VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Minecraft: Java Edition plugin "Woke Village"

Done by:

Nazar Mamedov

Emilijus Šileikis

Robertas Timukas

Patrik Voronovič

Supervisor:
dr. Linas Būtėnas

Vilnius
2022

Preface

This project was done during the 3rd semester of the study programme *Information Technologies* in the specialization of *Innovative studies*. We chose the topic *Minecraft NPC plugin* suggested during the project market on the first lecture of the subject ‘‘Problem-Based Project’’. The topic sounded very interesting because it involved development "on top" of an already existing software, which is not commonly taught in lectures. This meant we would have to improvise, test and research in a completely different way which would definitely improve our problem-solving abilities. The software in question also brought back fond memories of our childhood and we felt an inspiration to further improve that enjoyment to the new generation, by adding more fun functionality to the game.

June 13, 2022

Nazar Mamedov

Emilijus Šileikis

Robertas Timukas

Patrik Voronovič

Contents

Abstract

Minecraft has a huge procedurally-generated world with virtually infinite terrain, however, significant gaps exist in its features. One of the most lackluster additions were villagers, which are the only types of NPCs (Non-Playable Character) that communicate and trade with the player. The functionality is extremely limited and stale, they do not interact with terrain, do not pursue any goal, do not have personalities and their trades consist of completely random items and are limited by the cost of rare emeralds. Those villagers have different roles, but only one of them is entirely functional (farmer villager can sow seeds), while others are mainly re-skins with a few additional items in rotation within their trades. In order to fill these gaps, we created "Woke Village", which is a plugin for version 1.17.1 of Minecraft: Java Edition server using "Paper", to add a completely new type of NPCs.

"Woke Village" introduces a collection of active NPCs. They can have one of three roles and every role has the same rotation of random personalities. These villagers can be spawned by curing a zombie-villager and can roam around the world providing gathering services for the player. Minecraft is often regarded as a "grindy" game, which means the player has to do a lot of repetitive and simple tasks, to save up enough resources to build and progress through the game. This burden is entirely eliminated with the help of our NPCs. They can gather basic resources in bulk, but contrary to Minecraft villagers, they actually walk to their desired resource, take time to gather it and then come back to the place where the deal was struck and give those resources to the player. This approach also intensifies the immersion and gives the player an incentive to work with multiple NPCs at the same time.

We strived to create an easily accessible addition to the game, which would not force the player to download external files. Malware creators often mask their applications by renaming the file to something popular amongst children and putting it on their own websites, which is why we want to minimize the need for downloading. To enable the plugin, only the servers creator would require to install it before launch and the games target audience - mainly 5 to 15-year-olds, can enjoy it without risking accidental malware.

Santrauka

Minecraft: Java Edition papildinys
"Woke Village"

„Minecraft“ turi didžiulį procedūriniu būdu sukurtą pasaulį su beveik begaliniu reljefu, tačiau jo ypatybėse yra didelių spragų. Vienas iš silpniausių papildymų žaidimui buvo kaimo gyventojai, kurie yra vieninteliai nežaidžiami personažai, kurie prekiauja ir bendrauja su žaidėju. Jų funkcionalumas yra labai ribotas ir pasenęs, jie nesąveikauja su reljefu, nesiekia jokio tikslo, neturi asmenybių, o jų sandorius sudaro visiškai atsitiktiniai daiktai ir jų pirkimą riboja retų smaragdų kaina. Kaimo gyventojai atlieka skirtingus vaidmenis, tačiau tik vienas iš jų yra visiškai funkcionalus (kaimietis ūkininkas gali sėti sėklas), o kiti yra perdarytos išvaizdos su keliais papildomais daiktais savo prekių sąraše. Norėdami užpildyti šias spragas, sukūrėme „Woke Village“, kuris yra bet kurio „Minecraft: Java Edition“ serverio papildinys, kuris prideda visiškai naują nežaidžiamų personažų tipą.

„Woke Village“ pristato aktyvių nežaidžiamų personažų kolekciją. Jie turi vieną iš trijų profesijų ir kiekviena profesija turi tą pačią atsitiktinių asmenybių rotaciją. Šie kaimo gyventojai atsiranda išgydžius "zombie-villagers" ir gali keliauti po pasaulį, teikdami žaidėjui išteklių rinkimo paslaugas. „Minecraft“ dažnai laikomas "grindy" žaidimu, tai reiškia, kad žaidėjas turi atlikti daug pasikartojančių ir paprastų užduočių, kad sutaupytų pakankamai išteklių statyti ir tęsti progresą. Ši našta visiškai pašalinama su mūsų nežaidžiamų personažų pagalba. Jie gali surinkti visus pagrindinius išteklius dideliais kiekiais, tačiau, priešingai nei „Minecraft“ kaimo gyventojai, jie iš tikrųjų eina prie norimų išteklių, užtrunka juos rinkdami, grįžta į vietą, kurioje buvo sudarytas sandoris, ir atiduoda tuos išteklius žaidėjui.. Šis metodas taip pat skatina žaidėjo įsigilinimą į žaidimo pasaulį ir suteikia paskatą dirbti su keliais nežaidžiamais personažais tuo pačiu metu.

Mes siekėme sukurti lengvai prieinamą žaidimo priedą, kuris nepriverstų žaidėjo atsisiųsti išorinių failų. Norėdami įjungti papildinį, tik administratorius turės jį įdiegti prieš paleisdamas, o žaidimų tikslinė auditorija - daugiausia 5–15 metų amžiaus vaikai - gali ja mėgautis nerizikuodami atsitiktine kenkėjiška programa.

Introduction

Minecraft is a popular video game that can be modified with mods and plugins. We believe that default trading system among the in-game villagers can be boring and monotonous. So, the main goal of our “Woke Village” plugin is to expand the boundaries of the perception of the default game trading system by creating new type of NPC’s. Our NPC will have different roles and personalities which will affect the trading process and the perception of the game in general. “Paskui solutions” team is trying to pursue this goal in order to give a user a brand new experience while playing on vanilla Minecraft server. Our game extension is all about making the interaction with NPC’s more entertaining and useful. In this report you will find written and visual information on used algorithms and command execution processes with all the additional details about it. This report will also contain instances of pseudo-code and graph with all possible roles and personalities with full description of them. Our team included all the functional and non-functional requirements of this project where you can see the full functionality and the specifications of the plugin. The examples of testing are in this paper as well. Those cases can be useful in order to see the progress of the development and bug fixing. And lastly, we will bring our conclusions and recommendations for the future users.

1 Project Deployment

This project is built on more dependencies, than a regular one, because we are adding functionality "on top" of an already existing project. Firstly we are relying on PaperMC [5] which extends and improves the Bukkit and Spigot APIs which introduces more features and functionality. In the graph (Figure 1) [4] we can see the services we use. The Minecraft: Java Edition game server [3] (later Paper) is configured and ran by the game administrator using Paper.jar file on which this project (WokeVillage.jar) depends on and is launched by. Paper looks for plugins folder in which our plugin will installed and using JavaPlugin interface, Paper will launch it.

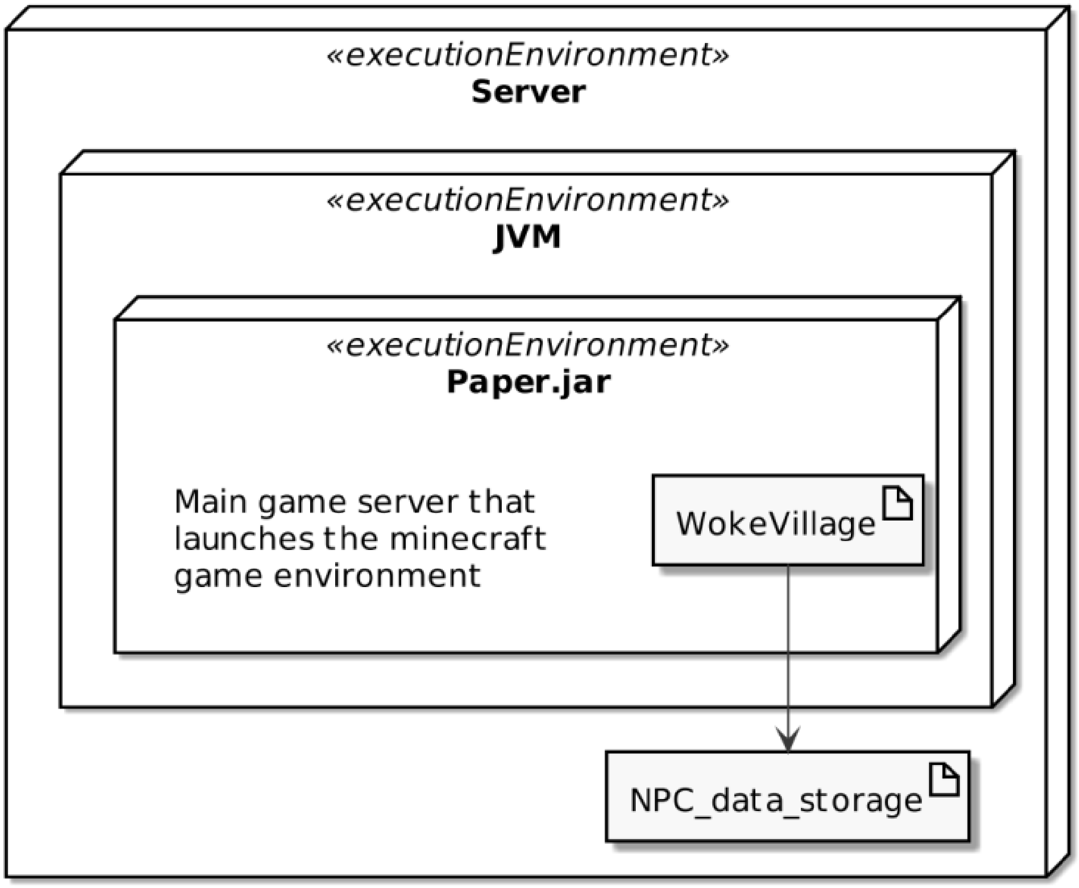

Figure 1. Deployement Graph

2 Gameplay

2.1 Starting

When the "Woke village" plugin is successfully installed, every player can use a "*/npc help*" command to know more about our implemented features. A non-admin player can spawn our NPC only by curing a zombie-villager. Precisely, there is a 100% chance to spawn the NPC, when the zombie-villager is of the "nitwit" profession. Other professions such as librarian, farmer and etc. have a 25% of spawning the NPC when cured.

2.2 Activity

To start the process of trading the player has to right-click on the NPC, which results in GUI appearing over the players screen. From there players can quit anytime, by pressing "E", "Esc" or left-click outside the GUI. In the GUI there are several slots with displayed items, which can not be stolen or dragged to the players inventory. By hovering the mouse-cursor over the mentioned items, the player can see a description of the selected trade deal. Left-clicking the desired trade deal takes away certain resources from the player (which were mentioned in the "Cost" section, while hovering over the trade) and starts the execution of gathering. After the gathering is completed, NPC returns back to where the deal was struck and the player receives their desired items. Graphical representation in Figure 2.

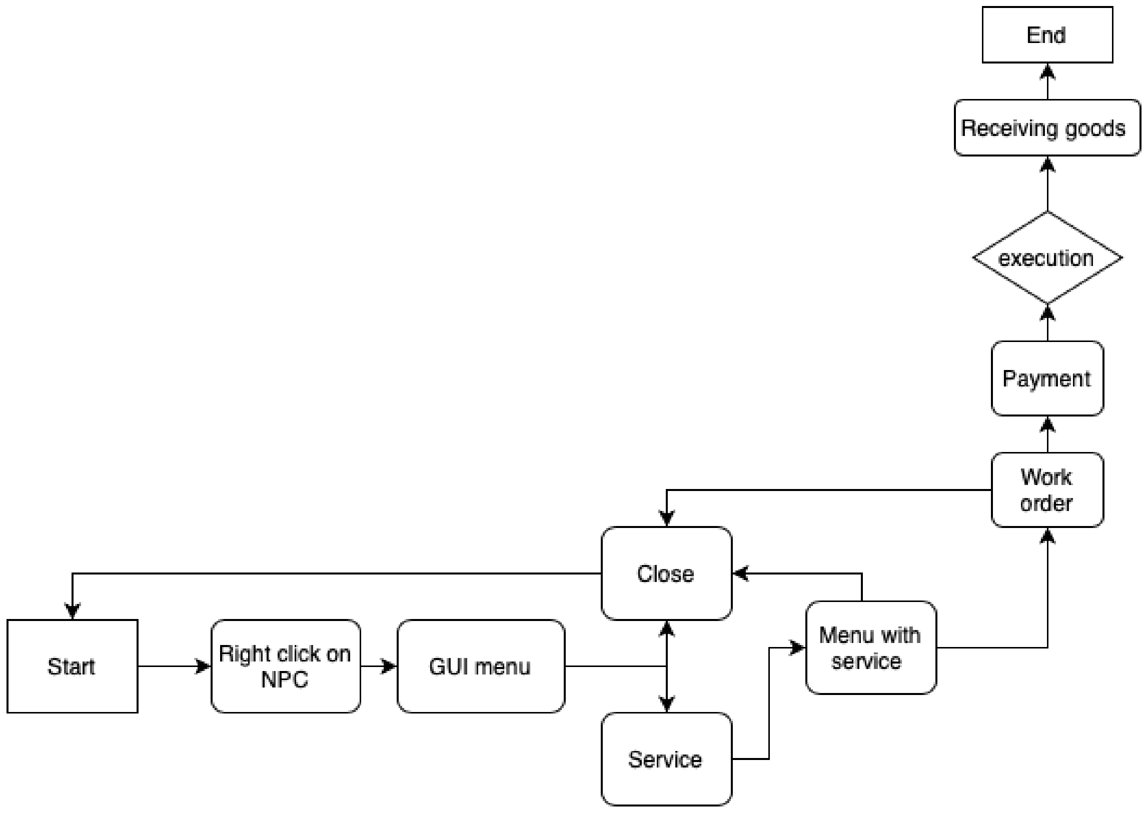

Figure 2. Activity Flowchart

3 Command Processing Algorithm

3.1 Purpose and Structure

Command processing is a key interface for Minecraft administrator to manage NPCs.
It is important to make the system modular and simple so that new commands would be quickly and easily developed/integrated.

Data is structered in a strict format, with variable flags and/or coordinates after "create" function which sends unique packets to spawn an entity [1]. NPC creation command can include no arguments after "create" function in the in-game chat, however console commands have to contain additional flag arguments which are saved in a hash-map. Examples of commands in console environment:

- npc create -n John -l x y z
- npc create -l x   z
- npc create -n John

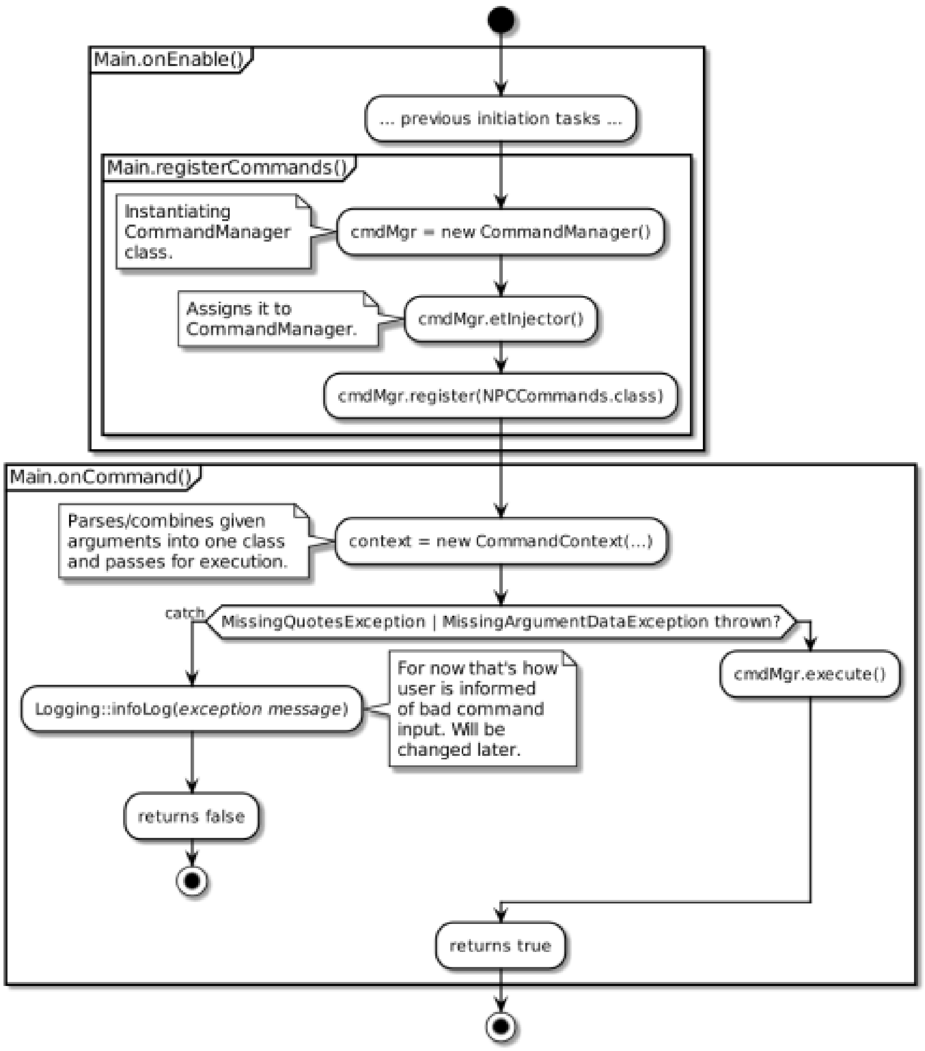
Main.onEnable()
... previous initiation tasks ...
Main.registerCommands()
Instantiating CommandManager class.
cmdMgr = new CommandManager()
Assigns it to CommandManager.
cmdMgr.etInjector()
cmdMgr.register(NPCCommands.class)
Main.onCommand()
Parses/combines given arguments into one class and passes for execution.
context = new CommandContext(...)
catch
MissingQuotesException | MissingArgumentDataException thrown?
Logging::infoLog(*exception message*)
For now that's how user is informed of bad command input. Will be changed later.
cmdMgr.execute()
returns false
returns true

Figure 3. Command Lifecycle

Data Processing: the plugin receives information that the user is trying to execute the command. The server recognizes that this is a plugin-owned command. The plugin processes the given information, explains which command, what modifier is used, what the value of the additional argument is. Then it structures the following known arguments into a list. Unknown - into separate variable [(-n : John) (-l : [x y z])].
From the processed information plugin CMDmanager tries to run the command, if it fails and the command does not exist or an error occurs, the plugin will report it. If successful, the command is called.

Real World Example:

1. Admin writes a command to create an NPC;
2. Plugin processes all arguments given to command by structuring them;
3. Plugin finds which command logic should executed;
4. Command gets executed.

3.2 Pseudo Code

```
Algorithm 1 parseArguments(Args, sender)
    IN: Args, sender
    OUT: ArgsHash

CommandFlag (CMD_ARGUMENT, NPC_NAME, NPC_LOCATION)
// each flag has ( flag, alias, cParam )
ArgsHash <- HashTable
argCount = 0

FOR i = 1 TO LENGTH OF Args
    arg = Args( i )
    flag = CF( arg )            // parse arg value and return CF type

    IF flag IS EQUAL CF.NPC_NAME THEN
        INCREMENT i
        CALL parseStringArgument( argCount, flag, i, Args, ArgsHash )
    ELSE IF flag IS EQUAL CF.NPC_LOCATION THEN
        INCREMENT i
        CALL parseLocationArguments( flag, o   , Args, sender, ArgsHash )
    ELSE // CF.CMD_ARGUMENT
        argKey = flag + i
        CALL parseStringArgument( argCount, argKey, i, Args, ArgsHash )
        INCREMENT argCount
    END IF
END FOR

RETURN ArgsHash
```

```
Algorithm 2 parseStringArgument( argCount, argKey, o  , Args, ArgsHash )
    IN: argCount, argKey, o  , Args, ArgsHash
    OUT: o  , ArgsHash

arg = Args( o   )

IF arg STARTS WITH " THEN
    IF arg ENDS WITH " THEN
        INSERT arg WITH argKey AS KEY INTO ArgsHash
    ELSE
        ArgConcat <- List
        INSERT arg INTO ArgConcat

        REPEAT
            INCREMENT o
            ASSERT o   < LEN( Args )
            arg = Args( o   )
            INSERT arg INTO ArgConcat
        UNTIL arg ENDS WITH "

        arg <- COMBINE ArgConcat INTO text

        INSERT arg WITH argKey as KEY ArgsHash
    END IF
ELSE
    ASSERT arg DOES NOT START WITH ’-’ OR ’--’
    INSERT arg WITH argKey AS KEY INTO ArgsHash
END IF

RETURN o  , ArgsHash


Algorithm 3 parseLocationArguments( flag, o  , Args, sender, ArgsHash )
    IN flag, o  , Args, sender, ArgsHash
    OUT o  , ArgsHash

xyz = List( 3 ) // list of numbers

FOR i = 0 TO flag(’cParam’)
    INCREMENT o
    val = Args( o   )

    BEGIN
        PARSE val AS number INTO xyz( i )
    EXCEPTION
        WHEN val IS NOT number
            IF val IS player('name') THEN
                xyz = player('location')
            ELSE IF val IS EQUAL '~' AND sender IS player THEN
                CASE i OF
                    0 : xyz( 0 ) = sender( 'location.x' )
                    1 : xyz( 1 ) = sender( 'location.y' )
                    2 : xyz( 2 ) = sender( 'location.z' )
                END CASE
            END IF
    END
END FOR

INSERT xyz WITH flag AS KEY INTO ArgsHash

return o  , ArgsHash
```

4 Roles and personalities

4.1 Roles

Roles are titles which define what type of resource gathering the NPC will be capable of. There are 3 different roles and every NPC will have only one, randomly assigned when it spawns (see Figure 4).

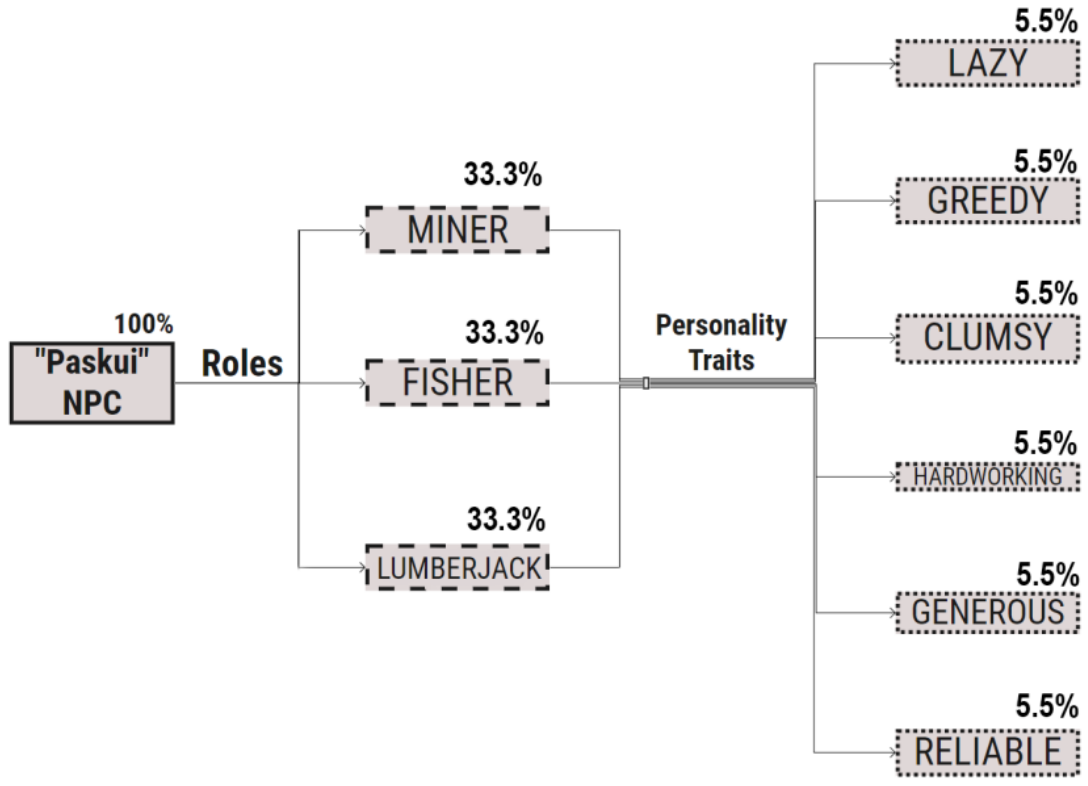

Figure 4. Role and personality graph

Spawn rate of every role is the same and the determining formula is:
NPC spawn rate when curing a "nitwit" villager (100%) / Number of roles (3) = 33.(3)%

Every role in depth:

- **Lumberjack** - gathers wood-related resources like, Oak Logs, Birch Logs, Spruce Logs, Dark Oak Logs, Acacia Logs, and Jungle Logs (amount=128). Another possible service is apple gathering. Since one of the most powerful healing items in the game are golden apples, we have decided that apples are a pretty useful resource, so the lumberjack role will be able to collect them if a player buys that service (amount=64). Lastly, this role will be able to gather various saplings (amount=16).

- **Miner** - gathers underground resources, like Cobblestone (amount=96), Coal (amount=64) and Iron ore (amount=32).

- **Fisher** - catches resources from the water, like Raw Cod, Raw Salmon, Tropical Fish, Puffer Fish (amount=64). Other two services however, will be a lot more fun. To try and make the trading process more unique and exciting, we will add a lottery of a sort to the fisher role. When selecting fishing for items or expedition services, the player can either be rewarded greatly, or get an absolutely useless item. For example a player buys the expedition service for 10 gold. Then, when the NPC comes back, the player either gets a name tag (good item) or a leather shoe (junk item).

However, when the resource is not reachable by the NPC (no such resource found or no path found), it will leave the area in a random direction and after acquiring some distance from the player it will despawn (disappear). After despawning a timer emulating gathering will start and after the dedicated time had passed, the same villager will respawn (reappear) and proceed with the transaction as usual. This system is in place to deter NPCs from burdening the player and making the whole system itself useless if an unaccounted situation occurs.

4.2 Personalities

Personalities are permanent modifiers to every NPC, which define certain service buffs (improvements) or debuffs (deterioration). There are 6 different personality traits and every NPC will have only one, randomly assigned when it spawns (see Figure 4).

Spawn rate of every personality trait is the same and the determining formula is:
NPC spawn rate when curing a "nitwit" villager (100%) / Number of roles (3) = 33.(3)% / Number of personality traits (6) = 5.(5)%

Every personality trait in depth:

- **Lazy** - takes a random amount of time longer to deliver your order (<240s) (without Lazy or Hardworking trait the base is 500s). Every transaction resets the previous debuff and calculates anew.
- **Greedy** - raises the price for services a random amount (>1x <2x). Every transaction resets the previous debuff and calculates anew.
- **Clumsy** - randomly raises the chance of failure for the NPC (<15%) (without Clumsy or Reliable trait the base is 5%). Every transaction resets the previous debuff and calculates anew.
- **Hardworking** - takes a random amount of time shorter to deliver your order (<240s) (without Lazy or Hardworking trait the base is 500s). Every transaction resets the previous debuff and calculates anew.
- **Generous** - lowers the price for services a random amount (>0.5x <1x). Every transaction resets the previous debuff and calculates anew.
- **Reliable** - randomly lowers the chance of failure for the NPC (<=5%) (without Clumsy or Reliable trait the base is 5%). Every transaction resets the previous debuff and calculates anew.

5 Requirements

5.1 Functional

NPC Roles:

- NPCs have different roles and personalities.
- One NPC can only have one role and personality.
- The personalities are generated randomly via RNG (Random Number Generator).
- The personality will be specified in the GUI (Graphical User Interface).
- The role will be specified above the NPC.
- Different personalities must have different outcomes for the trade process.
- Different roles must do different jobs.

Must Have Features:

- The plugin must run on these operating systems: Windows, Linux, macOS.
- The console user must use the -l flag when executing npc spawn command.
- The player must not be able to take out the items out of the GUI (Graphical User Interface).
- The NPC should be reloaded back into the server after a server restart.
- The NPC must be deleted from the data file after remove commands/after death.
- The NPC can be killed by other mobs/players.
- NPCs must have different roles and personalities which would affect the trading system.
- The plugin must work on Minecraft: Java Edition version 1.17.1.
- The NPC must simulate resource gathering while it is working.
- The NPC must be invincible while a player interacts with it.

Installation:

- Installation is really simple and quick.
- It requires only simple computer knowledge.
- No additional software needed in order to install the plugin.
- The installation does not require any permissions.
- To check if the plugin was installed look at the servers terminal.

Spawning:

- The NPC must spawn when a player executes /npc create command.
- The NPC can also be spawned via the console using the /npc create -l command.
- The NPC can have a custom name with a /npc create -n command.
- The NPC can be spawned with a specific role or personality while using the -r and -p flags.
- The NPC can be removed with the /npc remove command.
- To remove a certain NPC it is possible to use the /npc remove id command.
- To remove all the NPCs it is possible to use the /npc removeAll command.

Interaction:

- Players can interact with the NPC by right clicking it.
- A simple to use inventory-like menu.
- Pick a service by pressing on it.
- Receive the goods into the inventory.
- NPC must stop while a player is interacting with it.
- NPC must always have the same service menu.
- A "help" option with basic information is available at the interaction menu.

Gathering:

- The gathering is done by the NPC itself.
- The NPC uses "Minecraft" built-in navigation functions to walk to the required material.
- The NPC uses "Minecraft" destroy block mechanics to destroy that material.
- The NPC has a certain timer (depends on the role) to finish the gathering.
- Invisibility is used to simulate going really far away from the player.
- If no specific material is found nearby, the NPC goes further and becomes invisible.
- Collision is turned off for the NPC in order to not get pushed around while working.
- The NPC takes no damage while it is gathering (To make it more fair).
- If the NPC gets stuck in water it should teleport back when the timer runs out.
- The NPC takes the material to the player and ends the transaction.

Process:

- Very easy to use, does not require any confusing actions.
- The prices of the services are shown when a user hovers on the service.
- After paying all the user needs to do is just wait for it to come back.
- Once the NPC is back, the player receives the resources and the transaction ends.
- If the NPC loses the items during the process, the player won't get a refund.

Uninstallation:

- As simple as the installation part.
- All the user needs to do it just delete the plugin file.
- To avoid any unnecessary bugs or glitched NPCs it is also recommended to delete the data.yml file as well as the entities files located in world -> entities directory.

5.2 Non-Functional

Availability:

- The plugin should be available for every user who connects to the server.
- The plugin should not require any other additional software or file.

Maintainability:

- The plugin should be maintained by the developers or other contributors who decide to work on the project as well.
- The release version should be changed once updated.

Usability:

- The plugin should satisfy a maximum number of players.
- The plugin is free-to-use for everyone.
- The plugin should be easy to use for all age gaps.
- The plugin should not have any confusing commands or parts.

Performance:

- Relatively weak servers (1 GB of RAM) should have no problem dealing with the plugin.

- 300 (non-working) NPCs resulted in 20 TPS (which is just perfect for a "Minecraft" server).

Compatibility:

- Minecraft: Java Edition (version 1.17.1) game client.
- PaperMC (version 1.17.1) server client.
- Java 16 (or OpenJRE 16, or OpenJDK 16).
- Gradle (version 7.1.1 or higher).

Security:

- The plugin does not collect any information of it's user.

6 Command List

Help: **/npc help**
This command provides a set of instructions for players, guiding them to spawn their own NPC. Can be used by non-administrators. NPC creation:

- **/npc create**
  This command creates an NPC with a random role, random personality and random name at the players location.
- **/npc create -n (name) -l (location)**
  This command creates an NPC with a random role, random personality and specified name at the specified location.
- **/npc create -r (role lowercase) -p (personality lowercase)**
  This command creates an NPC with a specified role, specified personality and random name at the players location.

NPC deletion:

- **/npc remove**
  This command removes the latest spawned NPC.
- **/npc removeAll**
  This command removes all NPC's.

7 Testing

7.1 Functional Testing

System Testing

The testing of the whole plugin at a time is done manually. Only the server itself tests the plugin Compatibility (if the plugin is not compatible with the server, for example, the versions do not match, an error message is shown and the plugin simply will not start). The command testing is done by using the correct and incorrect syntax of the commands. The NPC interaction is tested out by trying to click on it by several player at a time or trying to steal the items from the GUI (Graphical User Interface) or even trying to put other items into it. As for data saving, a check is done manually every time an NPC is created or removed to see if the data is written or deleted. Same goes for roles, personalities, loot tables and so on. For this testing the testers use the "Gradle" RunServer task to launch the server with an already set plugin and test it out [2]. After the testing is finished, the tester provides feedback and points out the objects that are working correctly and those that need fixing.

Unit Testing

There were no proper implementation of unit testing for "Paper". The only fully functional unit testing method for "Bukkit" was called "MockBukkit". It provides mock implementation of "CraftBukkit" that can be completely controlled from a unit test. However, after trying to implement it, we realised that it is not compatible with "Paper", because "Paper" uses other external libraries, which do not cooperate with "Bukkit". Thus, only certain parts of the project can be tested by using unit testing [6] (those parts who do not require Bukkit/Spigot/Paper APIs). For example, command flag testing, loot table randomizing and maybe others.

Command Flag Detection Unit Test

Flags are a significant part of the plugin command management, they provide an additional argument to the commands and help the player to have more control over them. For the default case, when a player does not provide additional flags, a sample name is used as a mock argument to check if the default flag works. For the case when there are more command flags, a stream of flags and strings is created in order to simulate different command flags like "-n", "—name" and "-l". The result of the test is a notification from "Gradle" that the test was passed or failed for some reason.

Loot Table Randomization Unit Test

Loot table randomization is used in order to have a variety of treasures that the NPC could bring during one of their tasks. For this test a random number provider function gives a random integer from the length of the fisher loot table values (due to the reason that only the fisher type has loot tables for now). Then the test asserts a loot from that table which will be the one from the selected random number. The result of the test is a notification from "Gradle" that the test was passed or failed for some reason.

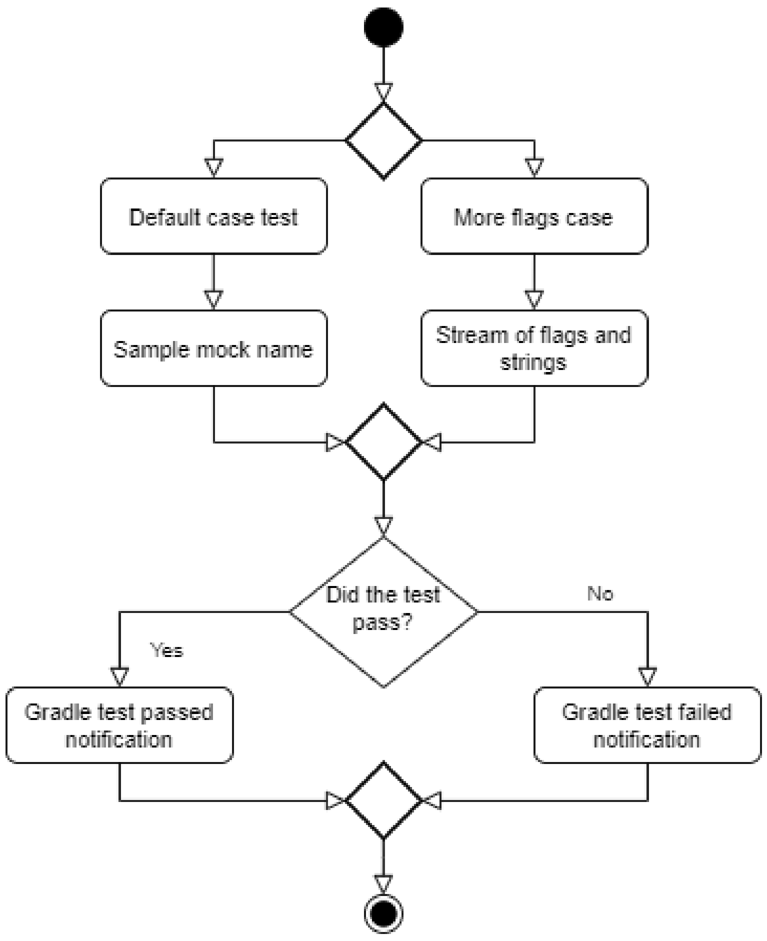
Default case test
More flags case
Sample mock name
Stream of flags and strings
Did the test pass?
Yes
No
Gradle test passed notification
Gradle test failed notification

Figure 5. Command Flag Diagram

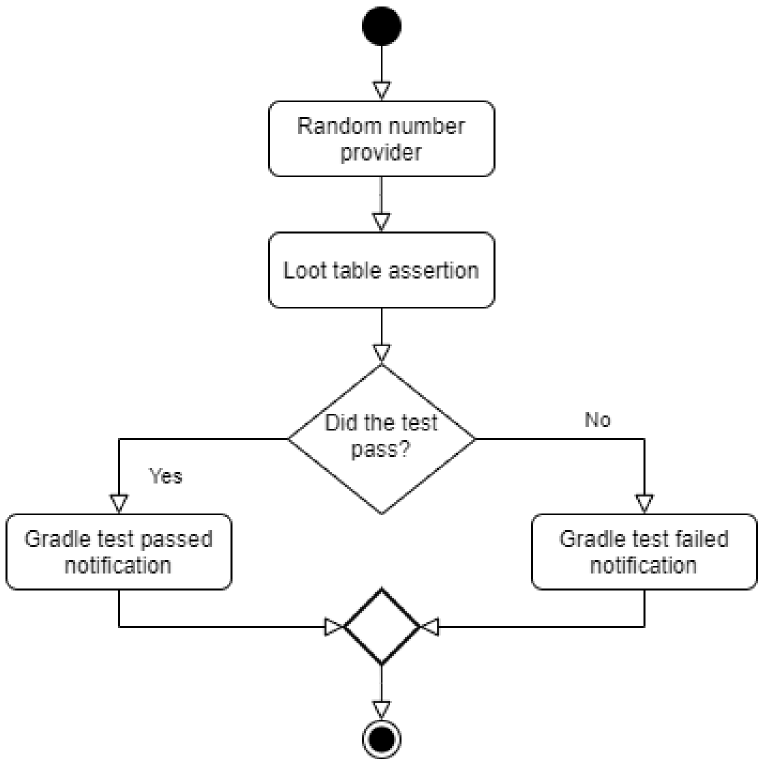

Figure 6. Loot Table Diagram

Acceptance Testing

The acceptance testing is designed to be identical, or as close as possible, to the anticipated production environment. In our case, the product itself is deeply engrained into an already existing application, which makes designing identical environment for formal showcase nearly impossible. This plugin is simple to review and test within a server as it is, so we will not be conducting acceptance testing outside of the application.

Integration Testing

We will be conducting integration testing via the logging system of our plugin. Following calls between modules, the application will be sending automatic logs to the server console and the logs directory, confirming that there are not any errors between them. Every server admin can run the server and follow the progress of integration testing to find possible malfunctions and know exactly which module contact caused it.

7.2 Non-Functional Testing

Performance Testing

This test has been conducted to measure how a system behaves under an increasing load of additional active NPCs. For this test we used a relatively weak, compared to industry standards, server of 1 GB RAM memory cap, with our latest "Woke Village" plugin installed. We have spawned NPCs one-by-one using the "/npc create" command and checked the servers TPS (Ticks Per Second), which is the main benchmarking tool used by Minecraft plugin/mod developers. The maximum TPS for a Minecraft server is 20 and without our NPCs the server was running at full capacity (20 TPS). We began spawning NPCs in increments of 5's, then 10's, then 25's, 50's and finally 100 reaching the total of 300 active, functional NPCs. During the test, servers TPS never went down below 20, which displays the scalability abilities of our plugin.

Usability Testing

First of all we have profiled the target audience of our product. After researching, we have found out that Minecraft has a very wide variety of players, from very young children to elderly players, no particular gender was dominant too, so the target audience was quite large. We have selected a group of five teenagers/adults with a wide spectrum of knowledge about the game, from beginners to long-time players and presented them with our plugin. The only thing that was instructed is how to spawn in the villagers and after that, the usability tests have been conducted.
1. Inventory accessing with right-click was a well known action to all the testers, no problems occurred.
2. After the testers have hovered over all the items in the menu, the majority clicked on the "Help" tab and proceeded to read about the functionality of the plugin, remaining testers clicked on random buttons before realizing that you need certain resources to activate tasks, after which they too have pressed the help tab.

3. After reading a short text printed out in in-game chat provided by the "Help" tab, testers have given themselves gold (currency) and proceeded to successfully make their trade deals with the NPCs.
The test was a success and testers were unanimous about the clear usability of the project. Colors were great indicators of importance, every button had title and description and most importantly, the "Help" tab could be clicked anytime the player would forget the functionality.

Security Testing

Various confidentiality, integrity, authentication and availability checks and guarantees are provided by "Minecraft: Java Edition". Our plugin "Woke Village" individually does not collect any sensitive data and cannot be altered or manipulated in-game. This plugin only holds data about NPCs: their id, role and personality, which is not related to any particular player, which ensures user safety and anonymity.

Compatibility Testing

The compatibility part of testing is conducted to find out if the product is fully functional on specified operating systems, hardware platforms, mobile devices and other designed third-party programs. Fortunately compatibility is already ensured via "Minecraft: Java Edition" itself. Plugins use the same materials/items/sprites and functionalities as the base game itself, only minimally altered to suit the purpose of developers, so it is not possible to create a non-compatible plugin for a platform in which "Minecraft: Java Edition" is compatible. All the compatibility info can be found at https://help.minecraft.net/hc/en-us/articles/360034753992-Different-Minecraft-Editions.

7.3 Case Testing

Random NPC Role generation

An administrator of the server should be able to spawn multiple types of NPC roles with one command.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The player typing the "/npc create" command is administrator of the server.

Test steps:
1. Press "T" to open in-game chat
2. Type "**/npc create**" command
3. Open their inventory to check their role, based on their available tasks

Expected result:
After performing this test multiple times, the player should encounter different NPC roles, however randomization can cause the same role to spawn multiple times in a row.

Random NPC Personality generation

An administrator of the server should be able to spawn multiple types of NPC personalities with one command.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The player typing the "/npc create" command is administrator of the server.

Test steps:
1. Press "T" to open in-game chat
2. Type "**/npc create**" command
3. Open their inventory to check their personality, based on the text above the NPC trading inventory.

Expected result:
After performing this test multiple times, the player should encounter different NPC personalities, however randomization can cause the same personality to spawn multiple times in a row.

Functional "Lazy", "Hardworking" personalities

The NPC must be faster at gathering, when the NPCs personality is "Hardworking" and slower at gathering, when the NPCs personality is "Lazy".
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: Assumption: The player has access to an NPC with "Lazy", "Hardworking" personalities, has enough resources to pay the NPC for the transaction.

Test steps:
1. Check if the NPC has "Lazy" or "Hardworking" personality
2. Perform a successful transaction to the NPC via trading menu services
3. Set a timer when transaction is complete
4. Stop the timer when NPC comes back with gathered goods

Expected result:
If the NPC has "Lazy" personality, it will come back within 750s-1000s.
If the NPC has "Hardworking" personality, it will come back within 250s-500s.

Functional "Generous", "Greedy" personalities

The NPC must have cheaper prices, when the NPCs personality is "Generous" and more costly prices, when the NPCs personality is "Greedy".
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The player has access to an NPC with "Generous", "Greedy" personalities, has enough resources to pay the NPC for the transaction.

Test steps:
1. Check if the NPC has "Generous" or "Greedy" personality

2. Perform a successful transaction to the NPC via trading menu services
3. Check how much gold has been deducted from your inventory

Expected result:
If the NPC has "Greedy" personality, it will take up to 200% of the price from the player.
If the NPC has "Generous" personality, it will take down to 50% of the price from the player.

Functional “Clumsy”, “Reliable” personalities

The NPC must never fail to deliver its resources, when the NPCs personality is "Reliable" and have 15% chance to fail at gathering, when the NPCs personality is "Clumsy".
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The player has access to an NPC with "Clumsy", "Reliable" personalities, has enough resources to pay the NPC for the transaction.

Test steps:
1. Check if the NPC has "Clumsy" or "Reliable" personality
2. Perform a successful transaction to the NPC via trading menu services

Expected result:
Repeat the test multiple times, to increase data sample-size.
If the NPC has "Reliable" personality, it will never fail to deliver the goods.
If the NPC has "Clumsy" personality, it will fail 15% of the time (up from 5% base chance).

Commands – Remove Command

A server administrator should be able to use the NPC remove command.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: A server administrator should be able to use the NPC remove command.

Test steps:
1. Press "T" to open the games chat window.
2. Type in the command "**/npc remove**".
3. Press "Enter" to submit the command.

Expected result:
The command gets executed and the latest spawned NPC is removed.

Commands – Create Command

A server administrator should be able to use the NPC remove command.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The command sender has administrator privileges.

Test steps:
1. Press "T" to open the games chat window.
2. Type in the command "**/npc create**".
3. Press "Enter" to submit the command

Expected result:
The command gets registered and the NPC is spawned near the command sender.

Commands – Remove All Command

A server administrator should be able to use the NPC remove all command to remove all the NPC from the world.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The command sender has administrator privileges and at least one NPC is present in the world.

Test steps:
1. Press "T" to open the games chat window.
2. Type in the command "**/npc removeAll**".
3. Press "Enter" to submit the command.

Expected result:
The command gets executed and all the NPC are removed from the world.

Commands – Command Flags

A server administrator and the server console user should be able to use command flags to pass additional arguments when spawning the NPC.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The command sender has administrator privileges or is a console user, and at least one NPC is present in the world.

Test steps
For player instance:
1. Press "T" to open the games chat window.
2. Type in the command "**/npc create -n (name) -l (location)**".
3. Press "Enter" to submit the command.

For console instance:
1. Open up the console.
2. Type in the command "**/npc create -n (name) -l (location)**". (Console must use -l flag).
3. Press "Enter" to submit the command.

Expected result:
The command gets executed and the NPC is spawned with the given name or/and at the given

location.

Data Management – Data Saving

The NPC data should be written into a document on NPC spawn.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: There is at least one administrator who can execute the commands.

Test steps:
1. Execute NPC create command.
2. Check if the NPC was spawned.
3. See if there is data in the data.yml file.

Expected result:
The NPC data (name, location, UUID, role, personality) is shown in the data.yml file.

Data Management – Data Loading

The NPC data should help to reload the NPC back into the server.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: The data.yml file is not empty.

Test steps:
1. Restart the server.
2. Log in to the server.
3. Check if all the NPCs are present.

Expected result:
After server restart, the NPCs should be present in the server.

Data Management – Data Deleting

The NPC data should be deleted once the NPC is removed.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: There is at least one administrator who can execute the commands, the data file is not empty.

Test steps:
1. Execute the NPC remove or remove all commands.
2. Check if NPC/NPCs were removed.
3. Look at the data.yml to see if the data was removed.

Expected result:
The NPC data (name, location, UUID, role, personality) is removed from the data.yml file.

Opening the trading inventory

A standard player should be able to open the NPCs trading inventory.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: "Woke Village" plugin NPC is standing right in front of the player.

Test steps:
1. Place in-game cursor within the hit-box of the trading NPC
2. Right-click your mouse, while the cursor is on the NPC

Expected result:
A menu pops out, displaying
1. "Help" button in the first trading slot.
2. Three items representing trading tasks are in the next three slots.
3. "Close" button in the last trading slot.

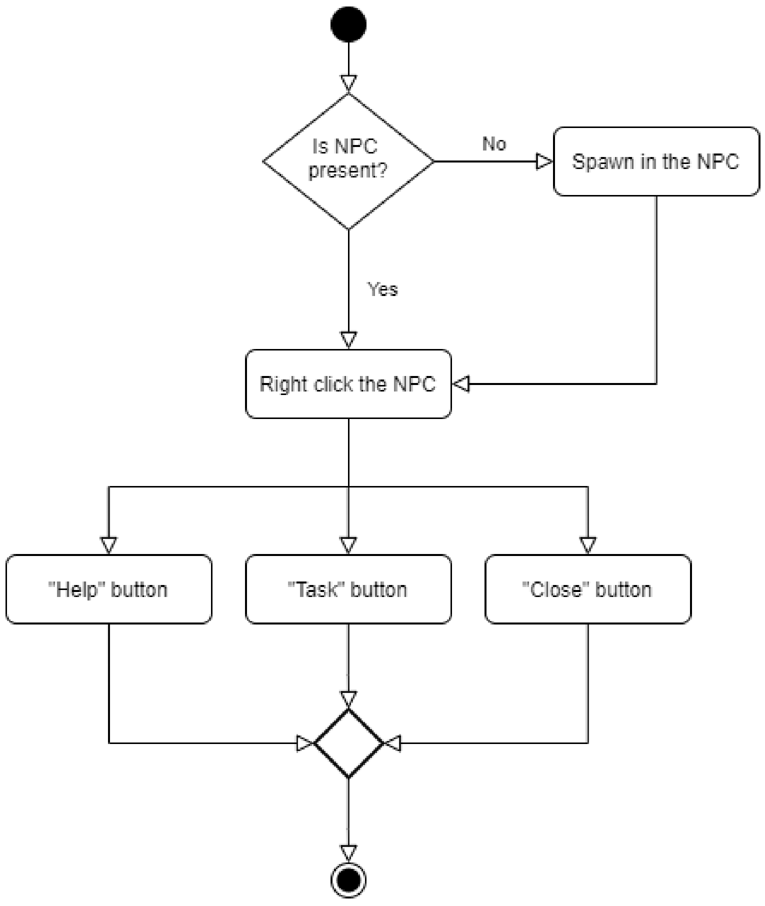

Figure 7. Inventory Accessing

Successful transaction

A standard player should be able to successfully complete a transaction with the NPC.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: "Woke Village" plugin NPCs trading inventory can be opened, the player has enough

resources to purchase traders goods.

Test steps:
1. Search for desired role of the NPC (ex.: for wood gathering, search for a "Lumberjack")
2. Select one of three available tasks

Expected result
1. A musical note will be played indicating the successful pay of the deal.
2. In-game chat displays a message: "You have bought villagers services!".
3. NPC starts pathfinding to desired resource.

Immovable menu items

A player should not be able to temper (remove, duplicate, change) with trading menus items.
Precondition: The server, in which the player is connected to has the "Woke Village" plugin.
Assumption: "Woke Village" plugin NPCs trading inventory is opened.

Test steps:
1. Drag the menus item away
2. Shift-click the menu item
3. Put items in the trading menu
4. Switch menus items places

Expected result:
Items in the trading menu will not disappear/change/duplicate. They will remain in their starting places and the player will not be able to drag them to its inventory.

Conclusions and Recommendations

Our development team saw many problems with the current "vanilla" Minecraft trade system and this project seeks to fill in all the functionality gaps and enhance the overall player experience, regarding both the addition of NPCs, which bring the concept of personality into the game and addition of a much better trading system, which will be useful to any player regardless of their intentions during the playthrough.
This work also may be extended further into personalizing each NPC and giving them the ability to build, interact between themselves and prosper if their trade is going well.

References

[1] *Minecraft Protocol*, 2021.

[2] Tim Berglund and Matthew McCullough. *Building and testing with Gradle*. " O'Reilly Media, Inc.", 2011.

[3] Joshua Bloch. *Effective java*. Addison-Wesley Professional, 2008.

[4] Hans J Kohler, Ulrich Nickel, Jörg Niere, and Albert Zundorf. Integrating uml diagrams for production control systems. In *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium*, pages 241--251. IEEE, 2000.

[5] Maddy Miller Shane Freeder, Daniel Ennis. *PaperMC Documentation*, 2021.

[6] Laurie Williams, Gunnar Kudrjavets, and Nachiappan Nagappan. On the effectiveness of unit test automation at microsoft. In *2009 20th International Symposium on Software Reliability Engineering*, pages 81--89. IEEE, 2009.