)) The humans velocity.

ActualVelocity ([Vector](#)) Difference from previous value unknown.

TransformFramesSkipped ([int](#)) Unknown value.

SteeringVector ([Vector](#)) Unknown value.

MoveSpeed ([float](#)) Move speed when walking and running.

SteeringHeadingOffset (float) Unknown value. Possibly an angle offset from 0 degrees.

RotateDirection (HumanRotateDirections) Determines which direction (clockwise, counter-clockwise) the human will rotate in.

StateSpeedThrottle (float) Unknown value.

RotateHeadingLeft (float) Unknown value.

RotateHeadingRight (float) Unknown value.

RotateInternalHeading (float) Unknown value.

InertialMotion (Vector) Unknown value.

GroundObjectHandle (unsigned int) Unknown value. Possibly the object handle of the ground.

GroundObjectContactPos (Vector) The point that the human is in contact with the ground.

GroundObjectAltBodyIDX (int) Unknown value.

GroundObjectShapeKey (int) Unknown value. Likely related to havok physics.

GroundObjectNormal (float) The normal vector of the point that the human is in contact with the ground.

ObjectFilterHandle (unsigned int) Unknown value.

BlockMovementAnimation (int) Unknown value.

BlockSteeringVector (Vector) Unknown value.

BlockMovementType (BlockMovementTypes) Unknown value.

BlockSteeringHeadingOffset (float) Unknown value.

ReleasedMovementAnim (int) Unknown value.

ReleasedMovementType (BlockMovementTypes) Unknown value.

CurrentState (HumanStates) The state of the human. Jumping, falling, in turret, etc. See [HumanStates](#) for more info.

CurrentMovementState (HumanMovementStates) Running, walking, idle, etc.

CurrentStance (HumanStances) Current stance. Standing, prone, crouching, etc.

LastStance (HumanStances) Last stance.

MovementMode (HumanMoveModes) The current movement mode. Pathfinding, pathfinding to vehicle, etc.

MovementSubmode (HumanMoveSubmodes) Movement submodes. Jump start, jump pre, jump mid, etc.

PreviousMovementMode (HumanMoveModes) The previous movement mode.

CustomFireAnimation (AnimationActions) Fire animation info.

WalkAnimationSpeedPercentage (float) Unknown value.

FrametimeTally (float) Unknown value.

ActualVelocityMagnitude (float) Magnitude of ActualVelocity.

MaxSpeed (float) Max speed enforced on human.

FallingVelocityMagnitude (float) Magnitude of falling velocity.

NotMovingUpdate (Timestamp) Desc... Can be `nil`.

CharInstance (CharacterInstance) Character instance info. May be `nil`.

BoneLODLevel (int) Unknown value.

WepAnimationFlags (WeaponAnimationFlags) Weapon animation flags.

RagdollState (HumanRagdollStates) Ragdoll state. Flinch, active, blend out, etc.

RagdollSourceHumanHandle (unsigned int) Unknown value.

RagdollIDX (int) Unknown value.

RagdollBlendPercentage (float) Unknown value.

RagdollLastFrameVelocity (Vector) Unknown value.

RagdollOnImpactTime (float) Unknown value.

RagdollOnImpactMinVelocity (float) Unknown value.

RagdollDamaged (Timestamp) Unknown value.

RootBoneOffset (float) Unknown value.

NanoMaterialFX (unsigned int[16]) Unknown value.

MaterialFXHandleForHeadSkin (unsigned int) Unknown value.

LastValidPositionBeforeRagdollIndex (unsigned int) Unknown value.

RagdollNumCollisionsWithWalker (int) Unknown value.

IK_Joints (IkJoint[4]) Inverse kinematics joints of the humans body.

InitialMaxHitPoints (int) Initial max hit points.

MaxHitPoints (int) Current max hit points.

HitPoints (float) Hit points.

MaxKnockdownHits (int) Unknown value.

KnockdownHits (float) Unknown value.

KnockdownTimestamp (Timestamp) Unknown value.

KnockdownTimeoutTimestamp (Timestamp) Unknown value.

CollisionDamageTimer (Timestamp) Unknown value.

CurrentCollisionDamage (float) Unknown value.

DoRagdollTimestamp (Timestamp) Unknown value.

FacialPoseTimestamp (Timestamp) Unknown value.

TurretHoldAnimations (Timestamp) Unknown value.

Cash (float) Not used by the game. Likely a remnant of Saints Row 1.

Inventory (InventoryItem) Desc... Can be `nil`.

DesiredEquippedInvItem (InventoryItem) Desc... Can be `nil`.

LastEquippedWeapon (InventoryItem) Desc... Can be `nil`.

SecondLastEquippedWeapon (InventoryItem) Desc... Can be `nil`.

GrenadeWeapon (InventoryItem) Desc... Can be `nil`.

OffhandProjectileHandle (unsigned int) Unknown value.

ShieldHandle (unsigned int) Object handle of the humans shield if they have one.

ReloadTimer (Timestamp) Timer used for weapon reload.

EquipTagIndex (int) Unknown value.

EquipOffhandTagIndex (int) Unknown value.

RootBoneIndex (int) Root bone index.

LeftFootBoneIndex (int) Left foot bone index.

RightFootBoneIndex (int) Right foot bone index.

LeftShoulderBoneIndex (int) Left shoulder bone index.

RightShoulderBoneIndex (int) Right shoulder bone index.

LeftHandBoneIndex (int) Left hand bone index.

RightHandBoneIndex (int) Right hand bone index.

LeftHipBoneIndex (int) Left hip bone index.

RightHipBoneIndex (int) Right hip bone index.

RunStandBlendPoseWeight (float) Unknown value. Likely used when blending run and stand animations when switching between the two.

LeanWeight (float) Unknown value.

LeanDirection (Vector) Direction the human is leaning in.

LeanDisabled (Timestamp) Unknown value.

LeanLastSteeringVector (Vector) Unknown value.

FootPlanted (FootPlant) Information about where and how the foot is planted.

FootPlantTime (Timestamp) Unknown value.

IsTurning (bool) Is `true` if turning.

StartingTurn (bool) Is `true` if starting to turn.

RenderAlpha (float) The humans render opacity.

CameraAlphaOverride (float) Unknown value.

CastsTransparentShadows (bool) Is `true` if it casts transparent shadows.

FadeTimer (Timestamp) Unknown value.

FadeTime (int) Unknown value.

StealthPercent (float) If set to `1.0` the human will have the stealth jetpack “invisible” effect on their body.

VehicleHandle (unsigned int) The object handle of the humans vehicle if they are in one.

ReservedVehicleHandle (unsigned int) Unknown value.

VehicleSeatIDX (VehicleSeatIndex) Unknown value.

BoredIdleTimestamp (Timestamp) Unknown value.

CorpseCleanupTimer (Timestamp) Unknown value.

TurnOffFireTimestamp (Timestamp) Unknown value.

HealthRestoreTimestamp (Timestamp) Unknown value.

HealthRestoreHitPoints (float) How many hitpoints to restore each healing tick.

HealthRestoreMinimumHitPoints (float) Unknown value.

LookAtPos (**Vector**) Unknown value.

LookAtHandle (**unsigned int**) Unknown value.

LookAtSpeed (**float**) The speed at which the human looks at things.

AimOverrideDirection (**Vector**) Unknown value.

DamagePercent (**float**) Percent of damage to hitpoints. A value of 1.0 is equal to 100%.

DamageFunctionHandle (**unsigned int16**) Unknown value.

DeathFunctionHandle (**unsigned int16**) Unknown value.

BreathTimer (**Timestamp**) Unknown value.

CrouchToStandTestTimestamp (**Timestamp**) Unknown value.

LadderHandle (**unsigned int**) The object handle of the ladder the human is attached to if they are attached to one.

LadderSlideSpeed (**float**) The speed at which the human slides down ladders.

LadderGrabRung (**int**) Unknown value.

LadderSlidePlayID (**int**) Unknown value.

CodeDrivenStartJump (**bool**) Unknown value.

CodeDrivenJumpTimer (**float**) Unknown value.

CodeDrivenJumpHeight (**float**) The humans jump height.

JumpStateTimer (**Timestamp**) Unknown value.

LastSupported (**Timestamp**) Unknown value.

AirTime (**Timestamp**) Unknown value.

UpdateTimer (**Timestamp**) Unknown value.

ImportanceLevel (**ObjectImportanceLevels**) Unknown value.

ScriptedActionNodeHandle (**unsigned int**) Unknown value.

NanoCBInfo (**NanoCallbackInfo[16]**) Unknown value.

NanoIndex (**int**) Unknown value.

LightEffects (**unsigned int[2]**) Unknown value.

LightTags (**int[2]**) Unknown value.

AvoidanceCheckTimer (**Timestamp**) Unknown value.

AvoidanceRequestTimer (**Timestamp**) Unknown value.

AvoidanceRequestHuman (**unsigned int**) Unknown value.

AvoidanceMoveDirection (**Vector**) Unknown value.

AvoidanceHintDirection (**Vector**) Unknown value.

AvoidancePauseTimer (**Timestamp**) Unknown value.

AvoidanceOriginalMovementState (**HumanMovementStates**) Unknown value.

MinimapFlags (**int**) Unknown value.

EquippedInventoryItemLastFrame (**InventoryItem**) Unknown value. May be `nil`.

EquippedInventoryItem (**InventoryItem**) Unknown value. May be `nil`.

CurrentTeam (HumanTeams) The humans team. EDF, Guerrilla, etc.

UndercoverTeam (HumanTeams) Unknown value.

DialogueFoleyInfo (int) Unknown value.

QueuedVoiceLine (VoiceLineHandle) Unknown value.

SituationalVoiceLine (VoiceLines) Unknown value.

VoicePriority (VoiceLinePriorities) Unknown value.

VoiceCuePriority (AudiolibCuePriorities) Unknown value.

RadioInstance (unsigned int16) Unknown value.

VoiceInstance (int) Unknown value.

VoiceTimeSinceFinish (Timestamp) Unknown value.

LipsyncHandle (LipsyncDataHandle) Unknown value.

VoiceDelayTime (Timestamp) Unknown value.

AcknowledgedTime (Timestamp) Unknown value.

ReportedTimer (Timestamp) Unknown value.

Lifetime (Timestamp) The time the human has been in existence.

Invincible (bool) If true the human won't take any damage. Shortcut for `someHuman.Flags.Invulnerable`

AIgnore (bool) If true, the human will be ignored by other AI. Useful for when you want the player to be ignored when walking around EDF bases. Shortcut for `someHuman.Flags.AIignore`

AllowRagdoll (bool) If false, the human won't ragdoll. Shortcut for `someHuman.Flags.DisallowFlinchesAndRagdolls`

Functions

Heal (Human Self) Fully heals the human. Since the first argument is self, you call it with a colon instead of a period.
So `someHuman:ResetMoveSpeed()`

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

HumanFlags

Additional flags used to describe the behavior of a given human.

Variables

BoredHeadtrackDisabled (bool) Unknown value.

Hidden (bool) Exact purpose is unknown. If true, odd behaviour in characters occur such as being frozen in their current state, certain parts of their model being removed, and being hidden on the minimap.

CapSpeed (bool) True if the came should limit the movement speed of this human.

WasRendered (bool) Unknown value.

LockedController (bool) Unknown value.

Invulnerable (bool) If true this human is invulnerable to all damage including killzones. In the case of the player they'll get stuck and unable to move if they fall into a killzone with this set to true.

MissionInvulnerable (bool) Unknown value.

NoDamage (bool) If set to true, this human cannot cause damage.

ActivityEngage (bool) Unknown value.

ConsideredArmed (bool) If true they're considered armed by the EDF.

RiotShield (bool) Unknown value.

SafehouseVIP (bool) Unknown value.

RadioOperator (bool) Unknown value.

ActivityRaidRequired (bool) Unknown value.

ActivityRaidOptional (bool) Unknown value.

ActivityHouseArrestHostage (bool) Unknown value.

RaidIntroductionLines (bool) Unknown value.

MinerPersonaLines (bool) Unknown value.

DamagedByPlayer (bool) True if this human was damaged by the player.

AIgnore (bool) If true, AI will ignore this player. If an AI is already in combat with this human then they'll stay aggressive even after setting this to true.

CastsShadows (bool) If true, human will cast a shadow. Objects such as weapons and accessories will still cast a shadow if false.

CastsDropShadows (bool) Unknown value.

IsTurning (bool) True if this human is turning.

IsFalling (bool) True if this human is falling.

DontDeformBones (bool) Exact purpose is unknown. If true, odd behaviour in ragdolls and characters occur.

DontLodBones (bool) Unknown value.

PlayingEquipAnim (bool) Unknown value.

PlayingUnequipAnim (bool) Unknown value.

DoInstantEquip (bool) Unknown value.

AnimDirectBlend (bool) Unknown value.

StartJump (bool) Unknown value. Setting this to true does not cause the human to jump.

SuperJump (bool) Unknown value. Setting this to true does not cause any noticeable change in jump behavior or height.

ProcessedThisFrame (bool) Unknown value.

SilentVehicleStart (bool) Unknown value.

SuppressFleeOnVehicleExit (bool) Unknown value.

DeathFuntionDone (bool) Unknown value.

FadingIn (bool) Very inconsistent at times. If FadingIn is set to true while FadingOut is also set to true, this human will be visible again.

FadingOut (bool) Very inconsistent at times. If set to true, this human will disappear with their weapons and accessories but are still collidable and visible on the minimap.

FadingOutFromNano (bool) Unknown value.

IsNanoEffectCurrentlyApplied (bool) Unknown value.

OnFire (bool) Unknown value. Untested, but it's possible this is set to true when a remote charge is thrown on a human.

DroppedCash (bool) Unknown value.

OnMover (bool) Unknown value.

RecalculateAtNode (bool) Unknown value.

SpinebendingDone (bool) Unknown value.

LastDestinationInRepulsor (bool) Unknown value. Likely has something to do with pathfinding.

JumpingFromBuilding (bool) Unknown value.

IsOnLadder (bool) True if they're on a ladder.

LadderForceSlide (bool) Unknown value.

LadderForceExit (bool) Unknown value.

LadderReEquipWeapon (bool) Unknown value.

AllowFlyingEquips (bool) Unknown value.

CorpseSpotted (bool) Unknown value.

CorpseSpottedByEnemy (bool) Unknown value.

DeathReported (bool) Unknown value.

RaidDeathAck (bool) Unknown value.

AnchorOrient (bool) Unknown value.

BonesTransformedThisFrame (bool) Unknown value.

UseCurrentVelocity (bool) Unknown value.

UseAsFinalVelocity (bool) Unknown value.

AimAtPos (bool) Unknown value.

BlockForcedMovement (bool) Unknown value.

CancellingMeleeAttack (bool) Unknown value.

DoContinousMeleeDamage (bool) Unknown value.

InAirMelee (bool) Unknown value.

CantHitWithMelee (bool) If true, this human cannot be hit with a melee weapon.

InvulnerableToDebris (bool) Unknown value.

OverrideDefaultAnimState (bool) Unknown value.

OverrideDefaultFireAnim (bool) Unknown value.

MovingAndTransitioningStates (bool) Unknown value.

CheckForCover (bool) Unknown value.

JumpTakeOff (bool) Unknown value.

RotateInternalHeadingUpdated (bool) Unknown value.

CrouchCover (bool) Unknown value.

CoverCrouchHighOnly (bool) Unknown value.

CoverCrouchNoLean (bool) Unknown value.

SideFiring (bool) Unknown value.

SideFiringWalkBack (bool) Unknown value.

RagdollOnImpactAllCollisions (bool) Unknown value.

RagdollOnImpactUseRagdollPos (bool) Unknown value.

DiveCapsule (bool) Unknown value.

MaintainAmbientProps (bool) Unknown value.

LeaningDisabled (bool) Unknown value.

OverrideSteeringHeadingOffset (bool) Unknown value.

PushesOtherHumans (bool) Unknown value.

PushesDebrisScripted (bool) Unknown value.

AllowSteepSlopes (bool) Unknown value.

ExternalForceApplied (bool) Unknown value.

RagdollShot (bool) Unknown value.

SavedPushesDebrisScripted (bool) Unknown value.

FilterHandleValid (bool) Unknown value.

JustGotUpFromRagdoll (bool) Unknown value.

DisablePathSmoothingForRequest (bool) Unknown value.

DisableAllPathSmoothing (bool) Unknown value.

InFetalPosition (bool) Unknown value.

LimitedVehicleExit (bool) Unknown value.

DriverlessExitOnly (bool) Unknown value.

StuckInVehicle (bool) Unknown value.

ConvoyVehicleExit (bool) Unknown value.

DisallowVehicleExit (bool) If true, this human will be unable to exit vehicles.

DisallowVehicleDrive (bool) If true, this human will be unable to drive vehicles.

AmbientEDF (bool) Unknown value.

BashedCharacterController (bool) Unknown value.

HeadLoaded (bool) Unknown value.

LodHeadLoaded (bool) Unknown value.

InVehicleInvisible (bool) Unknown value.

HighPriorityTarget (bool) Unknown value.

HealthChangeWasNegative (bool) Unknown value.

VoiceLinePlay2D (bool) Unknown value.
VoiceLinePainOnly (bool) Unknown value.
KilledByKillzone (bool) Unknown value.
FirstTimeStreamed (bool) Unknown value.
Tired (bool) Unknown value.
UseBigsteps (bool) Unknown value.
Stuck (bool) Unknown value.
LastPffFailed (bool) Unknown value.
ExtendedStuck (bool) Unknown value.
XrayVisible (bool) Unknown value.
WasGibbed (bool) Unknown value.
PreventRagdollSfx (bool) Unknown value.
AlwaysShowOnMinimap (bool) Unknown value.
UsedDeathBuffer (bool) Unknown value.
DoNotConvertToGuerrilla (bool) Unknown value.
DoNotPlayAmbientOrGreetLines (bool) Unknown value.
DisallowFlinchesAndRagdolls (bool) Unknown value.
OnlyUseActionNodes (bool) Unknown value.
ComplainWhenShot (bool) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

HumanInfo

Contains additional information about a human. Each human has their own unique copy of this type.

Variables

MaxHitPoints (int) Max hit points this human can have.
MaxKnockdownHits (int) The maximum number of melee hits before this human will be knocked over.
MaxSpeed (float) Max speed in m/s of this human.
DefaultEquipItem (InvItemInfo) The default item that this human has equipped.
DefaultLastEquipItem (InvItemInfo) The last item equipped.
Name (char*) The humans name.
NameCrc (unsigned int) The names crc hash.
ModelName (char*) This humans 3D model filename.
AnimSetName (char*) This humans animation set filename.

AnimSetNameCrc (**unsigned int**) The anim set filename's crc hash.

RigName (**char***) This humans rigging filename.

SRID (**int**) Likely a unique identifier for this human. Not confirmed.

SlotID (**int**) Unknown value.

DefaultHeightScale (**float**) The height scale of this human. With 1.0 being no scaling, 0.5 being half size, 2.0 being double the size, etc. Testing with this did not result in an immediate height change. A way to force rescaling may need to be found.

HeightScaleVariation (**float**) Unknown value.

DefaultTeam (**HumanTeams**) The team this human is on (EDF, Marauders, etc).

HurtThreshold (**int16**) Unknown value.

HomeDistrict (**int**) Unknown value.

AimErrorPercent (**float**) Unknown value.

FootEffects (**FootGroundEffects**) Unknown value.

NumHeads (**int**) Unknown value.

Heads (**CharDefHead*[4]**) Unknown value.

NumGeneralProps (**int**) The number of props this human has. The actual prop data has not yet been bound to lua.

LowerSpineBoneIndex (**int**) Unknown value.

UpperSpineBoneIndex (**int**) Unknown value.

PelvisBoneIndex (**int**) Unknown value.

HeadBoneIndex (**int**) Unknown value.

NeckBoneIndex (**int**) Unknown value.

EyeLeftBoneIndex (**int**) Unknown value.

EyeRightBoneIndex (**int**) Unknown value.

CalfLeftBoneIndex (**int**) Unknown value.

CalfRightBoneIndex (**int**) Unknown value.

CameraBoneIndex (**int**) Unknown value.

DropCashMin (**int**) Unknown value.

DropCashMax (**int**) Unknown value.

LightEffectHandle (**unsigned int**) Unknown value.

LightTagNames (**char*[2]**) Unknown value.

Flags (**HumanInfoFlags**) More flags used to describe this humans behavior.

HumanInfoFlags

More flags which describe a humans conditions.

Variables

Female (Bitfield) Unknown value. Range: -128 to 127

Coin (Bitfield) Unknown value. Range: -128 to 127

Preload (Bitfield) Unknown value. Range: -128 to 127

Officer (Bitfield) Unknown value. Range: -128 to 127

Unarmed (Bitfield) Set to 1 if unarmed. Range: -128 to 127

VIP (Bitfield) Unknown value. Range: -128 to 127

UseTechLevel (Bitfield) Unknown value. Range: -128 to 127

Sniper (Bitfield) Set to 1 if a sniper EDF soldier. Range: -128 to 127

EliteDodge (Bitfield) Unknown value. Range: -128 to 127

RiotShield (Bitfield) If set to 1 this human will have a riot shield in one of their hands. Works for the player, yet they won't use the proper shield animation so it'll behave like an armored gauntlet on one of their arms. Range: -128 to 127

BodyArmor (Bitfield) Unknown value. Range: -128 to 127

NonFlammable (Bitfield) Unknown value. Range: -128 to 127

NoAttachingProjectiles (Bitfield) Unknown value. Range: -128 to 127

GuerrillaTech (Bitfield) Unknown value. Range: -128 to 127

GuerrillaRfc (Bitfield) Unknown value. Range: -128 to 127

OldCoot (Bitfield) Set to 1 if the human is Old Coot. Range: -128 to 127

Dan (Bitfield) Set to 1 if the human is Dan. Range: -128 to 127

Jenkins (Bitfield) Set to 1 if the human is Jenkins. Range: -128 to 127

DriverlessExitOnly ('bool') Unknown value.

HumanMoveModes

The current move mode of a **Human**. Defined in `./Core/rfg/HumanMoveModes.lua`.

Access Variable	Value
<code>rfg.HumanMoveModes.None</code>	0
<code>rfg.HumanMoveModes.Direct</code>	1
<code>rfg.HumanMoveModes.Pathfinding</code>	2
<code>rfg.HumanMoveModes.PathfindingToVehicle</code>	3
<code>rfg.HumanMoveModes.Jump</code>	4
<code>rfg.HumanMoveModes.Fall</code>	5
<code>rfg.HumanMoveModes.FpMove</code>	6
<code>rfg.HumanMoveModes.Knockdown</code>	7
<code>rfg.HumanMoveModes.Satellite</code>	8
<code>rfg.HumanMoveModes.NumHumanMoveModes</code>	9

HumanMoveSubmodes

The current move submode of a **Human**. Defined in `./Core/rfg/HumanMoveSubmodes.lua`.

Access Variable	Value
<code>rfg.HumanMoveSubmodes.None</code>	0
<code>rfg.HumanMoveSubmodes.JumpStart</code>	1
<code>rfg.HumanMoveSubmodes.JumpPre</code>	2
<code>rfg.HumanMoveSubmodes.JumpMid</code>	3
<code>rfg.HumanMoveSubmodes.FallStart</code>	4
<code>rfg.HumanMoveSubmodes.FallNormal</code>	5
<code>rfg.HumanMoveSubmodes.FallWorry</code>	6
<code>rfg.HumanMoveSubmodes.FallPanic</code>	7
<code>rfg.HumanMoveSubmodes.FallFree</code>	8
<code>rfg.HumanMoveSubmodes.Land</code>	9
<code>rfg.HumanMoveSubmodes.KnockdownRagdoll</code>	10
<code>rfg.HumanMoveSubmodes.KnockdownGetUp</code>	11
<code>rfg.HumanMoveSubmodes.KnockdownTeleport</code>	12
<code>rfg.HumanMoveSubmodes.Pathfinding</code>	13
<code>rfg.HumanMoveSubmodes.GetUp</code>	14
<code>rfg.HumanMoveSubmodes.FinalAdjustment</code>	15
<code>rfg.HumanMoveSubmodes.CoverRight</code>	16
<code>rfg.HumanMoveSubmodes.CoverRightEdge</code>	17
<code>rfg.HumanMoveSubmodes.CoverLeft</code>	18
<code>rfg.HumanMoveSubmodes.CoverLeftEdge</code>	19
<code>rfg.HumanMoveSubmodes.CoverRightLean</code>	20
<code>rfg.HumanMoveSubmodes.CoverLeftLean</code>	21
<code>rfg.HumanMoveSubmodes.CoverRightLeanUp</code>	22
<code>rfg.HumanMoveSubmodes.CoverLeftLeanUp</code>	23
<code>rfg.HumanMoveSubmodes.CoverRightBlindfire</code>	24
<code>rfg.HumanMoveSubmodes.CoverLeftBlindfire</code>	25
<code>rfg.HumanMoveSubmodes.CoverRightBlindfire2</code>	26
<code>rfg.HumanMoveSubmodes.CoverLeftBlindfire2</code>	27
<code>rfg.HumanMoveSubmodes.NumHumanMoveSubmodes</code>	28

HumanMovementStates

The current movement state of a **Human**. Defined in `./Core/rfg/HumanMovementStates.lua`.

Access Variable	Value
<code>rfg.HumanMovementStates.Current</code>	4294967295
<code>rfg.HumanMovementStates.Idle</code>	0
<code>rfg.HumanMovementStates.Walk</code>	1
<code>rfg.HumanMovementStates.Run</code>	2
<code>rfg.HumanMovementStates.Sprint</code>	3
<code>rfg.HumanMovementStates.NumHumanMovementStates</code>	4

HumanRagdollStates

The current ragdoll state of a **Human**. Defined in `./Core/rfg/HumanRagdollStates.lua`.

Access Variable	Value
rfg.HumanRagdollStates.None	0
rfg.HumanRagdollStates.Active	1
rfg.HumanRagdollStates.BlendOut	2
rfg.HumanRagdollStates.Flinch	3
rfg.HumanRagdollStates.ActiveOnImpact	4
rfg.HumanRagdollStates.Deactivated	5
rfg.HumanRagdollStates.PoseSaved	6
rfg.HumanRagdollStates.NumHumanRagdollStates	7

HumanRaycastHitInfo

Stores raycast data used for collision detection of weapons.

Variables

FirePosition (Vector) 3D position that the weapon was fired from.

TargetPos (Vector) 3D position that the weapon was aiming at.

HitPoint (Vector) 3D position that the raycast impacted at.

HitLocation (ObjectHitLocations) If the ray hit a human this represents the location on their body it impacted. This is an enum value.

HitBone (int) If the ray hit a human this will be the index of the bone it hit.

HumanRotateDirections

Used in [Humans](#). Defined in `./Core/rfg/HumanRotateDirections.lua`.

Access Variable	Value
rfg.HumanRotateDirections.Shortest	0
rfg.HumanRotateDirections.Clockwise	1
rfg.HumanRotateDirections.CounterClockwise	2
rfg.HumanRotateDirections.UseLimits	3

HumanStances

The current stance of a [Human](#). Defined in `./Core/rfg/HumanStances.lua`.

Access Variable	Value
rfg.HumanStances.Current	4294967295
rfg.HumanStances.Stand	0
rfg.HumanStances.Crouch	1
rfg.HumanStances.Prone	2
rfg.HumanStances.NumHumanStances	3

HumanStates

The current state of a **Human**. Defined in `./Core/rfg/HumanStates.lua`.

Access Variable	Value
<code>rfg.HumanStates.Current</code>	4294967295
<code>rfg.HumanStates.Standard</code>	0
<code>rfg.HumanStates.Jump</code>	1
<code>rfg.HumanStates.Fall</code>	2
<code>rfg.HumanStates.Turret</code>	3
<code>rfg.HumanStates.Dead</code>	4
<code>rfg.HumanStates.Ladder</code>	5
<code>rfg.HumanStates.PlayerPathing</code>	6
<code>rfg.HumanStates.VehicleEntering</code>	7
<code>rfg.HumanStates.VehicleSitting</code>	8
<code>rfg.HumanStates.VehicleDriving</code>	9
<code>rfg.HumanStates.VehicleExiting</code>	10
<code>rfg.HumanStates.Spectator</code>	11
<code>rfg.HumanStates.Cover</code>	12
<code>rfg.HumanStates.AmmoBox</code>	13
<code>rfg.HumanStates.MpDefuse</code>	14
<code>rfg.HumanStates.ActionNode</code>	15
<code>rfg.HumanStates.Satellite</code>	16
<code>rfg.HumanStates.NumHumanStates</code>	17

HumanTeams

Used to determine the team/faction of a **Human**. Defined in `./Core/rfg/HumanTeams.lua`.

Access Variable	Value
<code>rfg.HumanTeams.None</code>	4294967295
<code>rfg.HumanTeams.Guerrilla</code>	0
<code>rfg.HumanTeams.EDF</code>	1
<code>rfg.HumanTeams.Civilian</code>	2
<code>rfg.HumanTeams.MPNeutral</code>	2
<code>rfg.HumanTeams.Marauder</code>	3
<code>rfg.HumanTeams.MPSpectator</code>	3
<code>rfg.HumanTeams.NumHumanTeams</code>	4

IkJoint

Inverse kinematics joint used when animating **‘Humans’_**.

Variables

AnchorTag (int) Unknown value.

JointTag (int) Unknown value.

IkTag (int) Unknown value.

IkStrength (float) Unknown value.

IkRateOfChange (float) Unknown value.

FreezeObjHandle (unsigned int) Unknown value.

FreezeOffsetPos (Vector) Unknown value.

Location (IkTypes) Unknown value.

IkTypes

Describes the type of **IkJoint**. Defined in `./Core/rfg/IkTypes.lua`.

Access Variable	Value
<code>rfg.IkTypes.LeftHand</code>	0
<code>rfg.IkTypes.RightHand</code>	1
<code>rfg.IkTypes.LeftFoot</code>	2
<code>rfg.IkTypes.RightFoot</code>	3
<code>rfg.IkTypes.NumIkTypes</code>	4

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

InvItemInfo

Holds data defining the behavior of an inventory item such as remote charges or the nanorifle. A variable containing weapon info has not been bound to lua yet due to time constraints and so is unlisted here.

Variables

Name (char*) Name of the inventory item.

NameChecksum (ChecksumStri) Checksum of the items name.

DisplayName (char*) Name used in menus and the hud.

Cost (int) Unknown value.

DefaultCount (int) Default amount of this item to be carried by the player.

MaxItem (int) Max amount of this item the player can carry.

Description (char*) Description of the items behavior and appearance.

ItemOrder ('char' _) Unknown value.

InventoryItem

Holds data representing an inventory item.

Variables

Next (**InventoryItem**) Next inventory item. May be `nil`.

Prev (**InventoryItem**) Previous inventory item. May be `nil`.

Info (**InvItemInfo**) Additional info about the item. May be `nil`.

Count (**int**) Item count.

SelectionSlot (**int**) Unknown value.

AttachmentProp (**int**) Unknown value.

WeaponHandle (**unsigned int**) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

LightingStateBlock

Lighting state values. Access this through `rfg.Lighting`. So, for example, if you wanted to set the value of `IndirectLightScale`, you'd do that like so: `rfg.Lighting.IndirectLightScale = 0.7`.

Variables

SubstanceDiffuseScaleEnabled (**bool**) Needs description. Default value: `true`

CoronaAdaptionRate (**float**) Needs description. Default value: `0.6`

SubstanceSpecAlphaScale (**float**) Needs description. Default value: `7.0`

SubstanceSpecPowerScale (**float**) Needs description. Default value: `16.0`

SubstanceFresnelAlphaScale (**float**) Needs description. Default value: `1.0`

SubstanceFresnelPowerScale (**float**) Needs description. Default value: `0.6`

GlassFresnelBias (**float**) Needs description. Default value: `0.3`

GlassFresnelPower (**float**) Needs description. Default value: `1.4`

GlassDirtEnabled (**bool**) Needs description. Default value: `true`

GlassReflectionEnabled (**bool**) Needs description. Default value: `true`

IndirectLightEnabled (**bool**) Needs description. Default value: `true`

AmbientSpecularEnabled (**bool**) Needs description. Default value: `true`

IndirectLightScale (**float**) Needs description. Default value: `0.019`

IndirectLightAmount (**float**) Needs description. Default value: `0.5`

MainLightColorScale (**float**) Needs description. Default value: `0.8`

AmbientColorScale (**float**) Needs description. Default value: `0.65`

BackAmbientColorScale (**float**) Needs description. Default value: `0.29`

AmbientSpecularScale (**float**) Needs description. Default value: `0.57`

EnabledHemisphericAmbient (**bool**) Needs description. Default value: `true`

LipsyncDataHandle

Unknown purpose/usage. Defined in `./Core/rfg/LipsyncDataHandle.lua`.

Access Variable	Value
<code>rfg.LipsyncDataHandle.LipsyncDataHandle</code>	4294967295
<code>rfg.LipsyncDataHandle.LipsyncDataHandleForceTo32Bit</code>	2147483647

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

LodInfo

Data used for lod (level of detail) system.

Variables

Dist (float) Likely to be the distance at which the game switches between low and high quality textures or models for the object that owns this.

Matrix

A 3x3 matrix. Commonly used to represent orientation of objects and also used to represent the orientation of the player camera. Supports the following operators: `+`, `-`, `==`.

- *Variables*
- *Functions*

Variables

rvec (Vector) The right vector. For the free cam this points directly right of the forward vector (90° offset).

uvec (Vector) The up vector. For the free cam this points up from the forward vector (90° offset).

fvec (Vector) The forward vector. For the free cam this points the direction the camera is pointing.

Functions

new (Matrix CopyMatrix) Creates a new matrix and sets it's values to the values of `CopyMatrix`.

new (float InitialValue) Creates a new matrix and sets it's values to `InitialValue`.

new (Vector Right, Vector Up, Vector Forward) Creates a new matrix and sets it's vectors directly with `Right`, `Up`, `Forward`.

SetAll (float Value) Sets all the Matrices variables to `Value`

Matrix43

A 4x3 matrix. Can contain both rotation and translation info. Supports the following operators: +, -, ==.

- *Variables*
- *Functions*

Variables

Rotation ([Matrix](#)) Rotation data. See the page on [Matrix](#) for more info.

Translation ([Vector](#)) Translation data. See the page on [Vector](#) for more info.

Functions

new ([Matrix43](#) [CopyMatrix](#)) Creates a new Matrix43 and sets it's values to the values of CopyMatrix.

new ([float](#) [InitialValue](#)) Creates a new Matrix43 and sets it's values to InitialValue.

new ([Matrix](#) [InitialRotation](#), [Vector](#) [InitialTranslation](#)) Creates a new Matrix43 and sets Rotation and Translation to InitialRotation and InitialTranslation respectively.

SetAll ([float](#) [Value](#)) Sets all the values of Self.Rotation and Self.Translation to Value

MessageBoxChoices

Used by [rfg.AddMessageBox](#) so message box callback functions can receive user input on yes/no or accept/cancel style message boxes. These are defined in `./Core/rfg/MessageBoxChoices.lua`.

Access Variable	Value
<code>rfg.MessageBoxChoices.Ok</code>	0
<code>rfg.MessageBoxChoices.Yes</code>	0
<code>rfg.MessageBoxChoices.Accept</code>	0
<code>rfg.MessageBoxChoices.Abort</code>	1
<code>rfg.MessageBoxChoices.No</code>	1
<code>rfg.MessageBoxChoices.Cancel</code>	1
<code>rfg.MessageBoxChoices.ForcedExit</code>	2
<code>rfg.MessageBoxChoices.Invalid</code>	3

MessageBoxTypes

Used by [rfg.AddMessageBox](#) when creating message boxes. These are defined in `./Core/rfg/MessageBoxTypes.lua`.

Access Variable	Value
<code>rfg.MessageBoxTypes.Invalid</code>	4294967295
<code>rfg.MessageBoxTypes.Ok</code>	0
<code>rfg.MessageBoxTypes.AcceptCancel</code>	1
<code>rfg.MessageBoxTypes.YesNo</code>	2
<code>rfg.MessageBoxTypes.LoadingCancel</code>	6
<code>rfg.MessageBoxTypes.AcceptCancelSelect</code>	7

MessageTypes

Used by `rfg.AddUserMessage` when creating new user messages. The exact differences between these have not been fully explored, but you should use `rfg.MessageType.Other` if you don't want the message to be positioned based on other visible messages. These are defined in `./Core/rfg/MessageTypes.lua`.

Access Variable	Value
<code>rfg.MessageTypes.Swap</code>	0
<code>rfg.MessageTypes.MpKill</code>	1
<code>rfg.MessageTypes.MpKillShadow</code>	2
<code>rfg.MessageTypes.Other</code>	3

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

MiscStateBlock

Misc graphics state values. Access this through `rfg.Misc`. So, for example, if you wanted to set the value of `FxaaEnabled`, you'd do that like so: `rfg.Misc.FxaaEnabled = true`.

Variables

FxaaEnabled (bool) Needs description. Default value: `true`

DecalDepthBias (float) Needs description. Default value: `-0.0002`

AlphaDistEnabled (bool) Needs description. Default value: `true`

AlphaDistStart (float) Needs description. Default value: `65.0`

AlphaDistEnd (float) Needs description. Default value: `75.0`

DistortionScale (float) Needs description. Default value: `0.09`

DistortionBlurScale (float) Needs description. Default value: `2.00`

DistortionDepthBias (float) Needs description. Default value: `-0.0002`

NanoCallbackInfo

Unconfirmed usage. Seems to be used to track the if a given `Human` has been effected by the nano rifle or any nanoforge type weapon.

Variables

TargetHandle (int) Unknown value. Likely an object handle.

KilledByNano (bool) Unknown value.

HumanNanoIndex (unsigned int) Unknown value.

Object

Nearly all things you can see in the game, and even some things you can't see, like spawn points or trigger regions, are derived from this type.

- *Variables*
- *Functions*

Variables

Position (Vector) The objects x, y, z position.

Orientation (Matrix) A 3x3 matrix which describes the orientation of the object. Consists of a right vector, up vector, and forward vector.

Child (Object) Pointer to one of the objects children. Is nil if it has no children.

ChildNext (Object) Shortcut for `Self.Child.FlaggedNext`. Is nil if no children

ChildPrevious (Object) Shortcut for `Self.Child.FlaggedPrevious`. Is nil if no children.

AttachInfo (AttachInfoData) Data about how this object is attached to other objects if applicable. May be nil if it is not.

HavokHandle (unsigned int) This may be the handle for the objects havok rigid body (unconfirmed). This handle is different from an object handle as havok rigid bodies are not objects.

ContactInfo (ObjectContactInfo) Data about all objects this object is in contact with.

ObjFlags (ObjectFlags) A set of flags used to track object behavior and state.

CheckingReset (int) Unknown purpose.

NameIndex (int16) Used by the game to access to the objects name on a as of yet unlocated list of object names.

FlaggedNext (Object) Pointer to another object, forming a doubly linked list of "flagged" objects. It's unknown what is meant by them being flagged. May be nil.

FlaggedPrevious (Object) Pointer to the previous flagged object. May be nil.

Handle (unsigned int) The objects handle. Handles are often used in place of a pointer to the object all across the game. Use `rfg.GetObjectByHandle` to gain access to the object who matches that handle.

Parent (unsigned int) The handle of the objects parent object if it has one.

BPOHandle (HavokBPO) Contains state data used by havok physics engine. May be nil.

AllIndex (int16) The index of the object on `rfg.ActiveWorld.AllObjects`

TypeIndex (int16) Used by the game to track the objects type. Exact usage/purpose unknown.

SubtypeIndex (int16) Used by the game to track the objects subtype. Exact usage/purpose unknown.

Type (char) The objects type. Should match one of the values in `rfg.ObjectTypes` a table of object types and their corresponding integer values.

SubType (char) The objects subtype. Should match one of the values in `rfg.ObjectSubTypes` a table of object subtypes and their corresponding integer values.

LastKnownBMin (Vector) One corner of the objects bounding box. A 3D point.

LastKnownBMax (Vector) The other corner of the objects bounding box. A 3D point.

SRID (unsigned int) Exact purpose unknown. Might to be a unique integer identifying the object (unconfirmed).

Functions

Warning: Be sure to check an objects type before casting it to avoid having garbage values or unintentionally changing adjacent memory. For example, if you are casting an object to [Human](#), double check that `Self.Type == rfg.ObjectTypes.Human` is true before casting it.

GetName (Object Self) Returns the objects name if it has one. Objects almost never have a name.

IsPlayer (Object Self) Returns a **'bool'** (true or false) for if this object is the player.

CastToHuman (Object Self) Returns this object as a [Human](#).

CastToPlayer (Object Self) Returns this object as a [Player](#).

CastToZone (Object Self) Returns this object as a [WorldZone](#).

CastToVehicle (Object Self) Returns this object as a [Vehicle](#).

CastToFlyer (Object Self) Returns this object as a [Flyer](#).

ObjectContactInfo

Stores contact node type for an object.

Variables

ContactList (ContactNode) Contact node data. `nil` if the object is not in contact with anything.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

ObjectFlags

A set of flags used to describe an objects state and determines how the game treats it.

Note: The names of these variables are the same names the game uses for them and they may not all have an obvious or immediate effect. You may also find that one of the descriptions here is innaccurate. Feel free to update the page with more accurate info.

Variables

FlaggedListState (Bitfield) Range: -4 to 3

LightingSetOnce (Bitfield) Range: 0 to 1

Destroyed (Bitfield) Set to 1 if the object has been destroyed. Range: 0 to 1

NoSave (Bitfield) Range: 0 to 1

ForceFullSave (Bitfield) Range: 0 to 1

DestroyOnStream (Bitfield) Unconfirmed behavior. Based on the name the object is likely destroyed and set to defaults when it's streamed in for the first time. Range: 0 to 1

CreatedByMissionOrActivity (Bitfield) Set to 1 if the object was created by a mission or activity. Range: 0 to 1

DontTransform (Bitfield) Range: 0 to 1

WorldFree (Bitfield) Range: 0 to 1

Streaming (Bitfield) Set to 1 if the object is being streamed. Range: 0 to 1

Streamed (Bitfield) Set to 1 if the object is done being streamed. Range: 0 to 1

Persistent (Bitfield) Range: 0 to 1

Original (Bitfield) Range: 0 to 1

Stub (Bitfield) Range: 0 to 1

PreserveHandle (Bitfield) Range: 0 to 1

BpoIndex (Bitfield) Range: -2 to 1

IsDependent (Bitfield) Range: 0 to 1

Visited (Bitfield) Range: 0 to 1

SpecialLifetime (Bitfield) Range: 0 to 1

SerializeProtected (Bitfield) Range: 0 to 1

DontUseMe (Bitfield) Range: 0 to 1

StreamingFixed (Bitfield) Range: 0 to 1

RenderFlags (Bitfield) Range: -8 to 7

ObjectHitLocations

Used when raycasting to represent the location the ray hit. Defined in `./Core/rfg/ObjectHitLocations.lua`.

Access Variable	Value
<code>rfg.ObjectHitLocations.None</code>	4294967295
<code>rfg.ObjectHitLocations.General</code>	0
<code>rfg.ObjectHitLocations.Torso</code>	1
<code>rfg.ObjectHitLocations.Pelvis</code>	2
<code>rfg.ObjectHitLocations.LeftArm</code>	3
<code>rfg.ObjectHitLocations.RightArm</code>	4
<code>rfg.ObjectHitLocations.Head</code>	5
<code>rfg.ObjectHitLocations.LeftLeg</code>	6
<code>rfg.ObjectHitLocations.RightLeg</code>	7
<code>rfg.ObjectHitLocations.NumHitLocations</code>	8

ObjectImportanceLevels

The importance level of an [Object](#). Defined in `./Core/rfg/ObjectImportanceLevels.lua`.

Access Variable	Value
rfg.ObjectImportanceLevels.Unknown	4294967295
rfg.ObjectImportanceLevels.None	0
rfg.ObjectImportanceLevels.Low	1
rfg.ObjectImportanceLevels.Medium	2
rfg.ObjectImportanceLevels.High	3
rfg.ObjectImportanceLevels.NumObjectImportanceLevels	4

ObjectRenderDistance

Stores an objects render distance.

Variables

ApparentDistance (float) Distance the object should be rendered at.

LastFrameProcessed (int) Untested, likely the frame number that the object was last processed.

ObjectSpawnPriorities

Enum for spawn priorities. Defined in `./Core/rfg/ObjectSpawnPriorities.lua`.

Access Variable	Value
rfg.ObjectSpawnPriorities.Invalid	4294967295
rfg.ObjectSpawnPriorities.Low	0
rfg.ObjectSpawnPriorities.Medium	1
rfg.ObjectSpawnPriorities.High	2
rfg.ObjectSpawnPriorities.Special	3

ObjectSubTypes

Another value that can be used to differentiate objects from each other and figure out what specifically they are. A good example of where this will be useful (once vehicles are binded to lua) is figuring out what type of vehicle an object is (flyer, walker, automobile). These are defined in `./Core/rfg/ObjectSubTypes.lua`. You can get an objects subtype via `Object.SubType`. A simple example of this would be counting the number of flyers and walkers, which both have the object type of vehicle, but different subtypes.

```

WalkerCount = 0
FlyerCount = 0
for i=0, rfg.ActiveWorld.AllObjects:Size(), 1 do
    Object = rfg.ActiveWorld.AllObjects[i]
    if Object.Type == rfg.ObjectTypes.Vehicle then
        if Object.SubType == rfg.ObjectSubTypes.Flyer then
            FlyerCount = FlyerCount + 1
        elseif Object.SubType == rfg.ObjectSubTypes.Walker then
            WalkerCount = WalkerCount + 1
        end
    end
end
end

```

(continues on next page)

(continued from previous page)

```
rsl.Log("Number of flyer objects found: {}\\n", FlyerCount)
rsl.Log("Number of walker objects found: {}\\n", WalkerCount)
```

Object Type	Access Variable	Value
GeneralMover	rfg.ObjectSubTypes.MoverGeneral	0
RfgMover	rfg.ObjectSubTypes.MoverRfg	1
Npc	rfg.ObjectSubTypes.HumanNpc	2
Player	rfg.ObjectSubTypes.HumanPlayer	3
Automobile	rfg.ObjectSubTypes.VehicleAuto	4
Flyer	rfg.ObjectSubTypes.VehicleFlyer	5
Walker	rfg.ObjectSubTypes.VehicleWalker	6
Weapon	rfg.ObjectSubTypes.ItemWeapon	7
Projectile	rfg.ObjectSubTypes.ItemProjectile	8
MultiObjectFlag	rfg.ObjectSubTypes.ItemMultiFlag	9
MultiObjectFlag	rfg.ObjectSubTypes.ItemMultiBackpack	10
N/A	rfg.ObjectSubTypes.NumObjectSubTypes	11
N/A	rfg.ObjectSubTypes.Invalid	4294967295

ObjectTypes

These values are used to differentiate each object type from each other. These are defined in `./Core/rfg/ObjectTypes.lua`. You can get an objects type via `Object.Type`. So, for example, you could loop through the global object list and count the number of human objects:

```
HumanCount = 0
for i=0, rfg.ActiveWorld.AllObjects:Size(), 1 do
    Object = rfg.ActiveWorld.AllObjects[i]
    if Object.Type == rfg.ObjectTypes.Human then
        HumanCount = HumanCount + 1
    end
end
rsl.Log("Number of human objects found: {}\\n", HumanCount)
```

Object Type	Access Variable	Value
Human	rfg.ObjectTypes.Human	0
Item	rfg.ObjectTypes.Item	1
Mover	rfg.ObjectTypes.Mover	2
Vehicle	rfg.ObjectTypes.Vehicle	3
Effect	rfg.ObjectTypes.Effect	4
Debris	rfg.ObjectTypes.Debris	5
Turret	rfg.ObjectTypes.Turret	6
Light	rfg.ObjectTypes.Light	7
PlayerStart	rfg.ObjectTypes.PlayerStart	8
CoverNode	rfg.ObjectTypes.CoverNode	9
NavPoint	rfg.ObjectTypes.NavPoint	10
Squad	rfg.ObjectTypes.Squad	11
Convoy	rfg.ObjectTypes.Convoy	12
ConvoyEnd	rfg.ObjectTypes.ConvoyEnd	13
Patrol	rfg.ObjectTypes.Patrol	14

Continued on next page

Table 1 – continued from previous page

Object Type	Access Variable	Value
GuardNode	rfg.ObjectTypes.GuardNode	15
Skybox	rfg.ObjectTypes.Skybox	16
Ladder	rfg.ObjectTypes.Ladder	17
Constraint	rfg.ObjectTypes.Constraint	18
Zone	rfg.ObjectTypes.Zone	19
TriggerRegion	rfg.ObjectTypes.TriggerRegion	20
MarauderAmbushRegion	rfg.ObjectTypes.MarauderAmbushRegion	21
RestrictedArea	rfg.ObjectTypes.RestrictedArea	22
SpawnRegion	rfg.ObjectTypes.SpawnRegion	23
AmbientSpawnRegion	rfg.ObjectTypes.AmbientSpawnRegion	24
VehicleSpawnRegion	rfg.ObjectTypes.VehicleSpawnRegion	25
NpcSpawnNode	rfg.ObjectTypes.NpcSpawnNode	26
TurretSpawnNode	rfg.ObjectTypes.TurretSpawnNode	27
ActionNode	rfg.ObjectTypes.ActionNode	28
SquadSpawnNode	rfg.ObjectTypes.SquadSpawnNode	29
RoadblockNode	rfg.ObjectTypes.RoadblockNode	30
ShapeCutter	rfg.ObjectTypes.ShapeCutter	31
District	rfg.ObjectTypes.District	32
MultiMarker	rfg.ObjectTypes.MultiMarker	33
PathRoad	rfg.ObjectTypes.PathRoad	34
LightParams	rfg.ObjectTypes.LightParams	35
Dummy	rfg.ObjectTypes.Dummy	36
ActivitySpawn	rfg.ObjectTypes.ActivitySpawn	37
RaidNode	rfg.ObjectTypes.RaidNode	38
Subzone	rfg.ObjectTypes.Subzone	39
HouseArrestNode	rfg.ObjectTypes.HouseArrestNode	40
DemolitionsMasterNode	rfg.ObjectTypes.DemolitionsMasterNode	41
RidingShotgunNode	rfg.ObjectTypes.RidingShotgunNode	42
DeliveryNode	rfg.ObjectTypes.DeliveryNode	43
BoundingBox	rfg.ObjectTypes.BoundingBox	44
MissionStartNode	rfg.ObjectTypes.MissionStartNode	45
Courier	rfg.ObjectTypes.Courier	46
CourierEnd	rfg.ObjectTypes.CourierEnd	47
Safehouse	rfg.ObjectTypes.Safehouse	48
BftpNode	rfg.ObjectTypes.BftpNode	49
AirstrikeDefenseNode	rfg.ObjectTypes.AirstrikeDefenseNode	50
UpgradeNode	rfg.ObjectTypes.UpgradeNode	51
AreaDefenseNode	rfg.ObjectTypes.AreaDefenseNode	52
N/A	rfg.ObjectTypes.Undefined	4294967295

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

Player

Holds the variables representing a player. You can use `rfg.ActivePlayer` or `Player` to easily access the current player. This only becomes available after a save is loaded. This type is an **Object** since it inherits **Human** which inherits **Object**.

Inherits [Human](#), which inherits [Object](#)

- [Variables](#)
- [Functions](#)

Variables

FrametimeMultiplier (**float**) Unknown value.

ActionObject (**UseableObject**) Unknown value.

ActionObjectTimestamp (**Timestamp**) Unknown value.

LastFireTime (**Timestamp**) Unknown value.

RadioId (**int**) Unknown value.

TagTrigger (**int**) Unknown value.

TagSeq (**TaggingSequences**) Unknown value.

TagTimer (**Timestamp**) Unknown value.

TagEffect (**int**) Unknown value.

HudMessage (**HudMessageHandle**) Unknown value.

HavokFilterGroup (**unsigned int**) Unknown value.

AimTarget (**unsigned int**) Unknown value.

PenetratingAimTarget (**unsigned int**) Unknown value.

PenetratingAimPos (**Vector**) Unknown value.

AimTargetDuration (**Timestamp**) Unknown value.

HeatSeekingTarget (**unsigned int**) Unknown value.

AimPos (**Vector**) Unknown value.

CombatTargetUpdateTime (**Timestamp**) Unknown value.

RemoteChargeTime (**float**) Unknown value.

DoingRemoteCharge (**bool**) Unknown value.

ZoomState (**PlayerZoomState**) Unknown value.

PreviousZoomState (**PlayerZoomState**) Unknown value.

CoverVector (**Vector**) Unknown value.

CoverVectorRight (**Vector**) Unknown value.

EnteringCover (**Timestamp**) Unknown value.

ExitingCover (**Timestamp**) Unknown value.

CoverMoveVel (**float**) Unknown value.

CoverApproachingEdge (**bool**) Unknown value.

CoverApproachingEdgeTs (**Timestamp**) Unknown value.

CoverEdgeHitPos (**Vector**) Unknown value.

CoverEdgeMissPos (**Vector**) Unknown value.

ThrownWeaponCheckPos (Vector) Unknown value.
CombatMoveQueue (int) Unknown value.
PreviousBulletHit (TimestampPercent) Unknown value.
JetpackFuelPercent (float) Unknown value.
JetpackFoley (int) Unknown value.
JetpackEffect (unsigned int) Unknown value.
AllowRagdoll (bool) Unknown value.
PlayerFlags (PlayerFlags) Unknown value.
HoldObjectAvailable (bool) Unknown value.
HoldObjectHandle (unsigned int) Unknown value.
NumAbandonedVehicles (int) Unknown value.
DeadTimestamp (TimestampPercent) Unknown value.
MatFxHandle[16] (unsigned int) Unknown value.
CurrentMatFx (unsigned int) Unknown value.
ScriptMode (PlayerScriptMode) Unknown value.
ScriptData (ScriptSpecificData) Unknown value.
JetpackUseLogHandle (int) Unknown value.
BackpackEquipLogHandle (int) Unknown value.
DistrictLogHandle (int) Unknown value.
CameraBoneTransform (Matrix43) Unknown value.
MovementVelocity (Vector) Unknown value.
DirectControlDir (Vector2) Unknown value.
DirectControlVel (Vector) Unknown value.
CoverCollisionNormal (Vector) Unknown value.
LastFrameCoverCollisionNormal (Vector) Unknown value.
CoverTestQueue (PlayerCoverTest) Unknown value.
CoverDelayTs (Timestamp) Unknown value.
CoverEnterTs (Timestamp) Unknown value.
CoverExitDelayTs (Timestamp) Unknown value.
CoverDiveCapsule (Timestamp) Unknown value.
CoverLeanDelay (Timestamp) Unknown value.
CoverRootOffset (float) Unknown value.
SpinebendRootOffset (float) Unknown value.
FacialIdle (Timestamp) Unknown value.
FootstepPropagateTime (Timestamp) Unknown value.
BumpIntoPropagateTime (Timestamp) Unknown value.

AmmoBoxHandle (**unsigned int**) Unknown value.

PathfindTimeOut (**Timestamp**) Unknown value.

CurrentInterfaceMode (**InterfaceModes**) Unknown value.

CommonControlsAllowed (**unsigned int**) Unknown value.

ControlsGeneralAllowed (**unsigned int**) Unknown value.

ControlsOnFootAllowed (**unsigned int**) Unknown value.

ControlsDrivingAllowed (**unsigned int**) Unknown value.

GeneralActionsAllowed (**unsigned int**) Unknown value.

WeaponSwapTimestamp (**Timestamp**) Unknown value.

SprintDelayTimestamp (**Timestamp**) Unknown value.

SprintStartTimestamp (**Timestamp**) Unknown value.

JumpRefreshTimestamp (**Timestamp**) Unknown value.

AllySquadHandle (**unsigned int**) Unknown value.

EscortSquadHandle (**unsigned int**) Unknown value.

CheckSquadsTimer (**Timestamp**) Unknown value.

PathfindInfo (**PathfindNavInfo**) Unknown value.

BloodDecalsFadeIndex (**int**) Unknown value.

ActivityInventoryBuffer[1024] (**char**) Unknown value.

ZoomCancelTimestamp (**Timestamp**) Unknown value.

NonInventoryItemHandle (**unsigned int**) Unknown value.

LastVehicleDriven (**unsigned int**) Unknown value.

QuickTurnOrient (**Matrix**) Unknown value.

QuickTurnSpeed (**float**) Unknown value.

Metadata (**PlayerMetadata**) Unknown value.

RadiationTimestamp (**Timestamp**) Unknown value.

RadiationDamage (**float**) Unknown value.

RadiationFoley (**int**) Unknown value.

IsStuckTimer (**Timestamp**) Unknown value.

LastStuckPos (**Vector**) Unknown value.

RagdollOverrideGetUpTime (**int**) Unknown value.

FadeBackpackTime (**TimestampPercent**) Unknown value.

CommTowerCheckPeriod (**Timestamp**) Unknown value.

NextRecord (**int**) Unknown value.

PositionalRecords[8] (**PlayerPositionalRecord**) Unknown value.

TrackingPeriod (**Timestamp**) Unknown value.

Salvage (**int**) The players salvage count. Shortcut for `Player.Metadata.Salvage`.

Functions

ResetMoveSpeed (*Player Self*) Resets the players move speed if it was overridden. Since the first argument is self, you call it with a colon instead of a period. So `Player:ResetMoveSpeed()`

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

PlayerFlags

Flags used to describe *Player* state and behavior.

Variables

ActionNodeAnimStarted (bool) Unknown value.

CustomMesh (bool) Unknown value.

IsLoaded (bool) Unknown value.

ScriptControlled (bool) Unknown value.

NeverDie (bool) Unknown value.

MissionNeverDie (bool) Unknown value.

DoSprint (bool) Unknown value.

DisableControls (bool) Unknown value.

JetpackStarted (bool) Unknown value.

JetpackReady (bool) Unknown value.

MovingToAStop (bool) Unknown value.

QuickTurn (bool) Unknown value.

CrouchingState (bool) Unknown value.

WeaponHidden (bool) Unknown value.

IsStuck (bool) Unknown value.

CoverFiring (bool) Unknown value.

CoverLeanup (bool) Unknown value.

UseCover (bool) Unknown value.

NoReticule (bool) Unknown value.

NoFineAim (bool) Unknown value.

CoverSearchForEdge (bool) Unknown value.

Coveragainstvehicle (bool) Unknown value.

InCommTowerRange (bool) Unknown value.

CommRangeChecked (bool) Unknown value.

PassengerBrake (bool) Unknown value.

LocalPlayerHitSomeoneThisFrame (bool) Unknown value.

WeaponSwapLock (bool) Unknown value.

UnlimitedAmmo (bool) Unknown value.

InRadiation (bool) Unknown value.

RagdollOnNextExplosion (bool) Unknown value.

LeavingSquadMemberBehind (bool) Unknown value.

IsWaitingForHostageVehicleExit (bool) Unknown value.

BackpackFadeOut (bool) Unknown value.

BackpackFadeIn (bool) Unknown value.

WaitingForLockerExit (bool) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

PlayerMetadata

Metadata about the [player's](#) activities in game such as play time and supply crates destroyed.

Variables

Salvage (int) The players current salvage amount. Useable to buy weapon upgrades at the upgrade table.

MiningCount (int) The amount of ore deposits the player has mined.

SupplyCrateCount (int) The amount of supply crates the player has destroyed.

DistrictHash (unsigned int) The hash for the current world district the player is in.

DistrictTime (int) The time in the players current district.

Upgrades[128] (UpgradeItem) An array of upgrades and their states. Each individual value may be `nil`, so be sure to check them before accessing them.

PlayTime (int) The players play time.

LastDeathTime (int) The players last death time.

PlayerScriptMode

Used in [Player](#) Defined in `./Core/rfg/PlayerScriptModes.lua`.

Access Variable	Value
<code>rfg.PlayerScriptModes.None</code>	0
<code>rfg.PlayerScriptModes.VehicleEnter</code>	1
<code>rfg.PlayerScriptModes.NumPlayerScriptModes</code>	2

PlayerZoomState

The `player's` gun/aim zoom state. Defined in `./Core/rfg/PlayerZoomState.lua`.

Access Variable	Value
<code>rfg.PlayerZoomState.Off</code>	0
<code>rfg.PlayerZoomState.Sprint</code>	1
<code>rfg.PlayerZoomState.FineAim</code>	2
<code>rfg.PlayerZoomState.Scope</code>	3

Primitive types

These are primitive types used by rfg. While Lua sees many of them as numbers, it's important to consider how the game sees them. For example, a `char` is an 8 bit type, meaning that it's values really only range from -127 to 127. While you can set it to higher values with a script, it won't make a difference trying to set it to a value beyond it's limits. In most circumstances this is unimportant, but you should keep it in the back of your mind.

Numbers

Note: You can set integers to decimal values, and vice-versa without issue. Just understand that rounding may occur as a result of this.

Integral types

These types can only be integers. That means that if you try setting one of these to a decimal value it will round towards zero.

Type	Range
unsigned int	0 to 4,294,967,295
int	-2147483648 to 2147483647
unsigned int16	0 to 65,535
int16	-32768 to 32767
unsigned char	0 to 255
char	-127 to 127

Floating point types

These types can be set to decimal values. The two floating point types are `float`, which is the most common with rfg, and `double`. The key difference between the two is that a `double` has greater precision and number range.

Type	Range	Precision
float	+/- 3.4E38	7 digits
double	+/- 1.7E308	15 digits

Bool

A bool (or boolean) type can either be `true` or `false`. You can use boolean values as expressions in if statements, like so:

```
a = true
if a then
    rsl.Log("a is true!", LogType.Info)
end
```

Bitfields

A bitfield is a type that allows the game to make variables smaller than 1 byte in size so that it may pack data in more tightly, and save on RAM usage. The important thing to know about this type is that it can widely vary in size and number range. As such, each bitfield will have it's number range listed. Bitfields are always an integral type so if you set a bitfield to a decimal value it will be rounded towards zero.

String

A string is an array of characters. Declaring an array should be done by surrounding a value with quotation marks, like so:

Note: `const char*` types can generally be treated as strings.

```
MyString = "Hello World!"
```

Strings can be concatenated using `..`

```
MyString = "Hello"
MyString = MyString .. " World!"
-- Now MyString == "Hello World!"
```

SalvageMaterialTypes

The material for an explosion to use when it creates salvage.

Access Variable	Value
<code>rfg.SalvageMaterialTypes.Metal</code>	0
<code>rfg.SalvageMaterialTypes.Ore</code>	1
<code>rfg.SalvageMaterialTypes.Chemical</code>	2
<code>rfg.SalvageMaterialTypes.NumSalvageMaterialTypes</code>	3
<code>rfg.SalvageMaterialTypes.Invalid</code>	4294967295

SaveLoadInfo

Flags used by the game to determine the right behavior during saving and loading. Seen in [World](#).

Variables

- PendingNewGame (Bitfield)** Equal 1 if a new game operation is pending. Range: 0 to 1
- ResetDestruction (Bitfield)** Equal to 1 if destruction should be reset on next load. Range: 0 to 1
- PendingSaveGame (Bitfield)** Equal to 1 if a save game operation is pending. Range: 0 to 1
- PerformingSaveGame (Bitfield)** Equal to 1 if a save game operation is occurring. Range: 0 to 1
- PendingSaveState (Bitfield)** Unknown value. Range: 0 to 1
- PerformingSaveState (Bitfield)** Unknown value. Range: 0 to 1
- PendingSingleZone (Bitfield)** Unknown value. Range: 0 to 1
- SaveGameWarp (Bitfield)** Unknown value. Range: 0 to 1
- GameSaveCheckpoint (Bitfield)** Unknown value. Range: 0 to 1
- GameSaveToDisk (Bitfield)** Equal to 1 if the game is being saved to the hard drive. Range: 0 to 1
- PendingLoadGameFromMemory (Bitfield)** Equal to 1 if an operation to load the game from memory (RAM) is pending. Range: 0 to 1
- AutoSaveGame (Bitfield)** Equal to 1 if the game is auto saving. Range: 0 to 1
- SavingStateData (Bitfield)** Unknown value. Range: 0 to 1
- PlayerStateAtSafehouse (Bitfield)** Unknown value. Range: 0 to 1
- PlayerStartPos (Vector)** The players start position for this save game. Range: 0 to 1
- PlayerStartOrient (Matrix)** The players starting orientation for this save game. Range: 0 to 1

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

ShadowStateBlock

Shadow state values. Access this through `rfg.Shadows`. So, for example, if you wanted to set the value of `ShadowPercent`, you'd do that like so: `rfg.Shadows.ShadowPercent = 0.5`.

Variables

- ShadowsEnabled (bool)** Needs description. Default value: `true`
- CloudShadowEnabled (bool)** Needs description. Default value: `true`
- CloudShadowScale (float)** Needs description. Default value: `0.5`
- CloudShadowIntensityScale (float)** Needs description. Default value: `4.5`
- CloudShadowIntensityBias (float)** Needs description. Default value: `-0.5`
- ShadowMapMaxDist (float)** Unknown purpose. Name is contradictory since increasing this value makes closeby shadows look worse. Default value: `100.0`
- ShadowMapFadePercent (float)** Needs description. Default value: `0.8`
- ShadowPercent (float)** Needs description. Default value: `1.0`
- DropShadowPercent (float)** Needs description. Default value: `0.4`

TerrainShadowMaxDist (float) Needs description. Default value: 240.0

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

SsaoStateBlock

SSAO state values. Access this through `rfg.Ssao`. So, for example, if you wanted to set the value of `Bias`, you'd do that like so: `rfg.Ssao.Bias = 0.65`.

Variables

Enabled (bool) Needs description. Default value: `true`

Bias (float) Needs description. Default value: 0.02

Clamp (float) Needs description. Default value: 0.83

Epsilon (float) Needs description. Default value: 0.4

Falloff (float) Needs description. Default value: 0.0

Intensity (float) Needs description. Default value: 4.0

Radius (float) Needs description. Default value: 1.2

DepthFadeRange (float) Needs description. Default value: 1.0

DepthThreshold (float) Needs description. Default value: 18.0

EdgeSharpness (float) Needs description. Default value: 10.0

ImagePlanePixelsPerMeterFactor (float) Needs description. Default value: 0.32

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

SunShaftsStateBlock

Sunshaft/godrays state values. Access this through `rfg.SunShafts`. So, for example, if you wanted to set the value of `Scale`, you'd do that like so: `rfg.SunShafts.Scale = 1.2`.

Variables

Enabled (bool) Needs description. Default value: `true`

SunMask (bool) Needs description. Default value: `true`

UseHalfResSource (bool) Needs description. Default value: `true`

SunSize (float) Needs description. Default value: 300.0

Scale (float) Needs description. Default value: 1.0

BaseLum (float) Needs description. Default value: 1.0

LumStepScale (float) Needs description. Default value: 1.0

Radius (float) Needs description. Default value: 5.0

ColorCurveBase (float) Needs description. Default value: -4.0

ColorCurveExp (float) Needs description. Default value: 1.60

ColorCurveShift (float) Needs description. Default value: -0.02

BlurMultiplier (float) Needs description. Default value: 1.0

BlurRho (float) Needs description. Default value: 0.75

CoronaAdaption (bool) Needs description. Default value: true

TaggingSequences

Usage unknown. Used in [Player](#). Defined in ./Core/rfg/TaggingSequences.lua.

Access Variable	Value
rfg.TaggingSequences.None	4294967295
rfg.TaggingSequences.Start	0
rfg.TaggingSequences.Tag	1
rfg.TaggingSequences.Done	2
rfg.TaggingSequences.NumTaggingSequences	3

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

TerrainStateBlock

Terrain state values. Access this through `rfg.Terrain`. So, for example, if you wanted to set the value of `FadeEnd`, you'd do that like so: `rfg.Terrain.FadeEnd = 5000`.

Variables

DetailMapTiling (float) Needs description. Default value: 1.3

DetailMapScale (float) Needs description. Default value: 4.8

DetailMapBias (float) Needs description. Default value: -0.035

DetailMapBlend (float) Needs description. Default value: 0.72

DetailMapEnable (bool) Needs description. Default value: true

SpecularEnable (bool) Needs description. Default value: true

RenderAlphaSkirts (bool) Needs description. Default value: true

AnisotropyLevel (int) Needs description. Default value: 4

FadeStart (float) Needs description. Default value: 140.0

FadeEnd (float) Increase this value to make terrain stay in high lod form at further distances. Default value: 250.0

MedLod (bool) Needs description. Default value: true

UseNewRenderer (bool) Needs description. Default value: true

Timestamp

Used by the game to measure time intervals. Not incredibly useful just yet as none of it's helper functions have been bound to lua.

Variables

Value (int) The value of the timestamp.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

TimestampPercent

Percentage based extension of [Timestamp](#). It's helper functions haven't been bound yet making it less useful.

Inherits [Timestamp](#)

Variables

SetMilliseconds (int) The timestamp value in milliseconds.

TriggerTypes

A weapons trigger type, used in [WeaponInfo](#) instances. These are defined in `./Core/rfg/TriggerTypes.lua`.

Access Variable	Value
<code>rfg.TriggerTypes.Single</code>	0
<code>rfg.TriggerTypesAutomatic</code>	1
<code>rfg.TriggerTypes.Charge</code>	2

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

UpgradeItem

An entry for an upgrade which can be accessed at an upgrade table at any safehouse.

Variables

CurrentLevel (char) The upgrades current level.

AvailabilityBitfield (unsigned __int16) Whether or not the upgrade is available.

UnlockedNotifiedBitfield (unsigned __int16) Unknown value.

NewNotifiedBitfield (unsigned __int16) Unknown value.

```
char current_level; //1 unsigned __int16 availability_bitfield; //2 unsigned __int16 un-
locked_notified_bitfield; //2 unsigned __int16 new_notified_bitfield; //2
```

UseObjectIcon

Icon of a useable object. Defined in `./Core/rfg/UseObjectIcon.lua`. The enum needs to be redefined on this page.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

UseableObject

Type description goes here.

Variables

Handle (**unsigned int**) Unknown value.

ActionType (**GeneralActionTypes**) Unknown value.

ActionIcon (**UseObjectIcon**) Unknown value.

DotProd (**float**) Unknown value.

DistSquared (**float**) Unknown value.

Vector

Used to represent several different values in the 3D world. For example, a 3D coordinate, direction, velocity, or a surface normal direction. Supports the following operators: `+`, `-`, `*`, `==`.

- *Variables*
- *Functions*

Variables

x (**float**) Coordinate or magnitude on the x axis. Positive x points east on the world map and negative x points west.

y (**float**) Coordinate or magnitude on the y axis. Positive y points directly up in the game world while negative y points directly down.

z (**float**) Coordinate or magnitude on the z axis. Positive z points north on the world map and negative z points south.

Functions

new (**Vector CopyVector**) Creates a new vector and sets it's values to the values of `CopyVector`.

new (**float InitialValue**) Creates a new vector and sets it's values to `InitialValue`.

new (**float x, float y, float z**) Creates a new vector and sets it's values directly with x, y, and z.

Cross (**Vector B**) Returns the cross product (a vector) of the calling vector and vector B.

Magnitude () Returns the magnitude of the vector as a `float`

Scale (**float Multiplier**) Returns a new vector made up of the components of the this vector times `Multiplier`.
You can use `1 / x` as a multiplier to divide all the components by that number. Be sure to check that `x ~= 0`.

SetAll (float Value) Sets all the vectors components to Value

UnitVector () Returns the **unit vector** of this vector. Which is a vector with the same direction of this vector, but a magnitude of 1.

GetDataString (Vector Self, bool Parentheses = false, bool Labels = true) Returns a **string** with the vector coordinates listed. If no arguments are provided it has false, then true as it's default args. Example usage:

```
a = rfg.Vector:new(1434.0, 15.0, 693.3)
DataString = a.GetDataString()
-- DataString == "x: 1434.0, y: 15.0, z: 693.3"
```

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

Vehicle

Vehicle object. Flyers, automobiles, and walkers are all vehicle objects. They each have their own classes which inherit **Vehicle**. Only **Flyer** has been binded so far.

Inherits **object**

Variables

SpawnPriority (ObjectSpawnPriorities) Unknown value.

Info (VehicleInfo) Unknown value.

LastPos (Vector) Unknown value.

LastOrient (Matrix) Unknown value.

LastVelocity (Vector) Unknown value.

LastAngularVelocity (Vector) Unknown value.

ForwardVelocity (float) Unknown value.

LastForwardVelocity (float) Unknown value.

AimHandle (int) Unknown value.

LastAnimTransform (Matrix43) Unknown value.

PfFailureTimeout (Timestamp) Unknown value.

DisableForNpcDriversTimer (Timestamp) Unknown value.

NoNpcEntryTimer (Timestamp) Unknown value.

RenderDistance (ObjectRenderDistance) Unknown value.

SeatInfo ('unsigned int[11]') Unknown value.

NumTurretMounts (int) Unknown value.

BombStatus (VehicleBombStatuses) Unknown value.

WalkerVelocityHack (Timestamp) Unknown value.

WalkerVelocityHackFollowup (Timestamp) Unknown value.

WalkerThrownEnergyScaled (Timestamp) Unknown value.

SavedAngularDampening (float) Unknown value.
WalkerVehicleCollisions (int) Unknown value.
VehicleVsBuildingDamage (Timestamp) Unknown value.
NumDamageEvents (int) Unknown value.
CorpseTimer (Timestamp) Unknown value.
CorpseAbsoluteLongestTimer (Timestamp) Unknown value.
OnFireTimestamp (Timestamp) Unknown value.
CorpseEffect (unsigned int) Unknown value.
ExhaustEffectNormal (unsigned int[4]) Unknown value.
ExhaustEffectBurst (unsigned int[4]) Unknown value.
Flags (VehicleFlags) Unknown value.
TurretAutofireMs (int) Unknown value.
NumSubPieces (unsigned int) Unknown value.
StreamPlacementFlags (VehicleSpawnFlags) Unknown value.
KillerHandle (unsigned int) Unknown value.
MostRecentDriver (unsigned int) Unknown value.
TeamOfMostRecentDriver (HumanTeams) Unknown value.
MostRecentDriverExitTime (Timestamp) Unknown value.
DamageSoundPropagateTimer (Timestamp) Unknown value.
KillerWeapon (WeaponInfo) Unknown value.
RammingDamageTaken (int) Unknown value.
ElectricalDamagePercent (float) Unknown value.
DamagePercent (float) Unknown value.
DamageFuncHandle (unsigned int16) Unknown value.
DestroyFuncHandle (unsigned int16) Unknown value.
OnEnterFuncHandle (unsigned int16) Unknown value.
OnExitFuncHandle (unsigned int16) Unknown value.
OnTakeDamageHandle (unsigned int16) Unknown value.
OnCollisionHandle (unsigned int16) Unknown value.
OnHitPedHandle (unsigned int16) Unknown value.
ReservedBy (unsigned int) Unknown value.
SoundDelayAfterExplosion (Timestamp) Unknown value.
FireDamageRate (float) Unknown value.
FireFractionalDamage (float) Unknown value.
FadeTimer (Timestamp) Unknown value.
FadeTime (int) Unknown value.

EmergencyLightTimer (Timestamp) Unknown value.

StreamLoadDistanceSqr (float) Unknown value.

StreamUnloadDistanceSqr (float) Unknown value.

EngineInst (int) Unknown value.

EngineStartedInst (int) Unknown value.

EngineHighLoadStartTimer (Timestamp) Unknown value.

PassByPlayId (int) Unknown value.

PassByDistance (float) Unknown value.

NavCellDetourRequestHandle (unsigned int) Unknown value.

NavCellDetourComponentRequestHandles (unsigned int[8]) Unknown value.

NumNavCellDetourComponentRequestHandles (unsigned int) Unknown value.

LastDamageReported (float) Unknown value.

SpawnNodeHandle (unsigned int) Unknown value.

SquadHandle (unsigned int) Unknown value.

VehicleCheckCoverTimestamp (Timestamp) Unknown value.

VehicleCoverTimestamp (Timestamp) Unknown value.

VehicleCoverCreationPos (Vector) Unknown value.

VehicleCheckCoverIndex (int) Unknown value.

VehicleCoverIndex (int) Unknown value.

InfiniteMass (bool) Unknown value.

ExtraMass (bool) Unknown value.

ExtraMassValue (float) Unknown value.

VehicleBombStatuses

Base vehicle types. Defined in `Core/rfg/VehicleBombStatuses.lua`. Access through `rfg.VehicleBombStatuses`.

Access Variable	Value
<code>rfg.VehicleBombStatuses.None</code>	0
<code>rfg.VehicleBombStatuses.Armed</code>	1

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleCameraSettings

The camera settings for a vehicle prototype. Found in `VehicleInfo` values.

Variables

LookatOffset (**Vector**) Unknown value.

LookatVehicleOffset (**Vector**) Unknown value.

LookatVehicleEnterOffset (**Vector**) Unknown value.

LookfromHeight (**float**) Unknown value.

EnterDistance (**float**) Unknown value.

FollowDist (**float**) Unknown value.

FollowHeight (**float**) Unknown value.

YAxisRotationSpeed (**float**) Unknown value.

CameraSteerAngle (**float**) Unknown value.

CameraSteerSpeed (**float**) Unknown value.

TurretCamera (**bool**) Unknown value.

ForceTurretCam (**bool**) Unknown value.

VehicleClassTypes

Vehicle class types. Defined in `Core/rfg/VehicleClassTypes.lua`. Access through `rfg.VehicleClassTypes`.

Access Variable	Value
<code>rfg.VehicleClassTypes.Automobile</code>	0
<code>rfg.VehicleClassTypes.Flyer</code>	1
<code>rfg.VehicleClassTypes.Walker</code>	2

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleFlags

Flags used to determine the behavior for **Vehicles**.

Variables

Activity (**bool**) Unknown value.

OldActivity (**bool**) Unknown value.

ActivityDestroyed (**bool**) Unknown value.

ConvoyVehicle (**bool**) Unknown value.

AmbientSpawn (**bool**) Unknown value.

DeathReported (**bool**) Unknown value.

NoRagdoll (**bool**) Unknown value.

DespawnedSquadVehicle (bool) Unknown value.
DeleteMissionVehicle (bool) Unknown value.
DoNotDestroyWithSquad (bool) Unknown value.
Corpse (bool) Unknown value.
DespawnAfterFade (bool) Unknown value.
DoorCollisionEnabled (bool) Unknown value.
EngineRunning (bool) Unknown value.
EngineSmoking (bool) Unknown value.
FadingIn (bool) Unknown value.
FadingOut (bool) Unknown value.
FadingFromNano (bool) Unknown value.
Hidden (bool) Unknown value.
Hijacked (bool) Unknown value.
ScriptedBail (bool) Unknown value.
ScriptedAbandon (bool) Unknown value.
HornDown (bool) Unknown value.
Invulnerable (bool) Unknown value.
MissionInvulnerable (bool) Unknown value.
IsPlayerCar (bool) Unknown value.
KillByVehicle (bool) Unknown value.
MpHonkedHorn (bool) Unknown value.
NeedsRepair (bool) Unknown value.
Occupied (bool) Unknown value.
OnFire (bool) Unknown value.
NanoSimFire (bool) Unknown value.
NanoSimSmoke (bool) Unknown value.
GrenadeInside (bool) Unknown value.
ReleasedFromCutscene (bool) Unknown value.
ResourceAccessDisabled (bool) Unknown value.
ReverseLastFrame (bool) Unknown value.
SeriesOfHonks (bool) Unknown value.
SirenDying (bool) Unknown value.
SirenAudioOn (bool) Unknown value.
UsingHorn (bool) Unknown value.
PassByGettingCloser (bool) Unknown value.
WheelsOnGround (bool) Unknown value.

AllWheelsOnGround (bool) Unknown value.

OnlyFireDamage (bool) Unknown value.

HeadlightsOn (bool) Unknown value.

EmergencyLightsOn (bool) Unknown value.

HighPriorityTarget (bool) Unknown value.

PlayerMayCapture (bool) Unknown value.

DisableForNpcDrivers (bool) Unknown value.

DisableForPlayer (bool) Unknown value.

DisableAccelerator (bool) Unknown value.

LockedInPlace (bool) Unknown value.

FixedMotion (bool) Unknown value.

DisableTurretsForNpcs (bool) Unknown value.

ImmediateSpawn (bool) Unknown value.

NoPlayerUse (bool) Unknown value.

InvisibleDriver (bool) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleInfo

Defines the values used in vehicle prototype. This means that if you edit a `VehicleInfo`, you're editing all vehicles of that type. You can access all the `VehicleInfo`'s by looping through `rfg.VehicleInfos`.

Variables

Name (string) Unknown value.

DisplayName (string) Unknown value.

MeshName (string) Unknown value.

VehicleClass (VehicleClassTypes) Unknown value.

SpawnSize (float) Unknown value.

SpawnBox (Bbox) Unknown value.

SlotId (int) Unknown value.

Srid (unsigned int) Unknown value.

VariantFamily (string) Unknown value.

VehicleClassification (BaseVehicleTypes) Unknown value.

DefaultTeam (HumanTeams) Unknown value.

MaxHitpoints (unsigned int) Unknown value.

AbandonHitpoints (int) Unknown value.

ChassisMass (float) Unknown value.

Value (float) Unknown value.

ChopShopProps (unsigned int) Unknown value.

NumLods (unsigned int) Unknown value.

LodInfo (LodInfo[4]) Unknown value.

NumTurretMounts (int) Unknown value.

RoadPreference (VehicleRoadPreferences) Unknown value.

EngineTorque (float) Unknown value.

AiEngineTorque (float) Unknown value.

MinRpm (float) Unknown value.

OptimalRpm (float) Unknown value.

MaxRpm (float) Unknown value.

MinRpmTorqueFactor (float) Unknown value.

MaxRpmTorqueFactor (float) Unknown value.

MinRpmResistance (float) Unknown value.

OptRpmResistance (float) Unknown value.

MaxRpmResistance (float) Unknown value.

ReverseTorqueMultiplier (float) Unknown value.

ClutchSlipRpm (float) Unknown value.

EnlargedWheelRadius (float) Unknown value.

AutobrakeSpeed (float) Unknown value.

EnforcedMaxSpeed (float) Unknown value.

AiEnforcedMaxSpeed (float) Unknown value.

EnforcedMaxRspeed (float) Unknown value.

PowerslideFriction (float) Unknown value.

PowerslidePower (float) Unknown value.

ForceAssist (float) Unknown value.

NumAxles (unsigned int) Unknown value.

AxleWheelInfos (VehicleInfoAxleWheelInfo[4]) Unknown value.

ExhaustTagId (int[4]) Unknown value.

TransInfo (VehicleInfoTransmissionInfo) Unknown value.

MaxSteeringAngle (float) Unknown value.

MaxSpeedSteeringAngle (float) Unknown value.

AiMaxSpeedSteeringAngle (float) Unknown value.

SteeringWheelMaxSpeed (float) Unknown value.

SteeringWheelMaxReturnSpeed (float) Unknown value.

SteeringWheelDampAngle (float) Unknown value.
SteeringWheelReturnDampAngle (float) Unknown value.
CounterSteerMinSpeed (float) Unknown value.
CounterSteerMaxSpeed (float) Unknown value.
AirSteerRollMaxVel (float) Unknown value.
AirSteerRollMaxAngleRad (float) Unknown value.
AirSteerPitchMaxVel (float) Unknown value.
AirSteerPitchMaxAngleRad (float) Unknown value.
MinPedalInputToBlock (float) Unknown value.
MinTimeToBlock (float) Unknown value.
AiMinTimeToBlock (float) Unknown value.
AirDensity (float) Unknown value.
FrontalArea (float) Unknown value.
DragCoefficient (float) Unknown value.
LiftCoefficient (float) Unknown value.
ExtraGravity (float) Unknown value.
CenterOfMassY (float) Unknown value.
CenterOfMassZ (float) Unknown value.
CameraSettings (VehicleCameraSettings[3]) Unknown value.
CameraSettingsCount (int) Unknown value.
EnergyScale (float) Unknown value.
CollisionMassScalar (float) Unknown value.
CollisionDamageScale (float) Unknown value.
TerrainDamageScale (float) Unknown value.
BulletDamageScale (float) Unknown value.
VehicleDamageScale (float) Unknown value.
FrictionEqualizer (float) Unknown value.
AiFrictionEqualizer (float) Unknown value.
TorqueRollFactor (float) Unknown value.
TorquePitchFactor (float) Unknown value.
TorqueYawFactor (float) Unknown value.
TorqueYawScalar (float) Unknown value.
ExtraTorqueFactor (float) Unknown value.
ChassisUnitInertiaRoll (float) Unknown value.
ChassisUnitInertiaPitch (float) Unknown value.
ChassisUnitInertiaYaw (float) Unknown value.

AiChassisUnitInertiaYaw (float) Unknown value.

ViscosityFriction (float) Unknown value.

AiMaxBrakingDecel (float) Unknown value.

AiMaxRadialAccel (float) Unknown value.

AlertMultiplier (float) Unknown value.

FoleyStart (int) Unknown value.

EngineId (int) Unknown value.

EngineWavebankId (int16) Unknown value.

FoleyOff (int) Unknown value.

FoleyEnginePeel (int) Unknown value.

FoleyShift (int) Unknown value.

FoleyGrind (int) Unknown value.

FoleyHonk (int) Unknown value.

FoleyImpactId (int[2]) Unknown value.

FoleyScrapingId (int) Unknown value.

FoleyCorpseImpactId (int) Unknown value.

FoleyComponentImpactId (int) Unknown value.

FoleyWheelImpactId (int) Unknown value.

FoleyChassisLandIdx (int[2]) Unknown value.

FoleyPassBy (int) Unknown value.

FoleyDoorOpenId (int) Unknown value.

FoleyDoorCloseId (int) Unknown value.

EffectEngineFire (unsigned int) Unknown value.

EffectEngineSmoke (unsigned int) Unknown value.

EffectCorpseSmoke (unsigned int) Unknown value.

EffectComponentFire (unsigned int) Unknown value.

EffectExhaustNormal (unsigned int) Unknown value.

EffectExhaustBurst (unsigned int) Unknown value.

EffectCollision (unsigned int) Unknown value.

EffectScrape (unsigned int) Unknown value.

EffectComponentDetach (unsigned int) Unknown value.

EffectHeadLightPrimary (unsigned int) Unknown value.

EffectHeadLightSecondary (unsigned int) Unknown value.

EffectFogLight (unsigned int) Unknown value.

EffectTailLight (unsigned int) Unknown value.

EffectBrakeLight (unsigned int) Unknown value.

EffectReverseLight (**unsigned int**) Unknown value.
EffectEmergencyLight (**unsigned int**) Unknown value.
EffectStrobeLight (**unsigned int**) Unknown value.
ExplosionInfo (**ExplosionInfo**) Unknown value.
BombExplosionInfo (**ExplosionInfo**) Unknown value.
NumVariants (**unsigned int**) Unknown value.
NormalSpinDamping (**float**) Unknown value.
NormalSpinDampingAi (**float**) Unknown value.
CollisionSpinDamping (**float**) Unknown value.
CollisionSpinThreshold (**float**) Unknown value.
CameraFovMultiplier (**float**) Unknown value.
CameraFovMinSpeed (**float**) Unknown value.
CameraShakeMinSpeed (**float**) Unknown value.
RadialBlurMax (**float**) Unknown value.
RadialBlurMinSpeed (**float**) Unknown value.
TrailerChance (**float**) Unknown value.
RigName (**string**) Unknown value.
AnimSetName (**string**) Unknown value.
FootstepEffects (**FootGroundEffects**) Unknown value.
FlyerMaxUpThrust (**float**) Unknown value.
FlyerMaxTurnAngvel (**float**) Unknown value.
FlyerMaxTurnAngaccl (**float**) Unknown value.
FlyerMaxThrustOffsetX (**float**) Unknown value.
FlyerMaxThrustOffsetZ (**float**) Unknown value.
FlyerMaxTiltAngVel (**float**) Unknown value.
FlyerMaxTiltAngAccel (**float**) Unknown value.
FlyerSpinBankScalar (**float**) Unknown value.
FlyerThrustTiltScalar (**float**) Unknown value.
FlyerBankTiltScalar (**float**) Unknown value.
FlyerDefLookatYScalar (**float**) Unknown value.
FlyerWingtipEffect (**unsigned int**) Unknown value.
FlyerThrusterEffect (**unsigned int**) Unknown value.
FlyerMainEngineEffect (**unsigned int**) Unknown value.
FlyerJetwashEffect (**unsigned int**) Unknown value.
Flags (**VehicleInfoFlags**) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleInfoAxleWheelInfo

Contains values which determine the behavior of a vehicle prototype's wheels and axles. Found in [VehicleInfo](#) values.

Variables

DoesSteer (bool) Unknown value.

DoesHandbrake (bool) Unknown value.

EngineTorqueFactor (float) Unknown value.

Mass (float) Unknown value.

Friction (float) Unknown value.

AiFriction (float) Unknown value.

BrakingTorque (float) Unknown value.

SpringCompression (float) Unknown value.

SpringExtension (float) Unknown value.

SpringStrength (float) Unknown value.

SpringPower (float) Unknown value.

CompressionDamping (float) Unknown value.

ExpansionDamping (float) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleInfoFlags

Flags used in [VehicleInfo](#) instances.

Variables

Preload (int8) Unknown value.

CanBlockRoads (int8) Unknown value.

ForceBreakLinks (int8) Unknown value.

SeatCountWarning (int8) Unknown value.

IsWalkerFlamer (int8) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleInfoTransmissionInfo

Contains values which determine the behavior of a vehicle prototype's transmission. Found in [VehicleInfo](#) values.

Variables

NumGears (**unsigned int**) Unknown value.

GearRatios (**float[6]**) Unknown value.

DownshiftRpms (**float[5]**) Unknown value.

UpshiftRpms (**float[5]**) Unknown value.

DifferentialGearRatio (**float**) Unknown value.

ReverseGearRatio (**float**) Unknown value.

ClutchDelay (**float**) Unknown value.

ClutchDelayPreShift (**float**) Unknown value.

ForwardToReverseDelay (**unsigned int**) Unknown value.

ReverseToForwardDelay (**unsigned int**) Unknown value.

VehicleRoadPreferences

Vehicle class types. Defined in `Core/rfg/VehicleRoadPreferences.lua`. Access through `rfg.VehicleRoadPreferences`.

Access Variable	Value
<code>rfg.VehicleRoadPreferences.None</code>	0
<code>rfg.VehicleRoadPreferences.Highway</code>	1
<code>rfg.VehicleRoadPreferences.NoHighway</code>	2

VehicleSeatIndex

Represents the seat that a [Human](#) occupies in a vehicle. Defined in `./Core/rfg/VehicleSeatIndex.lua`.

Access Variable	Value
<code>rfg.VehicleSeatIndex.Best</code>	4294967294
<code>rfg.VehicleSeatIndex.Invalid</code>	4294967295
<code>rfg.VehicleSeatIndex.Driver</code>	0
<code>rfg.VehicleSeatIndex.Passenger1</code>	1
<code>rfg.VehicleSeatIndex.Passenger2</code>	2
<code>rfg.VehicleSeatIndex.Passenger3</code>	3
<code>rfg.VehicleSeatIndex.Passenger4</code>	4
<code>rfg.VehicleSeatIndex.Passenger5</code>	5
<code>rfg.VehicleSeatIndex.Passenger6</code>	6
<code>rfg.VehicleSeatIndex.Passenger7</code>	7
<code>rfg.VehicleSeatIndex.Passenger8</code>	8
<code>rfg.VehicleSeatIndex.Passenger9</code>	9
<code>rfg.VehicleSeatIndex.Passenger10</code>	10
<code>rfg.VehicleSeatIndex.NumVehicleSeats</code>	11

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

VehicleSpawnFlags

Flags used for vehicle spawning. Used in [Vehicle](#).

Variables

HighPriority (bool) Unknown value.

MobileSpawn (bool) Unknown value.

StaticPlacement (bool) Unknown value.

SnapToGround (bool) Unknown value.

RoadSpawn (bool) Unknown value.

VoiceLineHandle

Purpose unknown. Defined in `./Core/rfg/VoiceLineHandle.lua`.

Access Variable	Value
<code>rfg.VoiceLineHandle.InvalidVoiceLineHandle</code>	4294967295
<code>rfg.VoiceLineHandle.HandleIsDLC</code>	32768
<code>rfg.VoiceLineHandle.HandleForceTo32Bit</code>	2147483647

VoiceLinePriorities

Priority of a voice line. Who would've guessed. Defined in `./Core/rfg/VoiceLinePriorities.lua`.

Access Variable	Value
<code>rfg.VoiceLinePriorities.None</code>	0
<code>rfg.VoiceLinePriorities.VeryLow</code>	1
<code>rfg.VoiceLinePriorities.Low</code>	2
<code>rfg.VoiceLinePriorities.Normal</code>	3
<code>rfg.VoiceLinePriorities.High</code>	4
<code>rfg.VoiceLinePriorities.VeryHigh</code>	5
<code>rfg.VoiceLinePriorities.Critical</code>	6
<code>rfg.VoiceLinePriorities.ForceAlways</code>	7
<code>rfg.VoiceLinePriorities.NumVoiceLinePriorities</code>	8
<code>rfg.VoiceLinePriorities.MinInterruptLevel</code>	9

VoiceLines

Enum for various preset voice lines. Defined in `./Core/rfg/VoiceLines.lua`. This is another fairly large enum, with several dozen entries so for now it hasn't been reproduced here to save on time. Please see the mentioned lua file for a list of it's values.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

WeaponAnimationFlags

A set of flags used to determine weapon animation behavior for a human instance.

Note: The names of these variables are the same names the game uses for them and they may not all have an obvious or immediate effect. You may also find that one of the descriptions here is inaccurate. Feel free to update the page with more accurate info.

Variables

Prone (Bitfield) Equals 1 if the human is prone. Range: 0 to 1

Crouch (Bitfield) Equals 1 if the human is crouched. Range: 0 to 1

DownReady (Bitfield) Range: 0 to 1

Ready (Bitfield) Range: 0 to 1

Aim (Bitfield) Range: 0 to 1

FineAim (Bitfield) Range: 0 to 1

NoAmmo (Bitfield) Range: 0 to 1

Walk (Bitfield) Range: 0 to 1

Run (Bitfield) Range: 0 to 1

Left (Bitfield) Range: 0 to 1

Right (Bitfield) Range: 0 to 1

Back (Bitfield) Range: 0 to 1

Backpedal (Bitfield) Range: 0 to 1

Switching (Bitfield) Range: 0 to 1

Drive (Bitfield) Range: 0 to 1

Ride (Bitfield) Range: 0 to 1

RideLeft (Bitfield) Range: 0 to 1

Visited (Bitfield) Range: 0 to 1

SpecialLifetime (Bitfield) Range: 0 to 1

SerializeProtected (Bitfield) Range: 0 to 1

DontUseMe (Bitfield) Range: 0 to 1

StreamingFixed (Bitfield) Range: 0 to 1

RenderFlags (Bitfield) Range: 0 to 1

WeaponClasses

A weapons class, used in [WeaponInfo](#) instances. These are defined in `./Core/rfg/WeaponClasses.lua`.

Access Variable	Value
<code>rfg.WeaponClasses.None</code>	4294967295
<code>rfg.WeaponClasses.Bullet</code>	0
<code>rfg.WeaponClasses.Launcher</code>	1
<code>rfg.WeaponClasses.Thrown</code>	2
<code>rfg.WeaponClasses.ThrownCharge</code>	3
<code>rfg.WeaponClasses.ThrownMine</code>	4
<code>rfg.WeaponClasses.ThrownGrenade</code>	5
<code>rfg.WeaponClasses.Melee</code>	6
<code>rfg.WeaponClasses.Sledgehammer</code>	7
<code>rfg.WeaponClasses.Gutter</code>	8
<code>rfg.WeaponClasses.Grinder</code>	9
<code>rfg.WeaponClasses.ArcWelder</code>	10
<code>rfg.WeaponClasses.NanoRifle</code>	11
<code>rfg.WeaponClasses.MassDriver</code>	12
<code>rfg.WeaponClasses.Harpoon</code>	13
<code>rfg.WeaponClasses.SingularityBomb</code>	14
<code>rfg.WeaponClasses.Repair</code>	15
<code>rfg.WeaponClasses.Flame</code>	16

WeaponInfo

Values used to determine a weapon types behavior. To access this type you should use the [WeaponInfo](#) list `rfg.WeaponInfos`. See the example scripts `WeaponInfoTest1.lua` and `WeaponInfoTest2.lua` in `RSL/Scripts/Included scripts/` for examples of how this can be done (included in each release). Note that when you edit a [WeaponInfo](#) instance, that you're editing the behavior of all weapons based off of that, unless you make your own copy of it.

Variables

Name (String) The instances name.

NameCrc (ChecksumStri) The CRC checksum for the name.

UniqueId (int) The unique id of this instance.

Flags (WeaponInfoFlags) Flags which determine additional behavior of the weapon.

WeaponClass (WeaponClasses) The weapon class.

WeaponInvItemInfo (InvItemInfo) The weapons inventory info.

DefaultTeam (HumanTeams) The default team for this instance.

IconName (String) The name of the icon used. May be `nil`.

SmallIconName (String) The name of the small icon used. May be `nil`.

ReticuleName (String) The name of the reticule used. May be `nil`.

FineAimReticuleName (String) The name of the fine aim reticule used. May be `nil`.

WeaponAnimGroup (AnimationGroups) The animation group for this weapon.

MuzzleFlashEffect (unsigned int) The id for the muzzle flash effect.

MuzzleSmokeEffect (unsigned int) The id for the muzzle smoke effect.

SpecialHitEffect (unsigned int) Unknown value.

SpecialEffect (unsigned int) Unknown value.

SecondarySpecialEffect (unsigned int) Unknown value.

OverheatedEffect (unsigned int) The id for the overheating effect.

TracerEffect (unsigned int) The id for the tracer effect.

FireCameraShakeIgnoreDisabled (bool) Unknown value.

AttachmentPoint (String) Unknown value.

FireSound (int) The id for the fire sound.

SecondarySound (int) The id for the secondary fire sound.

UpgradeSound (int) The id for the upgrade sound.

ReloadSound (int) The id for the reload sound.

ReloadSoundDelay (int) The delay for the reload sound in unknown units.

NoAmmoSound (int) The id for the no ammo sound.

SpecialSound (int) Unknown value.

FlybySound (int) The id for the bullet fly by sound.

NumWeaponPersonas (int) Unknown value.

NpcFireSounds[16] (int) Unknown value.

MaxRange (float) The max range in meters for the weapon.

RedRange (float) Unknown value.

MaxEngagementDist (float) Unknown value.

MinEngagementDist (float) Unknown value.

MaxAiPenetratingDist (float) Unknown value.

TriggerType (TriggerTypes) The trigger type (automatic/single).

AmmoType (AmmoTypes) The ammo type (electric, bullet, projectile, etc).

MagazineSize (unsigned int16) Magazine size.

MagazineStartNum (unsigned int16) The starting number of rounds in a magazine.

MaxRounds (unsigned int16) The max rounds carried.

MaxRoundsUpgrade (unsigned int16) Unknown value.

AmmoBoxRestock (unsigned int16) The amount of rounds to be restocked by an ammo box.

ToMinSpread (unsigned int16) Unknown value.

ToMaxSpread (unsigned int16) Unknown value.

MeleeGroupIndex (int8) Unknown value.

BulletGroupIndex (int8) Unknown value.

TracerFrequency (int8) Unknown value.

ShotsPerRound (int8) Unknown value.

FiringSoundRadius (float) The radius in meters that the weapon can be heard firing from.

NpcRefireDelay (float) The delay between shots for NPCs in seconds.

DefaultRefireDelay (float) The delay between shots for the player in seconds.

PrefireDelay (float) Unknown value.

DefaultReloadDelay (int) Unknown value.

LowScaleDamage (DamageScalingInfo) Unknown value.

HighScaleDamage (DamageScalingInfo) Unknown value.

ExplosionInfo (ExplosionInfo) The explosion that should be created when this weapons rounds impact. Will be `nil` if there's no explosion for this weapon.

AiExplosionInfo (ExplosionInfo) An optional separate explosion that is used for AI. May be `nil`.

FireConeDot (float) The angle of the fire cone. Range should be from `0.0` to `1.0`. At `1.0` the fire cone is perfectly accurate, firing exactly where you point, at `0.0` it's a 180 degree cone.

EvenSpreadAccuracyDot (float) Unknown value.

MaxSpread (float) The maximum spread of the weapons rounds. Units if any are unknown.

MinSpread (float) The minimum spread of the weapons rounds. Units if any are unknown.

FineAimMaxSpread (float) The max spread when in fine aim mode.

FineAimMinSpread (float) The min spread when in fine aim mode.

NpcMaxSpread (float) The max spread for NPCs.

NpcMinSpread (float) The min spread for NPCs.

SpreadMultiplierRun (float) The number spread should be multiplied by if you're running. If this is greater than `1.0` then your spread will increase while running.

RagdollForceShoot (float) The amount of force to apply to a ragdoll when shot by this weapon. Units unknown, but likely in newtons.

RagdollChance (float) The chance that getting hit by this weapon will cause the target to ragdoll.

RecoilCameraKick (float) How much the camera should react to recoil.

RecoilImpulse (float) Unknown value. Likely the impulse that should be applied to the player when this is fired.

OutOfAmmoReloadDelay (int) Unknown value.

OverheatCoolDownTime (float) Unknown value.

OverheatPercentPerShot (float) Unknown value.

DroppedAmmoScale (float) Unknown value.

BulletHoleScale (float) Unknown value.

HeadshotMultiplier (float) A number that damage is multiplied by when doing headshot damage.

ZoomMagnification (float) How much the view should be magnified when using fine aim.

AutoaimOverride (float) Unknown value.

NpcAutoaim (float) Unknown value.

AimAssist (float) Unknown value.

PlayerMoveSpeedMultiplier (float) Unknown value.

NpcMoveSpeedMultiplier (float) Unknown value.

AlertMultiplier (float) Unknown value.

ProjectileInfo (WeaponProjectileInfo) Information about the weapons projectiles if it uses projectile ammo.

StandingPrimaryMeleeAttack (int) Unknown value.

StandingSecondaryMeleeAttack (int) Unknown value.

StandingTertiaryMeleeAttack (int) Unknown value.

CrouchingPrimaryMeleeAttack (int) Unknown value.

CrouchingSecondaryMeleeAttack (int) Unknown value.

CrouchingTertiaryMeleeAttack (int) Unknown value.

WeaponInfoFlags

Flags which control the behavior of a weapon.

Variables

ArmorPiercing (bool) If true, the weapon is better at piercing armor.

CanFineAim (bool) If true, fine aim (default keybind is 'f') is possible with this weapon.

CanScope (bool) Unknown value.

Shatter (bool) Unknown value.

HeldInLeftHand (bool) If true, held in the left hand.

DrawProjectileInOppositeHand (bool) Exact behavior unknown, likely used for weapons like remote charges, where the detonator is in one hand and the thrown projectile is in the other.

SilentBullets (bool) Whether or not the bullets should make a sound when passing by and on impact.

PenetratingBullets (bool) If true, the bullets can penetrate structures. If true, NPCs also get "Xray vision" equivalent to what they get with the vanilla rail driver.

NonInventory (bool) Unknown value.

UseEvenSpread (bool) Unknown value.

DisablePlayerCover (bool) Unknown value.

IsObviousWeapon (bool) If true, the EDF will aggress the player if they are seen holding this weapon.

AutoAimCurvedTrail (bool) Unknown value.

LoopingEffects (bool) Unknown value.

NeverInCabinet (bool) Unknown value.

UseSecondaryWeaponHeat (bool) Unknown value.

DontDropOnDeath (bool) Unknown value.

WeaponProjectileInfo

Properties of the projectiles used by a given weapon. Used in the [WeaponInfo](#) type.

Variables

StartSpeed (float) The initial speed of the projectile in m/s.

MaxSpeed (float) The max speed of the projectile in m/s.

Acceleration (float) The acceleration of the projectile in m/s^2.

FuseTime (unsigned int16) The time that must pass before the projectile can explode if explosion.

MaxThrowDist (float) The max distance the projectile can be thrown in meters.

Gravity (float) The strength of gravity for the projectile (separate from global gravity) in m/s^2.

Sound (int) Unknown value. Likely the handle for the sound the projectile uses.

Effect1 (unsigned int) The id of the first effect the projectile uses.

Effect2 (unsigned int) The id of the second effect the projectile uses.

Effect3 (unsigned int) The id of the third effect the projectile uses.

Effect4 (unsigned int) The id of the fourth effect the projectile uses.

Flags (unsigned int) Unknown value.

InaccurateFlight (float) Unknown value.

TimeUntilPropelled (float) The time in seconds before the projectile starts getting accelerated.

TimeUntilPropExpire (float) Unknown value. Possibly the time until the projectiles propellant expires.

TimeUntilDrop (float) The time in seconds before the projectile starts falling towards the ground.

DamageEffect (unsigned int) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

World

The rfg world structure. Contains the global list of all objects in the game and info about each of the world zones. You can use `rfg.ActiveWorld` or `World` to access the current world.

Variables

MissionObjectCreationMode (bool) Unknown value.

LevelAmbient (Vector) The ambient light color of the environment.

LevelBackAmbient (Vector) A secondary ambient light color of the environment.

LastLoadedTerritory (char[64]) Unknown value.

MaxWorldObjects (int) The maximum number of objects the game world can support.

AllObjects (BaseArray<Object*>) An array containing all of the objects in the game world. You can iterate through this and filter objects by type to interact with the game world. When iterating through the list be sure to check if each object is `nil` before accessing it to avoid errors.

Objects (BaseArray<Object*>) An alternative name for `AllObjects`. Shorter and more convenient.

TechLevel (float) The current tech level, used to determine the weapons used by NPCs.

TechLevelMax (float) The maximum tech level possible.

FlaggedObjects (Object*) Unknown value. Possibly an array of flagged objects that the game uses. May be `nil`.

CurrentFlaggedObject (Object*) Unknown value. May be `nil`.

CurrentFlaggedMode (char) Unknown value. Likely related to the previous two variables.

DeserializeList (char[144]) Unknown value.

CurWorldState (WorldStateModes) Unknown value.

SIFlags (SaveLoadInfo) Flags used by the game to determine the right behavior when loading/saving the game. Such as if it should use the reset destruction feature on next load.

PendingGameSaveSlot (GameSaveInfo*) Info about the game state which would be stored in a save game if you save. Things like the number of missions completed or hours played. May be `nil`.

DlcBundleId (int) Unknown value.

PendingFilename (char[64]) Unknown value.

PendingGameLoadWarpToPos (Vector) Unknown value.

PendingGameLoadWarpToOrient (Matrix) Unknown value.

StreamPos (Vector) Unknown value.

NumTerritoryZones (int) Unknown value.

GlobalZoneGrid (WorldZone*[257]) Unknown value. May be `nil`.

IsTerritory (bool) Unknown value.

TerritoryName[128] (char) Unknown value.

NumStreamingObjects (int) The number of objects currently being streamed in by the game.

StubSerializationInProgress (bool) Unknown value.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

WorldStateBuf

Buffer containing info about the players movements throughout a `WorldZone`.

Variables

PlayerStartPos (Vector) The players starting position in this zone.

PlayerStartOrient (Matrix) The players starting orientation in this zone.

PlayerStartPosSafehouse (Vector) The players starting safehouse position.

PlayerStartOrientSafehouse (Matrix) The player starting safehouse orient.

Buf (**char***) Unknown value. Likely an array that needs to be wrapped first to be used through lua.

CurSize (**int**) The current size in bytes of Buf.

MaxSize (**int**) The max size in bytes of Buf.

WorldStateModes

Used in [World](#). Exact purpose unknown.

Access Variable	Value
<code>rfg.WorldStateModes.Default</code>	0
<code>rfg.WorldStateModes.Checkpoint</code>	1
<code>rfg.WorldStateModes.NumWorldStateModes</code>	2

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

WorldZone

Represents a cube of the game world. The game world is split up into zones and loaded/unloaded as you move around in this. This generally isn't visible unless you maximize the far clip distance, disable fog, and move to a higher altitude where you'll see the individual zones as square region that load in as you move around.

Variables

Bmin (**Vector**) One “minimum” corner of the zones cube.

Bmax (**Vector**) The “maximum” corner of the zones cube.

Name (**char[64]**) The name of the zone.

State (**WorldZoneStates**) The state of the zone.

DeserializeHeader (**ZoneHeader***) The header data from the zones `rfgzone_pc` file. May be `nil`.

StoredZoneState (**WorldStateBuf**) Info about the players movements through this zone.

Srid (**unsigned int**) Unknown value.

IsBorderZone (**bool**) Set to `True` if the zone is a border zone. On the edge of the game world.

Gid (**unsigned int16**) Unknown value.

WorldZoneStates

Used in [WorldZone](#). Exact purpose unknown.

Access Variable	Value
<code>rfg.WorldZoneStates.Unloaded</code>	0
<code>rfg.WorldZoneStates.Streaming</code>	1
<code>rfg.WorldZoneStates.Loaded</code>	2

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

ZoneHeader

The header data found in the first 24 bytes of rfgzone_pc files. Each **WorldZone** stores this data from it's own rfgzone_pc file.

Variables

Signature (unsigned int) Magic header / unique signature found at the start of every valid rfgzone_pc file. Used to confirm that the file was properly extracted and to identify the format from other types. For a valid rfgzone_pc file this equals 1162760026. The same value is shared by layer_pc files which likely means they are the same format.

Version (unsigned int) The file format version. For RFGR this should always equal 36.

NumObjects (int) The number of objects in this zone when it's unaltered.

NumHandles (int) The number of handles in this zone when it's unaltered. Object handles can be used to reference an object. You can get an object using it's handle by passing it's handle to `rfg.GetObject`, which has two overloads. One which takes an objects name, a **string**, and one that takes an objects handle, an **unsigned int**.

DistrictHash (unsigned int) Likely a hash of the zones district name. Unconfirmed.

Flags (unsigned int) Stores flags determining the zones state or behavior. Exact values and uses unknown.

Attention: This page is incomplete and needs better descriptions and research into the behavior of the variables.

last updated: 09.08.2019 - added several descriptions for variables. By Vaxis

hkpSolverInfo

Values used by the physics solver. Mostly untested and unresearched for now. Should be able to look at open source physics engines and research papers to determine what these values do and their purposes. You generally shouldn't change these much if you want a stable game experience.

Variables

Tau (float) Vanilla Value: 0.600 Angular Velocity and rotational force (Torque) (Vanilla is recommended)

Damping (float) (Vanilla Value: 1.000) (Max recommended Value: 1.520) Damping is based on how bouncy a rigid object will behave when falling onto the ground.

FrictionTau (float) (Vanilla Value: 0.300) (Recommended Value: 1.780) – For a more realistic vision. Friction is a part of collision resolution. Friction always applies a force upon objects. In RedFactionGuerilla, it will make buildings and objects harder to break, and buildings will fall in bigger pieces.

DampDivTau (float) (Vanilla Value: 0.600) Torque and Damp Stress Values - !! - do not touch this. Editing this will not yield any beneficial results .

TauDivDamp (float) (Vanilla Value: 3.333) Effect is Unknown for now.

DampDivFrictionTau (float) (Vanilla Value: 0.300) Friction and Damp Stress Values - !! - do not touch this. Editing this will not yield any beneficial results .

FrictionTauDivDamp (float) Unknown value. Friction and Damp Stress Values - !! - do not touch this. Editing this will not yield any beneficial results .

ContactRestingVelocity (float) Friction and Damp Stress Values - !! - do not touch this. Editing this will not yield any beneficial results

DeltaTime (float) Unknown value. `ReadOnly`. Set by the game manually each frame/physics update based on the amount of time passed (educated guess).

InvDeltaTime (float) Unknown value. `ReadOnly`. Equals $1 / \text{DeltaTime}$.

NumSteps (int)

“Step” is a process of calculating system’s next state.

Timestep is the time interval for which simulation will progress during next “step”. It works simliar to Timestep multiplier.

NumMicroSteps (int) Half the Numstep value for better percision when editing values.

InvNumMicroSteps (float) Unknown value. `readonly`. Equals $1 / \text{NumMicroSteps}$.

InvNumSteps (float) Unknown value. `readonly`. Equals $1 / \text{NumSteps}$.

ForceCoherentConstraintOrderinginSolver (bool) Unknown value.

5.2.2 Functions

Lists all free functions in the `rfg` namespace. Note that any member functions such as `rfg.Object.CastAsHuman` will be displayed in their respective types page.

AddMessageBox

Adds a message box to the screen using RFGs UI system.

Arguments:

Type (MessageBoxTypes) The type of message box to create (OK, yes/no, accept/cancel, etc, see [MessageBoxTypes](#) for more info).

Title (String) The title of the message box.

Description (String) The description/content of the message box.

CallbackFunction (Function) optional A lua function which is used to process any user input for this message box. Useful for yes/no style message boxes that expect an option to be chosen. See the examples section for an example of how this works.

Button1Override (String) optional A custom name for the first button of the message box if you don’t want the standard “yes” “no” “ok” labels.

Button2Override (String) optional A custom name for the second button of the message box.

Returns:

Handle (int) The handle of the message box. Currently has no use.

Examples:

This example shows how to use a callback function to determine if the user selected yes or no on a yes/no message box:

```
--This function is called when the user selects on of the message box options
local function MyMboxCallback(CallbackData)
    if(CallbackData.Choice == rfg.MessageBoxChoices.Yes) then
        rsl.Log("Selected \"Yes\" option in the message box\n")
    elseif(CallbackData.Choice == rfg.MessageBoxChoices.No) then
        rsl.Log("Selected \"No\" option in the message box\n")
    end
end

--Create the message box, register it's callback function as the last arg
rfg.AddMessageBox(rfg.MessageBoxTypes.YesNo, "Test message box", "Player pos: " ..
↪Player.Position:ToString(), MyMboxCallback)
```

AddUiMessage

Creates a new ui message in the top left corner of the screen. Suitable for one-off messages, not suitable for constantly changing information.

Arguments:

Info (String) The message to be displayed.

DisplayTime (float) optional The amount of time in seconds to display the message for. Default value is 3.0 if not specified.

UseSecondaryAnim (bool) optional Whether or not to animate the background of the message. Default is false if not specified.

ForceRedisplay (bool) optional Unknown purpose. Default is false if not specified.

Returns:

- None

Examples:

```
--Call it with no optional arguments
rfg.AddUiMessage("Player position: " .. Player.Position:ToString())
--Call it with the DisplayTime optional arg
rfg.AddUiMessage("Player position: " .. Player.Position:ToString(), 8.0)
--Call it with the UseSecondaryAnim optional arg to animate the message
rfg.AddUiMessage("Player position: " .. Player.Position:ToString(), 8.0, true)
```

AddUserMessage

Creates a new user message in the specified position on the screen. Has no animation or sound, so it's useful for data that frequently changes

Arguments:

Text (String) The text to be displayed.

PositionX (float) The x position of the text on the screen. Uses normalized coordinates, so for example, 0.5 would be the middle of the screen.

PositionX (float) The y position of the text on the screen. Uses normalized coordinates, so for example, 0.5 would be the middle of the screen.

Outlined (bool) Whether or not the message should be surrounded with a grey rectangle.

Lifespan (float) The time in seconds that the message should be on screen.

Type (MessageTypes) The type of user message. The exact differences between these have not been fully explored, but you should use `rfg.MessageType.Other` if you don't want the message to be positioned based on other visible messages.

Returns:

Handle (int) The handle of the new user message. Can be used to update the messages text or to delete it.

ChangeUserMessage

Change the value of a pre-existing user message. Use `rfg.AddUserMessage` to create them.

Arguments:

Handle (int) The handle of the user message you want to edit. This is returned by `rfg.AddUserMessage` when you create a user message.

NewText (String) The new text to be displayed for the user message.

Returns:

Handle (int) The handle of the user message you've edited. This shouldn't be different from the one you entered.

ExplosionCreate

Spawns an explosion.

Arguments:

Info (ExplosionInfo) The properties of the explosion to be spawned. You can use `rfg.GetExplosionInfo` to get one of the games explosion presets, or iterate the preset list as shown in the [explosion examples](#).

Position (Vector) The position to spawn the explosion in.

Alternative:

Info (ExplosionInfo) The properties of the explosion to be spawned. You can use `rfg.GetExplosionInfo` to get one of the games explosion presets, or iterate the preset list as shown in the [explosion examples](#).

x (float) The x position to spawn the explosion in.

y (float) The y position to spawn the explosion in.

z (float) The z position to spawn the explosion in.

Alternative:

Info (ExplosionInfo) The properties of the explosion to be spawned.

Source (Object) The source of the explosion.

Owner (Object) The owner of the explosion.

Position (Vector) The position to spawn the explosion in.

Orientation (Matrix) The orientation to spawn the explosion at.

Direction (Vector) The direction the explosion should point in.

Returns:

- None

GetAlertLevel

Gets the current alert level (green, yellow, orange, red).

Arguments:

- None

Returns:

Level (**AlertLevel**) The current alert level.

GetExplosionInfo

Attempts to get the **ExplosionInfo** data for an explosion of the inputted name. Alternatively you could iterate the explosion preset list, as explained in one of the [explosion examples](#).

Arguments:

Name (**String**) The name of the explosion preset to get info for.

Returns:

Info (**ExplosionInfo**) The explosion info. Will be `nil` if no explosion with the specified name was found. You can use the explosion spawning menu or write a simple logging script to get a list of all explosion preset names.

GetFarClip

Gets the far clip distance for the camera.

Arguments:

- None

Returns:

Distance (**float**) The current far clip distance of the camera.

GetGravity

Get's the current gravity acceleration vector. Each component is in m/s^2 .

Arguments:

- None

Returns:

Gravity (**Vector**) The current gravity acceleration vector.

GetHighLodFarClip

Gets the high lod far clip distance for the camera.

Arguments:

- None

Returns:

Distance (float) The current high lod far clip distance of the camera.

GetObject

Attempts to find a game object with the provided name or handle. Most objects don't have names, so the Handle overload is more useful.

Arguments:

Name (String) The name of the object to find.

Alternative:

Handle (unsigned int) The handle of the object to find.

Returns:

Object (Object) May be `nil` depending on if an object of the given name or handle is found.

GetVersion

Arguments:

- None

Returns:

Version string (String) Returns the version string of the rfg build that you are using. At the time of writing this page the version string is `cs:4931`. While the game hasn't been updated in nearly a year at this point, this could prove useful to make sure people are using an up to date version of the game.

HavokBodyApplyLinearImpulse

Applies an impulse to a havok rigid body at it's center of mass.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

LinearImpulse (Vector) The impulse to be applied.

Returns:

HavokBodyApplyPointImpulse

Applies an impulse to a havok rigid body from a specified position.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

Impulse (Vector) The impulse to be applied.

Position (Vector) The position to apply the impulse from.

Returns:

HavokBodyForceActivate

Activates the specified havok rigid body. Physics objects can get deactivated when they aren't needed to save on performance. This activates an object instead.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

Returns:

HavokBodyGetPosOrient

Gets the position and orientation of a havok rigid body.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

Position (Vector) This vector will be set to the position of the havok rigid body. This should already be initialized by you.

Matrix (Matrix) This matrix will be set to the orientation of the havok rigid body. This should already be initialized by you.

Returns:

HavokBodySetMovable

Sets whether or not a havok body is movable. Does not work on all objects, notably buildings, and you may notice that some objects have no collision once made movable.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

Movable (bool) Whether or not the body should be made movable.

Returns:

HavokBodySetPosOrient

Sets the position and orientation of a havok rigid body.

Arguments:

Handle (unsigned int) The handle of the havok rigid body.

Position (Vector) The new position of the body.

Matrix (Matrix) The new orientation matrix of the body.

Returns:

HavokBodySetPosition

Sets the position of a havok rigid body.

Arguments:

Handle (**unsigned int**) The handle of the havok rigid body.

Position (**Vector**) The new position of the body.

Returns:

HideFog

Sets the visibility state of fog.

Arguments:

Hide (**bool**) If true fog will be hidden. If false fog will be made visible.

Returns:

- None

HideHud

Sets the visibility state of the hud.

Arguments:

Hide (**bool**) If true the hud will be hidden. If false the hud will be made visible.

Returns:

- None

OpenUpgradesMenu

Opens the upgrade menu that normally shows up when you use the upgrades bench.

Arguments:

- None

Returns:

- None

OpenWeaponSelector

Opens the weapon selector menu that normally shows up when you open a weapons locker or ammo box.

Arguments:

- None

Returns:

- None

RemoveUserMessage

Remove a user message from the screen completely. Use `rfg.AddUserMessage` to create them. You should not use this messages handle after removing it.

Arguments:

Handle (int) The handle of the user message you want to remove.

Returns:

- None

SetAlertLevel

Sets the current alert level (green, yellow, orange, red).

Arguments:

Level (AlertLevel) The new alert level.

Returns:

- None

SetFarClip

Sets the far clip distance for the camera. Anything further than this distance is not rendered. Note that this cannot be used to view the entire game world at once. Even if this is set to obscene values only so much of the world is loaded at once as is.

Arguments:

Distance (float) The new far clip distance.

Returns:

- None

SetGravity

Set's the current gravity acceleration vector. Each component is in m/s^2 .

Arguments:

Gravity (Vector) The new gravity acceleration vector.

Alternative:

x (float) The x component of the new gravity acceleration vector.

y (float) The y component of the new gravity acceleration vector.

z (float) The z component of the new gravity acceleration vector.

Returns:

- None

SetHighLodFarClip

Sets the high lod far clip distance for the camera. Sets the distance at which terrain and objects are rendered at high lod. Increasing can slightly improve distance terrain textures.

Arguments:

Distance (float) The new high lod far clip distance.

Returns:

- None

SetShadowResolutions

Sets the shadow resolutions used by the game. Need to change a graphics setting or resize the game window for the change to go into effect. Pressing alt + enter twice is an easy way to do this. These values are identical to the ones you can set at the bottom of the graphics tweaks menu.

Arguments:

res0 (int) Shadow res 0.

res1 (int) Shadow res 1.

res2 (int) Shadow res 2.

res3 (int) Shadow res 3.

Returns:

- None

TeleportHuman

Teleports a human to the specified position. If you want to teleport the player there's a helper function to make that easier, [TeleportPlayer](#).

Arguments:

Target (Human) The human to teleport.

Position (Vector) The position to teleport the target to.

Orientation (Matrix) optional The orientation (direction they're facing) of the target that should be set after they're teleported.

Returns:

- None

TeleportPlayer

Teleports the player to the specified position. If you want to teleport other humans you should use [TeleportHuman](#).

Arguments:

Position (Vector) The position to teleport the player to.

Orientation (Matrix) optional The orientation (direction they're facing) of the player that should be set after they're teleported.

Returns:

- None

ToggleFog

Toggles the visibility of fog. If fog is already visible it will be hidden and if it's already hidden it'll be made visible.

Arguments:

- None

Returns:

- None

ToggleHud

Toggles the visibility of the hud. If the hud is already visible it will be hidden and if it's already hidden it'll be made visible.

Arguments:

- None

Returns:

- None

5.3 rsl

This table contains the functions and types used by rsl. This mainly consists of helper functions for things like logging.

5.3.1 Types

Lists all types functions in the rsl namespace.

LogType

Used by logger to categorize each message logged. Also used to determine which log files the message will be printed to. For example, a message with `LogType.All` will show up in all logs, while one with `LogType.Info` will only show up in logs with that flag set with their filtering settings.

Value	Resulting Log Tag
None	No tag
Info	[Info]
Warning	[Warning]
Error	[Error]
Fatal Error	[Fatal Error]
Lua	[Lua]
Json	[Json]
All	No tag

5.3.2 Functions

Lists all free functions in the `rsl` namespace.

GetVersion

Arguments:

- None

Returns:

Version (String) Returns a string for the current RSL version. For example, the result for version 0.4.0 is `0.4.0-Alpha`. This can be used to confirm that the user has a new enough version of RSL before running a script.

Log

Logs a message to the lua console, logger window, and log files. This is the default logging function and is equivalent to `rsl.LogInfo`, meaning it has the `[Info]` tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogError

Logs a message to the lua console, logger window, and log files. Has the `[Error]` label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogFatalError

Logs a message to the lua console, logger window, and log files. Has the `[Fatal Error]` label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogNone

Logs a message to the lua console, logger window, and log files. Has the [Info] label tag. See the [logging examples](#) for more info on how to use it. Equivalent to `rsl.Log`.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogJson

Logs a message to the lua console, logger window, and log files. Has the [Json] label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogLua

Logs a message to the lua console, logger window, and log files. Has the [Lua] label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogNone

Logs a message to the lua console, logger window, and log files. Has no label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

LogWarning

Logs a message to the lua console, logger window, and log files. Has the [Warning] label tag. See the [logging examples](#) for more info on how to use it.

Arguments:

Format string (String) The format string to be passed into the logger.

Arguments (Several types) optional Values to be substituted into the format string, as explained in more detail in the [logging examples](#).

Returns:

- None

This page contains a list of guides for lua scripting for RFGR using the RSL. Note that these guides will often skip explaining the syntax of lua. There are already many guides on using lua online so it's better to keep these guides focused on scripting for RFGR.

Warning: Lua scripting is not a mature feature. Expect there to be breaking changes as it's continually updated and tweaked. Be sure that you can support and update your mod in case of any breaking changes.

6.1 Autorun scripts

RSL supports scripts which automatically run once RSL activation is complete (when you hear the 3 activation beeps). Using them is simple. All it requires is that a script named `main.lua` exists in one of the subfolders of the Scripts folder. RSL checks all the subfolders of the scripts folder `/RSL/Scripts/some_subfolder` for files named `main.lua` and tries to run them upon RSL activation.

Note: Depending on when you inject the RSL your autorun scripts could run before certain values like the player or world structures have been initialized. You should double check that `rfg.ActivePlayer`, `rfg.ActiveWorld`, `rfg.ExplosionInfos`, and `rfg.PhysicsSolver`, are not `nil` before using them in an autorun scripts code that is immediately executed.

6.1.1 Console commands

A lua console is provided by the RSL. It's available by pressing F4. The console's inputs are ran directly by the lua interpreter, they're essentially single line scripts. This also means that adding console commands is very easy. You simply run a script with a function definition, and as long as there are no errors that function will be available for use in the lua console. The only downside to this is that you'll need to run the script with that function every time you start the game, by using an autorun script you can automate that process. Below is an example of an autorun script which adds a useful console command:

```
-- Takes an object name or handle and reports back on if an object with that
-- name/handle exists and what it's position is.
function ObjectExists(TargetName)

    TargetObject = rfg.GetObject(TargetName) --GetObject has an overload
    --that takes a name, and an overload that takes a handle.

    if TargetObject == nil then
        rsl.Log("Target object of name/handle \"{}\" not found\n", TargetName)
    else
        rsl.Log("Target object of name/handle \"{}\" found\n", TargetName)
        rsl.Log("Position: {} \n", TargetObject.Position:GetDataString())
    end
end
```

If you don't want a function to be available in the lua console, simply mark it as a local function with the `local` keyword.

6.2 Script events

The RSL supports binding lua functions to different events. This allows you to have functions that run when certain things such as mouse clicks or keypresses occur. There's also a per-frame event available for anything the other events don't cover, but it's preferred to use more specific events when possible. See the [Events](#) page for a list of events that can be used in scripts currently.

6.2.1 Registering events

Below is an example of how to register a lua function to be called when an event occurs. In this case, we're using a Keypress event to detect when the q key is pressed and toggle the hud.

```
--This function will be called each time the event it's registered to is triggered
--EventData will contain the data specific to that event
--In this case, it's info on what keys are being pressed
function MyKeypressEvent(EventData)
    if(EventData.KeyCode == rfg.KeyCodes.q) then
        rsl.Log("q pressed! Toggling Hud\n")
        rfg.ToggleHud()
    end
end

--First arg is the type of event to register the func to
--^Also known as the "RegisterId"
--Second arg is the function to register
--Third arg is just a label for the event viewer. Useful for debugging.
rfg.RegisterEvent("Keypress", MyKeypressEvent, "ToggleHud keypress event hook")
```

So, the process for using events is simple. 1. Choose your event type. 2. Write a function that you want to be called each time that event is triggered. 3. Register your function with that event.

6.3 Introduction

Here you'll find all the info you need to start writing lua scripts for RFGR. While traditional modding involves editing existing data files to change the properties and behavior of the game, lua scripting allows for a much more dynamic modding experience. Some of the advantages of lua scripting are:

- You aren't restricted to editing existing data found in xtbls, due to the nature of the RSL the game can be opened up much more than before.
- Lua scripts are responsive and can change their behavior over time, unlike xtbls which are simply static data files.
- You can edit lua scripts live instead of needing to restart -> unpack -> edit -> repack -> test -> repeat for even the smallest of changes. The higher iteration rate should speed up mod development and reduce hair pulling.

Warning: RSL is still in early stages of development. As such, you may find incomplete features, bugs, and experience breaking changes in updates. Keep this in mind before using it in it's current state.

6.3.1 Calling lua scripts

Currently calling lua scripts is as simple as placing a .lua file in your Scripts folder, `Red Faction Guerrilla Re-MARS-tered\RSL\Scripts` and running it through the built in script editor, or script select menu. You might need to hit the refresh button if the script was created externally after RSL was started.

You can also open the built-in script editor after loading RSL and save/load/create scripts right from the game. It's good for quick edits and testing, but I'd still recommend using an external editor for major editing.

A later release will feature more advanced methods for loading and running scripts. Likely having each script, set of scripts, or mod, be it's own package with information like version, and dependent packages.

Note: While the RSL includes a built in script editor with some useful features like syntax highlighting and error highlighting, it's still recommended to use a external editor for large amount of editing since they can offer more features than the internal one does.

6.3.2 Core library

The core library, in `Red Faction Guerrilla Re-MARS-tered\RSL\Core`, includes useful functions, and types. It's loaded when RSL is started. You generally shouldn't distribute edits to this with a mod as it's meant to be a common library that all users have. You should however take a look at it. There are many useful values defined in the core library which are referenced in the docs.

One example of this is `ObjectTypes.lua`, which defines the table `rfg.ObjectTypes`. This table has all object types (which will be discussed more in the next section), and their corresponding integer values, which can be used to sort objects by type.

6.3.3 Objects

RFGR considers many things in the game world to be objects. Many of the things you can see in the game are objects, such as the player, humans, or vehicles, and even some things you can't see, like spawn points and cliff killzones. You can see a full list of object types in `ObjectTypes.lua`, or in the `ObjectTypes` page here in the docs.

Since objects are so pervasive, being able to access them is key to interacting with the game world.

Accessing objects

There are several ways to access objects. The first way to is use `rfg.GetObject`. It takes a string or a number, which is the name or handle of the object you're looking for, respectively. Using this with a name isn't the most reliable way of finding objects, since most of them don't have names. But, passing a handle to it can be quite useful since many objects store handles to other objects. For example, `Human.ShieldHandle` is the object handle of that humans shield if they possess one. Another way of accessing objects is by looping through the world object list, like so:

```
HumanCount = 0
for i=0, rfg.ActiveWorld.AllObjects:Size(), 1 do --Loop through the global object list
    CurrentObject = rfg.ActiveWorld.AllObjects[i] --Make a reference variable to the
    ↪current object for convenience.
    if CurrentObject.Type == rfg.ObjectTypes.Human then --Check if current object is
    ↪a human object
        HumanCount = HumanCount + 1
    end
end
rsl.Log("HumanCount: {} \n", HumanCount)
```

Note: While lua tables use 1 based indexing, the rfg object list uses 0 based indexing. This is a side effect of c++ using 0 based indexing, but, this may be changed in a future update to avoid inconsistency with existing lua standards.

Note that if you tried accessing a human specific variable (or other derived types variable) before casting to that object type you should experience an error when running the script. You can use functions like `Object:AsHuman()` to cast the type to a human type, or the equivalent for the object type you're dealing with. The example below loops through the global object list, finds human objects, then casts the object to a human object and accesses `HitPoints`, which is a variable that humans have, but other objects dont.

```
for i=0, rfg.ActiveWorld.AllObjects:Size(), 1 do --Loop through the global object list
    CurrentObject = rfg.ActiveWorld.AllObjects[i] --Make a reference variable to the
    ↪current object for convenience.
    if CurrentObject.Type == rfg.ObjectTypes.Human then --Check if current object is
    ↪a human object
        if CurrentObject.AllIndex ~= rfg.ActivePlayer.AllIndex then --Make sure this
    ↪human object isn't the player
            ObjectAsHuman = CurrentObject:AsHuman()
            ObjectAsHuman.HitPoints = 0 --Kill this human instance by setting it's
    ↪HitPoints to 0
        end
    end
end
```

Important: As of release 0.5.0 not all object types have been bound to lua yet due to time constraints. Therefore you can only cast to a few of them for now. The available object types for 0.5.0 are Human, Player, Zone, and District. For all other object types you'll only be able to access variables available to all objects.

6.3.4 What next

There are many other functions, types, and values available to scripts. Too many to list here. To see a list of them and more details you should view the [API](#) page. For more usage examples you should read the rest of the [guides](#), and look through the [examples](#) provided. You should also look through the [useful values](#) page for info on some useful preset values like `rfg.ActivePlayer` and `rfg.PhysicsSolver`. If you'd like to contribute the the docs you should read [contributing](#).

6.4 Lua overrides

While a major goal of RSL is to add a lua scripting api to RFG, the game infact already has lua built in. Albeit, it's a very old version of lua 5.0, and it's not used very extensively. RFG uses about a dozen lua scripts to provide some of it's user interface logic.

This guide describes how to use the lua overrides feature which lets you override rfg lua scripts with your own versions of them. You can already edit these scripts by extracting rfgs vpp_pc and str2_pc files and locating them. There are a few in `misc.vpp_pc`, and a few hidden in `str2_pc` files in `interface.vpp_pc`. That however, is a hassle, and you won't be needing to do that for this. The lua overrides feature lets you access and replace these scripts without ever touching the data folder.

Important: Override scripts currently do not have access to the RSL scripting api. This means that override scripts can only use things built into lua 5.0 and things already exposed to the rfg lua state. This is because RSL and rfg have two separate lua states, and manipulating both creates additional technical challenges. This also means that error reporting for override scripts is generally worse and errors in overrides are more likely to crash the game.

6.4.1 Dumping rfg scripts

Dumping rfg's built in scripts is an automatic process. Each time one is loaded RSL will save a copy of it in `/RSL/RFG_Lua/Lua_Dumps`. You'll also see a message in the log each time a script is dumped, such as `[Info] Dumping rfg_ui_safehouse_weapons.lua`, which is the script for the safehouse weapons locker gui.

Scripts are only dumped when they are run, and some scripts are only run at startup. Therefore, the best practice to dump all rfg scripts is to run the RSL from game startup and open all menus you can except MP and wrecking crew menus, since they'll crash the game with the RSL active.

6.4.2 Overriding rfg scripts

To override a script you must make a script with the same name as it in `RSL/RFG_Lua/Overrides/`. So for example, if you wanted to override `rfg_ui_globals.lua`, you'd create a file with that name in the overrides folder, and make your edits there. From there the real script will automatically be overridden next time it's ran by the game. A message will be logged each time a script is overridden, such as `[Info] Overriding rfg_ui_globals.lua`.

Note: Some scripts, such as `rfg_ui_globals.lua` are only run once at game startup, and so to override them you must run RSL from game startup and have your override script ready in the overrides folder. Scripts that are called many times like the weapons locker ui script can be repeatedly edited and tested while the game is running by repeatedly opening and closing the related ui.

CHAPTER 7

Examples

Lua scripting examples. Also check the “Included scripts” folder with each RSL release for more examples.

7.1 Explosions

RSL lets you spawn explosions, create custom explosions, and globally modify explosion values via lua scripts.

7.1.1 Getting explosion info

`ExplosionInfo` is a very important type when it comes to handling explosions. There are several ways to get the `ExplosionInfo` instance for the games preset explosions. One way is by passing the explosions name to `rfg.GetExplosionInfo` as shown in the example below:

```
MineExpInfo = rfg.GetExplosionInfo("mine")
if MineExpInfo ~= nil then
    rsl.Log("Radius: {}\\n", MineExpInfo.Radius)
end
```

You can find the names of all the explosion presets in the explosion spawner gui. The other way is to loop through the `ExplosionInfo` list. As shown below:

```
for i = 0, rfg.ExplosionInfos:Size(), 1 do --Go through the ExplosionInfo list with a
    ↪for loop
    rsl.Log("{}: {}\\n", i, rfg.ExplosionInfos[i].Name) --Log the name of each
    ↪explosion preset.
end
```

7.1.2 Global and local explosion infos

Using the two previous methods provides you with what would be considered “global” `ExplosionInfo` instances. That means that if you edit the values of these instances, it will change those values for all explosions spawned from that

preset. If you wanted to have a “local” `ExplosionInfo` instance you should make a copy of the explosion preset you want to edit, and edit that copy. Then spawn the explosion using that copy. The example below shows this:

```
ChargeExpInfo = rfg.GetExplosionInfo("remote_charge") --Get the remote charge
↳ExplosionInfo preset.
ChargeExpInfo.Radius = 20.0
--Set the radius for the remote charge preset.
--Since no copy was made this is editing the global instance of this preset.
--Meaning that all remote charge explosions will have increased radius now.

ChargeExpInfoCopy = rfg.ExplosionInfo:new(ChargeExpInfo)
ChargeExpInfoCopy.ImpulseMagnitude = 120000
--Since this is editing the copy, normal remote charge explosions will not be
↳effected.
--Only explosions you spawn by using the copy will have this increased impulse.
```

7.1.3 Spawning explosions

Explosions can be spawned by using `rfg.ExplosionCreate`, as shown below:

```
GrenadeExp = rfg.GetExplosionInfo("grenade_exp") --Get grenade explosion info
rfg.ExplosionCreate(GrenadeExp, 1434.0, 15.0, 693.3) --Spawn grenade explosion
↳outside the Oasis safehouse
```

Be sure to check the page for `rfg.ExplosionCreate` for additional overloads of the spawning function.

7.1.4 More examples

This first example goes over a bit of each of the previous subjects.

```
-- Get a pointer to the remote charge ExplosionInfo instance
RemoteChargeExp = rfg.GetExplosionInfo("remote_charge")
RemoteChargeExp.Radius = 12.0 --Change the remote charge damage radius
RemoteChargeExp.SecondaryRadius = 15.0

-- If you don't want to edit the actual explosion values you need to make a copy
RemoteChargeExpCopy = rfg.ExplosionInfo:new(RemoteChargeExp)
-- Now we can edit the values of RemoteChargeExpCopy without changing the global
-- remote charge explosion values.
RemoteChargeExpCopy.AiSoundRadius = 2.0
RemoteChargeExpCopy.Radius = 25.0
RemoteChargeExpCopy.SecondaryRadius = 30.0
RemoteChargeExpCopy.ImpulseMagnitude = -100000.0

SpawnPosition = rfg.Vector:new(rfg.ActivePlayer.Position)
SpawnPosition.y = SpawnPosition.y + 10.0
-- Note that here we're not exactly spawning a remote charge explosion
-- But a modified version of it
rfg.ExplosionCreate(RemoteChargeExpCopy, SpawnPosition)
-- ^^ Also has an overload with many more args. Check the docs for more info on that.

-- You can also iterate through the games explosion list, as shown with the func
↳below:

-- Normally it's good practice to stick console commands like this func
```

(continues on next page)

(continued from previous page)

```
-- in their own file and to label them as such. Placing it here for examples sake.
function ListExplosionInfoNames()
    for i = 0, rfg.ExplosionInfos:Size(), 1 do
        rsl.Log("{}: {}\n", i, rfg.ExplosionInfos[i].Name)
    end
end
```

The next example shows an example of how to change the values of all explosion presets.

```
Multiplier = 2.0

for i = 0, rfg.ExplosionInfos:Size(), 1 do
    IndexExp = rfg.ExplosionInfos[i];
    IndexExp.Radius = IndexExp.Radius * Multiplier
    IndexExp.SecondaryRadius = IndexExp.SecondaryRadius * Multiplier
    IndexExp.Impulse = IndexExp.Impulse * Multiplier
end
```

7.2 Logging

RSL provides several options for logging values via lua scripts. The simplest way to log something is to pass a string to `rsl.Log`, like so:

```
rsl.Log("Test log\n")
```

Note that you must pass your own newline character `\n` for each message you log unless you want the next message to be on the same line.

7.2.1 Formatting

It also supports formatting by passing your format string as the first argument, and the values to format as the remaining arguments. One way to do this is shown below:

```
a = 10
b = 27.5
c = "dog"
d = 2

rsl.Log("a: {}, b: {}, c: {}, d: {}, a + d: {}\n", a, b, c, d, a + d)
```

```
[ Info ] a: 10.0, b: 27.5, c: dog, d: 2.0, a + d: 12.0
```

Fig. 1: Output for the above logging example

If you wanted to have the same formatting as above without using a format string you'd have to do something like this:

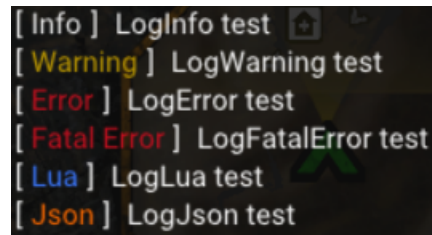
```
a = 10
b = 27.5
c = "dog"
d = 2
rsl.Log("a: " .. tostring(a) .. ", b: " .. tostring(b) .. ", c: " .. c .. ", d: " ..
tostring(d) .. ", a + d: " .. tostring(a + d) .. "\n")
```

Which is much longer and messier (which means more chance of bugs), and likely less performant. Hopefully with these examples it's clear how formatting is extremely useful. Several different format string syntaxes are supported, such as printf style formatting or c# style formatting. See this page for more examples of fmt string syntax: <https://fmt.dev/latest/syntax.html>

7.2.2 Filtering

There are also logging functions which let apply different filtering tags to your log message, as shown below:

```
rsl.LogNone("LogNone test\n") -- Has no filtering tag
rsl.LogInfo("LogInfo test\n") -- Really the same thing as rsl.Log
rsl.LogWarning("LogWarning test\n")
rsl.LogError("LogError test\n")
rsl.LogFatalError("LogFatalError test\n")
rsl.LogLua("LogLua test\n")
rsl.LogJson("LogJson test\n")
```



```
[ Info ] LogInfo test
[ Warning ] LogWarning test
[ Error ] LogError test
[ Fatal Error ] LogFatalError test
[ Lua ] LogLua test
[ Json ] LogJson test
```

Fig. 2: Output for each tagged logging function.

If you look in the RSL logs folder, you'll see several files after running RSL at least once such as `Error Log.txt`, `Lua Log.txt`, and `General Log.txt`. These logs contain anything you log with one of the rsl logging functions and are filtered based on which logging tags you applied to each message.