**Sergeyev K.G.**

**Technical documentation for the program**

**"Escape from tarkov item editor"**

**Moscow 2022**

**Contents**

1. Introduction 3

2. the Terms of Reference for the EFT editor 4

3. The structural diagram of the software 9

3.1 General structural diagram of the application 9

4 A description of how the modules work 10

4.1 File Integrity Check Module 10

4.1.1 General description of the module 10

4.1.2 Linkage scheme 11

4.2 Settings module 12

4.2.1 General description of the module 12

4.2.2 Linkage scheme 14

4.3 Panel module for the editor's work and adding

items to the preset 15

4.3.1 General description of the module 15

4.3.2 Linkage scheme 18

4.4 Module for selecting an item by search groups and adding

to preset 19

4.4.1 General description of the module 19

4.4.2 Linkage scheme 21

4.5 Module for editing items and global settings 22

4.5.1 General description of the module 22

4.5.2 Linkage scheme 23

4.6 Preset module for saving, modifying,

deletions and downloads 24

4.6.1 General description of the module 24

4.6.2 Schematic diagram of the preset module and preset load module 28

**1. Introduction**

The SPT-AKI Escape from tarkov (EFT) subject editor was created in order to understand how effectively the acquired knowledge of the Java language can be implemented to write a working, usable program. Since the program was created during 3 months based on the open-source code (+ 1.5 months for reading Herbert Shield's book and recommendations of fullstackoverflow seniors), most of its functions are test and in version v.1 will be finalized and partially rewritten.

Also, in addition to the program were applied fully automatic systems of analysis of the catalog of items and groups of characteristics in the game documents (items.json and globals.json) to record by groups in a separate object. Due to their efficiency and automated type, they will not be attached to the editor in this version.

Program design, fronted and backend development, as well as testing functions, I performed 90% myself, as well as understanding the principles of work, because ready-made solutions either were not effective in view of the common methods, or took up a lot of space in the code and processed unnecessary data.

There were several modules in the development plan to accomplish the tasks:

1. Settings module
2. Module for selecting, changing, saving items and global settings
3. Module for creating presets of item characteristics based on user-added items
4. Module for loading and unloading presets, with the ability to view item positions
5. Module for reading changes made by the program, to view them and roll them back if necessary

And all except module 5 are written and working in test mode, because the code is written to test classes and methods available in JAVA 9.

Version v.1 will add multithreading, serialization, rewrite code for more efficient, and will add the ability to use the program and created presets on different versions of the game.

Java 9 for the backend and Java FX + Scene builder for the frontend were used for development.

**2. the Terms of Reference for the EFT editor**

1.Introduction

The name of the development: "Escape from tarkov item editor" (hereinafter "EFT editor").

The EFT Editor is not part of the EFT game and Battlestate Games development, but is free third-party licensed open source software that allows you to edit open game files.

2. Purpose of development

The EFT editor in version v.0 is designed for 2 tasks.

The first task of the program is to edit the characteristics of items in the SPT-AKI game catalog, with the additional possibility to influence a limited list of global settings (character stamina, number of HP, experience, etc.).

The second task is to read the selected characteristics of objects into a separate list (hereinafter presets), in order to replace the characteristics in the file with the data from the preset, while the program will save the previous values for recovery from the copy. The program also allows you to transfer the saved characteristic presets between users.

3. SOFTWARE REQUIREMENTS

3.1 Requirements for functional characteristics

3.1.1 Requirements for the tool environment

3.1.1.1 The EFT editor application is designed for stand-alone operation under Windows

3.1.1.2 The subject editor is developed in Java 9 (backend) and Scene builder 8.5.0 + Java FX (fronted)

3.1.2 Requirements for the user interface

3.1.2.1 The application works as a separate process with its own code implemented through the Java compiler integrated into Windows, also using additional features of Windows.

3.1.2.2 The application functions in a graphical resolution of not

lower than 1024x768 pixels.

3.1.2.3. The application window size is 600x400 pixels.

3.1.2.4 The implementation of color identification of button presses must be implemented by means of ready solutions of the Java FX language and CSS style.

3.1.2.4.1 Unpressed buttons should have silver color and black text

3.1.2.4.2 Pressed buttons should have the same colors, but with 50% less transparency and should be blocked until the code is executed

3.1.2.4.3 The last button pressed must have a blue border reset by pressing another button

3.1.2.5 Commands to execute code must come from pressing buttons in the application, except for checks during program initialization

3.1.3 Requirements for the implemented functions

3.1.3.1 Requirements for the subject change function

3.1.3.1.1 The application after initialization should open the menu of subject selection for search with refinement of search groups; all choices are additionally equipped with the function of similar elements selection, which is enabled by typing into the field of symbols from the keyboard

3.1.3.1.2 Search groups must contain a global item type, subtype and tags for searching the item

3.1.3.1.3 Selecting an item should be implemented by selecting already selected items in the buffer

3.1.3.1.4 The found characteristics of the item must be recorded in a separate list with the possibility of changing, the field change of one characteristic is equipped with a module of value search and replacement

3.1.3.1.5 In addition to the list that changes the characteristics of the item, you must make a module that will allow you to save changes to the selected change file, replace the game file with an editable file, save changes to the buffer to undo changes made, undo changes in the line, undo all changes in the list of editing the item

3.1.3.2 Requirements for preset functions

3.1.3.2.1 The preset module must record a preset based on the selected characteristics, be able to modify already recorded presets and load, unload and delete available presets with the ability to view them

3.1.3.2.2 To start recording a preset, there should be a button to create a preset in the item selection window, with the selection of the add mode

3.1.3.2.2 Read characteristics comes from the loaded list of characteristics from the main list of item characteristics

3.1.3.2.3 The demonstration of already recorded characteristics is done with the list of items in the presets module, it must have modes of displaying items or types of characteristics

3.1.3.2.4 The items to be added are in order relative to the selection groups, and their characteristics are in the order of the list of loaded characteristics

3.1.3.2.5 When working with the preset, it is possible to remove characteristics and items from the list of added items

3.1.3.2.6. When opening the presets loading window in the game files, the function of showing already loaded presets is implemented

3.1.3.2.7 When loading presets into the game the replaceable characteristic is written to be rolled back during unloading, the file in which this is written is a copy of the preset except for the characteristic

3.1.3.3 Requirements for the setting function

3.1.3.3.1 The application language change with a language-specific flag is implemented in the settings

3.1.3.3.2 A choice of enabling/disabling the animation of the sliding panels of the application is implemented

3.1.3.3.3. The selection of file directories, the game and the program itself has been implemented

3.1.3.3.4 Implemented selecting, creating and deleting a copy of a clean game file for editing

3.1.3.3.5 The option to restore the selected original game file from the available

3.1.3.4 Requirements for application libraries

3.1.3.4.1 Develop a library of universal methods for application modules

3.1.3.4.2. Develop a library of items for the selected game version

3.1.3.5 Requirements for pop-up errors and warnings

3.1.3.5.1 Errors and warnings must be in the selected language

3.1.3.6 Application initialization requirements

3.1.3.6.1 When initializing different modules, implement a check for files and settings for reading, in case of errors there is a recovery of damaged files

3.2 Reliability requirements

3.2.1 The program must be protected against repeated button presses during code execution

3.3 Requirements for the composition and parameters of technical means

3.3.1 The application is intended to run on Windows 7 and later

3.3.2 Minimum system requirements intel core I5 processor or AMD equivalent.

3.3.3 Requirements for other components correspond to the requirements of Windows

3.3.4 The program must occupy no more than 350 MB of RAM

3.4 Requirements for information and software

compatibility

3.4.1 The application must be fully functional on Windows 7 and above, and work with any kind of computer hardware suitable for the system requirements of the application.

3.4.2 The EFT v.0 editor is intended for a certain version of the game and full functionality on other versions is not guaranteed

3.4.3. The editor can only use the open game files items. json and globals.json, and cannot read closed files

3.5 Special requirements

There are no special requirements.

4. TECHNICAL AND ECONOMIC INDICATORS

The requirements for technical and economic indicators are not

are.

5. STAGES AND PHASES OF DEVELOPMENT

1) Development of the draft version of the application.

2) Testing of implemented functions, addition of universal methods

3) Development of a module of errors and warnings when the code works

4) Developing a language change module

5) Testing and debugging a stable offline version of the application

**3. The structural diagram of the software**

3.1 General flowchart of the application

Launching an application

Initial selection of the game directory and automatic setting

Checking the integrity of the settings

Preset module for saving, changing, deleting and loading

Panel module for working with editable objects and presets

Item editing and global settings module

Module for selecting an item by search groups and adding it to a preset

Launching the main block of item editing modules

Pressing the window change buttons in the main menu

Restoring directory paths and settings

Launching the settings module

Opening the "About" window

Selecting the language, paths to directories, working with editable files

Saving settings

Universal modules for working with objects and presets

Integrated module for working with a file containing objects and readobjectarray groups

Module containing gsErrs errors

Common application modules

Integrated writemodules module with universal code

**4 Description of the modules operation**

4.1 File Integrity Checker

4.1.1 General description of the module

The Integrity Check module is designed to create paths to the application and game directories, creating a primary settings file. The validation module implements the following functions:

* Checking and restoring paths to application folders
* Checking and restoring the settings file
* Initial selection of the main directory of the game and setting paths in the settings file

Checking and restoring the paths to the application's folders is done every time the application starts up by calling chkPath(), this is an automatic system implemented on a simple check, ! file. exists() > file. mkdir().

The settings file is restored if the request file. canread() request comes false, for this purpose Wm. writeSettings() is started, after which the settings.txt file is created in the application folder and then the prewritten lines are written.

For the initial generation of settings it is necessary to select the directory of the game, to optimize the selection of paths code, was inserted in the initialization of the main unit of the modules editing items and presets, if in the settings there is no path to the items. json from the game directory runs Wm. findGameLocation(), universal code that allows you to select the directory, when selecting the directory it checks if the folder of the game is specified, if it is specified correctly, it automatically determines the type of directories (there were different layouts depending on the game and different paths) and puts all the paths, otherwise an error window will be displayed indicating the correct directory of the game.

4.1.2 Linkage scheme

Directory Existence Error

Launching an application

Recovering directory and file paths

Checking directory and file paths

Check the settings in the Settings. txt file

Read settings error or no file

Restoring the settings in the file

File or directory is missing

Opening the main application window, starting work

Selecting the directory of the game, setting all paths to directories

Checking items. json from the game directory

4.2 Settings module

4.2.1 General description of the module

Settings module is designed to change the global functions in the application, as well as designed to work with the game files. At the moment, settings implement the following functions:

* Changing the application language
* Changing the modes of pop-up panels
* Changing directory paths
* Restore the selected game file with parameters to the original
* Deleting, creating and selecting a file for editing characteristics and then loading them into the game

The settings module is implemented as a group of subsystems that implement global changes in functions, or work with the main application files. To optimize any change of settings is implemented through the application of changes by pressing the OK button, followed by checking the two settings buffers for changes to settings. If there are changes in the settings, then there is an analysis of changes and overwrites the settings file. In all main modules for this is responsible gStngBufr and gStngBufr1.

To change the language the subsystem of language change is implemented, it is written in the initialization of the main modules and reads the selected language from the settings file. In the settings window it looks like combobox [languageChangerChoiceBox] with language description and country flag [languageFlagImageView]. In order to change the language it is necessary to select one of the available languages in the combobox, and in case of changing the value will change the flag and change the value in the settings buffer doubler [gStngBufr1], the working principle is based on EventHandler with activation through the trigger [languageChangerChoiceBoxmousereleased].

Changing the mode of the pop-up panels is implemented through radiobutton[panelanimationMenuAnimationRadioButton, panelanimation\*...] and a simple check of the buffer value (1 or 0), to change the mode you must press the signed button after which the value in the buffer doublet will change to 1 or 0 respectively and the button will change its visual style.

File paths are implemented via a locked textfield [pathGameDirectoryTextField, path\*TextField...] and a search button on the side of textfield[pathGameDirectorySearchBtn, path\*SearchBtn...]. Textfield will always show path to selected directory. To change the path, a universal module findLocation is implemented based on either FileChoser for the file path or DirectoryChoser for the directory path. When it is run, it receives 4 error texts in case of incorrect input and a position index in gStngBufr1 in which the new path is written.

Then there are two subsystems for working with configuration files

The first subsystem is designed to restore the original file of the game, this is necessary if you have loaded the wrong settings or the user wants to play a clean version of the game. To restore, select an item group in the combobox "Select item group" [makefileCategoryChoseItemChoiceBox] from the available and click on "Restore file" [makefileCategoryItemRestoreButton], then the confirmation window Wm. dialogWindow in which passes a warning from the list of errors gsErrs, if the answer is positive will replace the selected game file on the original, otherwise nothing happens. The same group of objects changes the choice of file to change the settings.

The second subsystem is needed to create several options files with settings items and global game settings, for example to make certain weapons and their ammunition more effective than others, make grenades completely useless and shotgun magazines more roomy (value chosen by the user), to make the xp player above 100 for example 9000, etc. To do this, implemented 3 main buttons and a few hidden.

The subsystem is implemented inside the event of the pressed "Create file" button [makefileMakeNewItemsFileBtn] with the trigger [makefileMakeNewItemsFileBtnaction], the EventHandler and the global index from the buttons "Save"(1) [makefileApplyBtn] and "Cancel"(0) [makefileCloseBtn] are used to create or cancel it. If the index is 1, the eventhandler trigger is triggered and the file creation mode is started in the directory of the specified group.

To start changing items it is necessary to select an item group [makefileCategoryChoseItemChoiceBox] and a copy of the file [makefileItemsFileChoiseBox], in case there is no file for changes in the group it is necessary to create it through the "Create File" button, on clicking it the textfield [makefileMakeNewFileTextField] opens where you must enter the file name and confirm creation through the "Save" [makefileApplyBtn] or click on "Cancel" [makefileCloseBtn]. When saving, the trigger is started and the file is saved, also the subsystem is equipped with a Wm. dialogWindow with a warning message if the file already exists, a warning window Wm. errorWindow if there is no file name and a radiobutton [makefileCopyOldFileToNewRadioBtn] to enable/disable saving all values from the already selected file copy.

If the copy of the file is no longer needed it can be removed via the "Delete" button [makefileDeleteItemsFileBtn], after which the dialog box Wm. dialogWindow with a warning, if the choice is positive the file will be deleted and you will need to select the desired copy of the file, if the choice is negative nothing will happen.

4.2.2 Linkage scheme

Reading settings in settingbuffer and settingbuffer1

Pressing the "Settings  
" button

Replaces the main window with a Settings window. fxml

Action with the file in the save/ FileChanges/Game\*/\* directory

Changing the settingbuffer1

Restore the selected game file to the source file from the backup

Changing the application language, changing the operation of pop-up panels, selection of directories and file directories

Search for files in the application directories to replace characteristics

Creating, modifying and deleting files of the selected type for replacing characteristics and loading into the game

Pressing the "OK" button and saving the settings by comparing settingbuffer and settingbuffer1

Selecting a file type and restoring a selected game file from backup storage

Getting started, the main modules of the application

Replacing the main window with the Main window. fxml

4.3 Panel module for the editor's work and adding

items to the preset

4.3.1 General description of the module

The module of panels consists of two panels in the right window, the first window [globalItemsValuePane] is designed to work with the editor, the second window [globalPresetsMakePane] is designed to work with adding lines to the preset. Their choice is realized automatically on the basis of switching when creating a preset, so as long as the preset is being created, window 2 is shown, otherwise window 1 is shown.

Window 1 is a hidden panel of type Pane in which there are 5 buttons, 5 descriptions and 1 image to show the selected item visually (so far does not work because it does not yet have a database of images). The buttons have three functions:

* Saving characteristics to a copy of the file, or replacing the game file with the selected copy of the file in the settings
* Saving modified item characteristics to the characteristics buffer
* And undo changes on the selected position, or completely undo changes from the characteristics buffer

The first function is made in two variations, the first is saving the characteristics in a copy of the game file, this is implemented in order to save changes without affecting the game file, the user could track the changes in a particular copy, replace the source at once all you need. This is implemented through the buttons "Save to backup" [mainsidepaneSaveInBckp] with the trigger [Saveinbackuponaction] and "Save to game" [mainsidepaneUpldToGame] with the trigger [Uploadtogameinaction], when you click "Save to backup" the trigger is run, after which the copy file opens and begins to read the file and write to the new, before the beginning of the object all read lines are written to the object, and when you reach the beginning line of the object the list of characteristics is replaced by the list from the editor, when you reach the end, the read lines are written to the new file, after the end of reading the old is deleted and the new changes its name.

Saving changes to the game is implemented more simple and performs a simple function, when you click "Save to Game" begins to replace the game file on a copy of the file, the name of the new file is (the name of the old file + 1), when finished the old is removed, and the new changes its name.

The second function is implemented as a string list equal in size to the loaded list of characteristics from the selected item. By pressing "Save to buffer" [mainsidepaneGlblSaveInBfr] the [GlobalSaveInBufferAction] trigger is triggered, and the buffer list of characteristics is cleared and all characteristics from the list of the item are copied. Thus, when you undo changes in the list of characteristics of the item, the values will be changed to the values of the buffer.

The third function is implemented to undo changes in the list of item characteristics, there are two options to choose from, "Restore value" [mainsidepaneRstChngs] with the [Globalresetchangesaction] trigger and "Do not register" [mainsidepaneGlblRstAll] with the [Globaldontregistrateaction] trigger. When you click on "Reset value" if a line in the item editor or global settings is selected, it will be replaced by the original line, otherwise there will be an error about no selection. "Do not register" replaces the list of item characteristics with the list of buffer, thus cancelling the changes. In the case if the editor is not opened, an error will be given out about the opening of the item editor or global settings.

Window 2 is a hidden panel of type Pane in which there are 3 buttons and 1 list of type ListView to add the selected characteristics to the preset, taking into account their local and global position.

The buttons have three functions:

* Resetting line selection in the list
* Adding selected lines to a preset
* Analyzing and adding an array of lines (object) to a preset

The line selection is reset by a simple [ClearPresetSelectionAction] trigger from the "Deselect" button [wrkwthprstClearPrstSelection]. When clicked, the [ClearPresetSelectionAction] trigger is triggered and the user is shown the Wm. dialogWindow with a selection confirmation, on triggering of which the selection in the list is cleared [wrkwthprstVluesOnPrstItms].

Adding selected lines through the "Add lines" button [wrkwthprstAddToLst] to the preset is implemented on a similar principle, but with the addition of a countdown for the selection of lines in the list, when the [AddToListAction] trigger is activated, it creates a cyclic addition of lines to the preset, taking into account their position in the list of object characteristics.

Adding an array through the "Add array" button [wrkwthprstArrToChng] differs only in the way of adding, in this method the addition of lines with the characters [ and { is implemented, in the case of selection and addition the trigger [wrkwthprstArrToChngaction] is activated which starts the cyclic addition of objects to the preset, taking into account their position. In total there are three types of objects by which the sorting, type assignment and recording is going, it is necessary for the finalized loading of preset to be possible to record the characteristics in the object (for example the number of slots in 3 pouches of military vest), which at the moment is made by 60%.

4.3.2 Linkage scheme

Enabling Preset Creation

Selecting an item to change characteristics

Copying characteristics from the item editor to the list of panel characteristics

Get the IDs of the loaded item id, parent, etc.

Search for item characteristics in the file to copy

Panel for working with presets

Filling the list of characteristics in the editor and the buffer

Panel for working with the editing module

Item editing module

Deselect lines in the panel list

List of characteristics to add to the preset

Features buffer

Items.json item editor

Restore the value of the selected characteristic

Restore the value of all characteristics

Add an array of subject lines to the preset

Button to save the list of editor's characteristics to the buffer

Add subject lines to the preset

Selected file for changing characteristics and loading

Button to save the characteristics to a file for downloading

Trigger activation in the code

Create a preset with items

Selected file for changing characteristics and loading

Saving (replacing) the game file with a download file

Trigger activation in the code

Preset creation module

Files in directories

4.4 Module for selecting an item by search groups and adding it to a preset

4.4.1 General description of the module

The module of selecting an item by search groups and adding to the preset is divided into 2 independent systems. The first system implements the selection of an item to load in the characteristics editor, the second system implements a simple method of adding an item to the preset and changing the panel to create a preset.

Implemented the selection of an item by limiting the search to specified groups and tags, for example, for a quick search for body armor you can select groups (Gear, Armor vests) and additionally select tags for example USA to search for American body armor. For selecting groups, tags and items, signed comboboxes with selection help system [\*helpselect] are implemented:

* "Type selection" [wrkwthprstChsTypeOfItm] with trigger

[Choisetypeofitemaction]

* "Subtype selection" [wrkwthprstChsScndType] with trigger

[Secondtypeaction]

* "Select tags" [wrkwthprstChsTags] with a trigger

[Choiseitemsaction]

* "Item selection" [wrkwthprstChsItm] with a trigger

[Enteritemaction]

Therefore, a universal module FindItems was implemented for the main type of items and subtypes with index 0 for main, 1 for subtype, it finds matches by groups and adds them to the dynamic array of items to be read from the main array of items. It then erases the previous item selection and picks up a new list of items in Item Selection.

There is also a complex subsystem for adding tags and working with them, the interaction with it is automatic in the case of a set of names and semi-automatic if the user manually begin to change the tags by clicking on the arrow on the side of the combobox "Tag Selection".

All modules are equipped with additional subsystem for the analysis of the entered word and the search for similar words, but they are simplified in comparison with the tagging system, that is, if you enter a word Rounds in a subtype it will find only it, or for example in the selection window to enter the caliber of cartridge 7.62x54 it will find only similar caliber cartridges. For this purpose, there is a dynamic list of readable objects from the main array of objects matching the criteria. For convenience of work with the search there is a cutting off of unnecessary values to the left and to the right of the selected value.

When you select an item, the combobox "Select item" trigger is started, it finds an item in the list of items to read using the index from the combobox, then finds a copy file and reads the characteristics of the selected item from it. Each time you select an item, the index in the combobox replaces the global item index [gPrstItmIndx] for convenience.

The read characteristics are placed in two lists. The first is the item editor list, the second is a copy of the item editor list, which is a buffer for undoing changes in the editor.

Creation of the preset is made through integration into the module of the selection, to start creation you need to click "Create preset" then change the panel for convenient adding characteristics, and the button will be replaced by a combobox with a mode of adding and a search button. To add an item to the preset, select the type of addition in combobox and press search (At the moment you can add only an item), then from the item editor is copied to the list of characteristics on the panel, for convenience characteristics can be changed. The function of changing characteristics is implemented through the overloaded method CellFactory universal factory ItemsValuesListCell. When you click to create a preset, a preset object [gTmpPrstLst] is created, where items, lines and data are added to quickly find items in items. json.

4.4.2 Linkage scheme

Select tags

Choice of type

Selecting a subtype

Matching items in a group

Setting the global item identifier

Selecting a subject

Adding identifiers to the "To work with items" panel

Adding characteristics to the Items editor. json

Copying characteristics to the restore buffer

Panel module for working with objects

Pressing the "create preset" button

Enabling the subsystem for adding groups of items to a preset

Changing the sidebar display mode to preset mode

Deselect button

List of features

Selecting a group of items to be added

The "Add line array" button

Adding an item to a preset

The "Add Line" button

Copying characteristics from the Item Editor to the "To work with presets" panel list

The second window in the item selection and preset creation module

Create a preset with items

Highlighted lines in the list

4.5 Module for editing items and global settings

4.5.1 General description of the module

To edit items and global settings a cascade of Accordion panels is implemented under the main menu, in the cascade there are two panels "Object Editor" [globalItemValues] and "Globals.json Editor" [globalGlobalsEditor], inside them there is an editable ListView list using overloaded CellFactory with universal factory ItemsValuesListCell. This factory allows you to determine where a value is located inside a list cell with a string, write it to a field and allow you to work only with the value ignoring extra data, when you press enter the whole string with the replaced value will be shown.

To edit global values, a second panel has been added in which, in addition to the implemented list, a combobox is added with a selection of a group of global values, this was done in view of the lack of explicit groups within the globals. json, and a faster selection of data compared to the selection of the subject.

Combobox of global settings subtype selection works by triggering eventhandler, in case new value is not equal to old one. The list is re-filled by reading a copy of the globals file, in this case, unlike the subject search, the data set is searched by the key strings prescribed manually during initialization in simple lists. If there is a match, then it is read and written to the editor's list and to the buffer copy of the main list, the principle of which is identical to that of the item editor.

4.5.2 Linkage scheme

Item selection module

Selecting a group of global values

Opening the globals.json editor window

Sidebar for working with the editor

Game file selected by the panel automatically

Selected application file to replace the characteristics

Replacing the game file with a characteristics file from the app

Saving editor changes to a file

Saving characteristics to a backup list

Restore characteristics from the backup list

Backup list of the selected editor

Changing the characteristics in the list of the selected editor

Selecting characteristics in the editor list and backup list

Convert text in a cell when clicked to a clean value in a line and reverse conversion when editing is finished

Opening the items.json editor window

Selecting a subject in the subject selection module

4.6 Preset module for saving, changing, deleting and loading

4.6.1 General description of the module

The presets module is implemented as two subsystems:

* Preset creation and preset modification subsystem
* Subsystem for loading a preset into the main game file

To create and modify a preset, use the second hiding window "Work with presets" in the menu tab, which contains the following

Standard listview list [wrkwthprstAddedPrstVlues] for displaying items and strings added.

The two Radiobutton "Display Items/Lines" [wrkwthprstPrstListWrkMod] and "Lines/Array" [wrkwthprstArrMd] buttons with changing list modes, they work simultaneously to adjust the list. The first button changes the display of objects/lines, and the second allows you to change the lines/line arrays. Their work is related to the preset [gTmpPrstLst] they create, from which they take certain groups of items. The mode of operation is made through a radio button trigger [PresetListWorkModeAction, ArrayModeAction] which reads the mode of operation and reads from the group through the preset path.

Four buttons for working with preset items:

1. The first button [wrkwthprstDltItm] implements item removal through the [DeleteItemAction] trigger, it works with the created preset [gTmpPrstLst], and removes the selected in the list [wrkwthprstAddedPrstVlues] item indices from it, thus completely erasing items and lines in it from the created preset [gTmpPrstLst].
2. The second button [wrkwthprstRmvLines] implements the removal of added lines in the listview by a similar principle through the [RemoveLinesAction] trigger.
3. The third button [wrkwthprstChngItmOnPrst] implements changing the item selected in the preset list by replacing the global index [gPrstItmIndx] through the [ChangeItemOnPresetAction] trigger, for quick selection of search links, after which the file search starts and re-fills the add characteristics panel [wrkwthprstVluesOnPrstItms] in the preset, this reduces the time to re-pick the item in case the user wants to add more characteristics.
4. The fourth button [wrkwthprstAddNewItm] unlocks the addition of another item through the [AddNewItemAction] trigger, through the first tab of the item selection menu. It also resets the index of the selected item to -1 so that no values can be added to the item.

And two buttons to work with the preset. The first button offers to close the preset creation [wrkwthprstCancelPrst] with the [CancelPresetAction] trigger, to clear the preset file [gTmpPrstLst] and to reset the selection index [gPrstItmIndx] to -1. The OK [wrkwthprstPrstOk] buttons with the [PresetOkAction] trigger and the Cancel [wrkwthprstPrstCancel] buttons with the [PresetCancelAction] trigger appear to confirm it. If canceled, the preset will continue to exist and you can continue to add items to it.

The second button implements the preset saving [wrkwthprstSavePrstBtn] with the [SavePresetAction] trigger, when clicked, the [saveprstPresetsPane] appears where you must write the preset name in the textfield [saveprstTextField], its description in the textarea [saveprstTextArea] and confirm [saveprstOk] or reject the saving [saveprstCancel]. If you confirm, the [SavePresetOkAction] trigger is triggered and the preset editor folder is saved, followed by the clearing of temporary variables and objects. If the [SavePresetCancelAction] is canceled, the panel hides and allows you to continue creating the preset.

Changing the preset is implemented as the previous module, but with a change in the principle of operation and adding a new panel, to change the preset press the button "Change preset" [wrkwthprstChngPrstBtn], after which the trigger [Changepresetbuttonaction] opens the panel to replace the preset [chngprstMainPane]. Then you select a preset from the combobox [chngprstComboBox] through the trigger [ChangePresetComboBoxClicked], and if you want the user writes (or leaves the previous) name [chngprstLabelPresetName] and description [chngprstDescription], then comes confirmation and work with the preset. After that you need to confirm the change with the "Overwrite preset" button [chngprstLoadBtn] with the [ChangePresetLoadAction] trigger or close the preset change with the "Cancel" button [chngprstCloseBtn] with the [ChangePresetCloseAction] trigger.

The preset loading subsystem is implemented as several complex modules:

* Subsystem for loading a preset into the game and copying the replaced settings
* Preset description and added items viewer

To start loading a preset you need to click the "Load Preset" button [wrkwthprstLoadPrstBtn], then the [Loadpresetbuttonaction] trigger is activated, and the preset loading window [loadprstGlobalAnchorPane] opens over the main window [globalGlobalpane], and the reading of the available presets list [gAllPrsts] to add to the list of available for loading [loadprstAvailablePrsts] starts, with the condition that they are not already added, after the initial download of presets goes to adding presets to the list of already downloaded [loadprstAddPrsts] through checking files in the application folder FileChangesMain, with the condition that the presets are available for download. The load window that opens has a horizontal split for loading presets and viewing items, the line is not visually displayed. The upper part displays the list of available for download presets [loadprstAvailablePrsts] and the list of already downloaded presets [loadprstAddPrsts].

In order to work with the lists of presets there are three buttons between them "loadprstAddRemove" [loadprstViewPrst] and "delete preset" [loadprstDeletePrst]. "LoadprstAddRemove], View [loadprstViewPrst], Remove Preset [loadprstDeletePrst]. In addition to view presets buttons are equipped with work with many rows in the list and work by a simple click trigger (if you select more than 0, then do action), the view button has a condition to view only 1 preset.

If the preset view is clicked, the [LoadPresetViewPresetAction] trigger is started and the global identifier of the preset being viewed [gChsdPrstIndx] is selected from the list of readable presets [gAllPrsts] which will refer to the global array of presets [gAllPrsts] to read. This immediately reads the item description in the textarea [loadprstDescription] of the second subsystem of the first window.

The second subsystem is implemented as an independent subsystem with partial integration with the presets loading subsystem. Integration takes place through the radiobutton "Work mode" [loadprstWrkMode], which allows to change the type of view (description/items) by the [LoadPresetWrkModeAction] trigger.

To enable viewing of items, press the "Work mode" button, after which a second window will appear above the panel with the description, in this panel the manual control of viewing items is implemented in view of optimization. If you need to view only the description of the preset, you can not switch mode because the first window [loadprstDescriptionPane] with the description is shown in the first mode.

Also the subsystem is divided into two parts vertically, the left half has two lists for viewing items and lines, the right half shows the selected positions, namely label "Preset" [loadprstChosedPrstLabel], label "Items from preset" [loadprstChosedItemLabel] and label "Item Name" [loadprstNameItemLines].

The label names will be selected by three buttons, the first of which is "View" [loadprstViewPrst] responsible for the label "Preset" [loadprstChosedPrstLabel], the button "Load. Preset" button [loadprstLoadItms] with a simple trigger [LoadPresetLoadItemsAction] implementing the reading of all items from the preset [gAllPrsts] to the list [loadprstItemsInLoadPrst] from the item with the identifier [gChsdPrstIndx], the "LoadLoadItems" button responsible for the "preset" label [loadprstChosedPrstLabel]. Lines" button [loadprstLoadLinesInItm] with the trigger [LoadPresetLoadLinesInItemAction] responsible for loading all the characteristics of the selected item from the list of items [loadprstItemsInLoadPrst] to the list [loadprstItemsLinesVievMode] from [gAllPrsts].

The list of all features has two modes of operation through the radiobutton "Lines/ Array in Subject" [loadprstLinesArray] lines are read each time from the array of presets [gAllPrsts] using the [LoadPresetLinesArrayAction] trigger.

4.6.2 Schematic diagram of the preset module and preset load module

Opening the "Working with presets" pop-up window in the menu section

Delete lines

Replacing a selected item

Delete item

Created preset with items to create a preset file

Adding characteristics/array of characteristics to a preset subject

Change the display of the Subject/Line list

Selected subject

Creating a Preset

Adding items to a preset

Selecting a preset to replace

Entering a preset description

Entering a preset name

Writing a preset to a file

Beginning of the change

Opening the preset save panel

Cancellation

Save

Button replace preset (changes the order of saving)

Preset save button

Preset load button

Selected List Display Mode

Item editing panel

Opening the presets editing panel in the window

General view of the preset save panel

Addition when changing a preset

Item selection module

Changing the global panel

Changing the global panel

Displaying the list of added objects and lines in the created preset

Change the display of the String/Line array list

Preset reading subsystem

Closing the presets loading panel and opening the main global panel

Changing game files and saving files in the application directories

Load/unload analysis subsystem

Button to close the global item download panel

Load presets button

Load/Unload preset from game button

Button Delete Preset

Global list of files to upload a preset

Preset view button

Load item button

Button to load lines

List of strings/arrays of strings from the preset subject

List of items from the preset

Selected display mode

Display mode change button

Text description of the preset

Presets loading panel

Button to change the operating mode of the preset view

Selected preset view mode

List of downloaded presets

List of available presets

Global list of application presets

Opening the presets editing panel in the window