DebugScanner

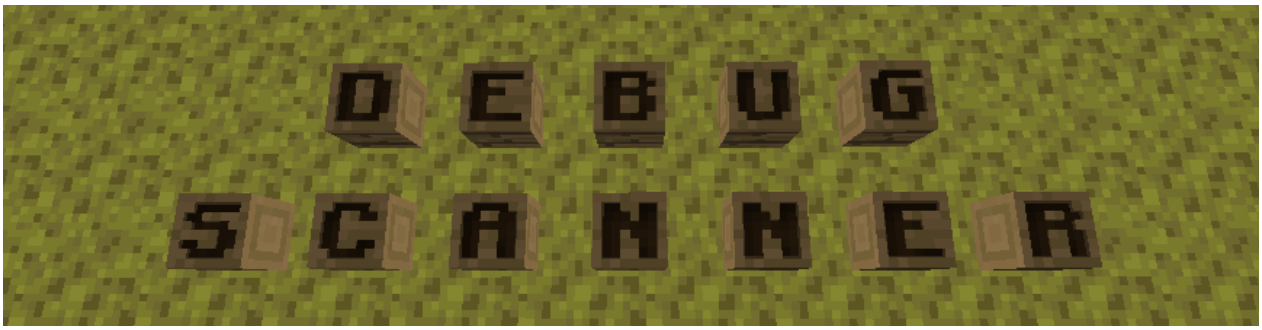
DebugScanner Developers

Table of contents

Introduction

Contributing

Introduction



DebugScanner is really only of use to those involved with the GeyserMC project. It is to be used with a Debug world and provides tools to allow testing block states between various editions of Minecraft.

Auto scan will will put the player into creative mode and teleport them at a configured interface to each block in succession with the block number shown in the action bar.

It will start with the block at (1 70 1) and will continue till it cannot find any more blocks. The first block is always assumed to be air.

Video *(https://www.youtube.com/watch?v=-p5zkeR5nZA)*

How to Use¶

1. Generate a Debug world by creating a single player world and holding shift when selecting type. One of the options will be Debug.
2. Issue the following command:

   Example

   /gamerule randomTickSpeed 0

3. Import the world into spigot by copying the save file and renaming to world. You will also need to rename level_old.dat to level.dat or copy a level.dat from a non debug world (note if you do this you'll need to set the gamerule again on first load).
4. Start up the server and join the world
5. Use for the Auto Scannner the following command to start:

   Example

/debugscanner auto start [-start {block number}] [-interval {ticks}] [-pitch {pitch}] [-yaw {yaw}] [player]

6. To stop early use the following command:

Example

/debugscanner auto stop [player]

Last update:

Contributing¶

Here are some ways that you can help contribute to this project.

New ideas or Bug Reports¶

Need something? Found a bug? Or just have a brilliant idea? Head to the Issues *(https://github.com/Bundabrg/DebugScanner/issues)* and create new one.

Contributing Code¶

If you know Java then take a look at open issues and create a pull request.

Do the following to build the code:

```
git clone https://github.com/Bundabrg/DebugScanner
cd DebugScanner
mvn clean package
```

Contributing Documentation¶

If you can help improve the documentation it would be highly appreciated. Have a look under the `docs` folder for the existing documentation.

The documentation is built using `mkdocs`. You can set up a hot-build dev environment that will auto-refresh changes as they are made.

Requirements¶

- python3
- pip3
- npm (only if changing themes)

Install dependencies by running:

```
pip3 install -r requirements.txt
```

Dev Environment¶

To start a http document server on `http://127.0.0.1:8000` execute:

```
mkdocs serve
```

Change PDF Theme¶

Edit the PDF theme under `docs/theme/pdf`. Rebuild by doing the following:

```
cd docs/theme/pdf
npm install
npm run build-compressed
```

This will update `pdf.css` under `docs/css/pdf.css`. Rebuilding the docs will now use the new theme.

Last update: