Construction Set Extender

shadeMe

Version 6.3

Introduction

The Construction Set Extender is an OBSE plugin that enhances the TES4 Construction Set by fixing various bugs and adding new tools. The CSE can be used to create any plugin and it doesn't add any dependencies to them.

Enhancements

Handling of Plugins and Masters

Creation and Modification of Master Files:

Master files can be edited and saved in the CS by setting them as active plugins. They will retain their master file status upon saving.

Modification of Master File Header Data:

The Author and Description fields of master files are no longer disabled by default and can be edited like any other plugin file.

Removal of the Need for Mod De-isolation:

The CS will now automatically save loaded ESP files as masters of the active plugin. The behaviour can be toggled through the Save Options sub-menu in the File menu.

Saving Plugins as ESM Files:

The CS can now save plugins as either ESP or ESM files.

Save As Option:

Active plugins can be saved under a different name by using the new Save As option, found in the File menu.

Timestamp Preservation:

The editor is now able to save plugins without modifying their Last Modified file timestamp. The behaviour can be toggled through the Save Options sub-menu in the File menu. This will preserve load order while editing plugins.

Automatic Backup Creation:

Backups of the active plugin get saved to the Backup folder in the workspace's Data directory just before a plugin save operation begins. The behaviour can be toggled through the Save Options sub-menu in the File menu.

Work-flow Improvements

Start-up Options:

The CSE allows the user to automatically perform the following operations right after the editor's start-up:

- Load a plugin and set it as the active file
- Load a script
- Change the active workspace

The start-up plugin can be set in the Data dialogue, by selecting the required file in the list-view and clicking on the *Set as Start-up File* button. The start-up script/workspace can be set through the CSE Preferences dialogue, which is invoked from the File menu.

Workspaces:

The CSE allows the user to switch between multiple working directories when using the editor. Each workspace can be considered a separate root directory (one that contains the Data folder) that can house plugins and asset files independent of each other. Custom workspaces need to be placed inside the original game directory. Master files present in the default workspace (*<root>\Data*) are shared with custom workspaces.

The Set Workspace tool can be accessed from the File menu.

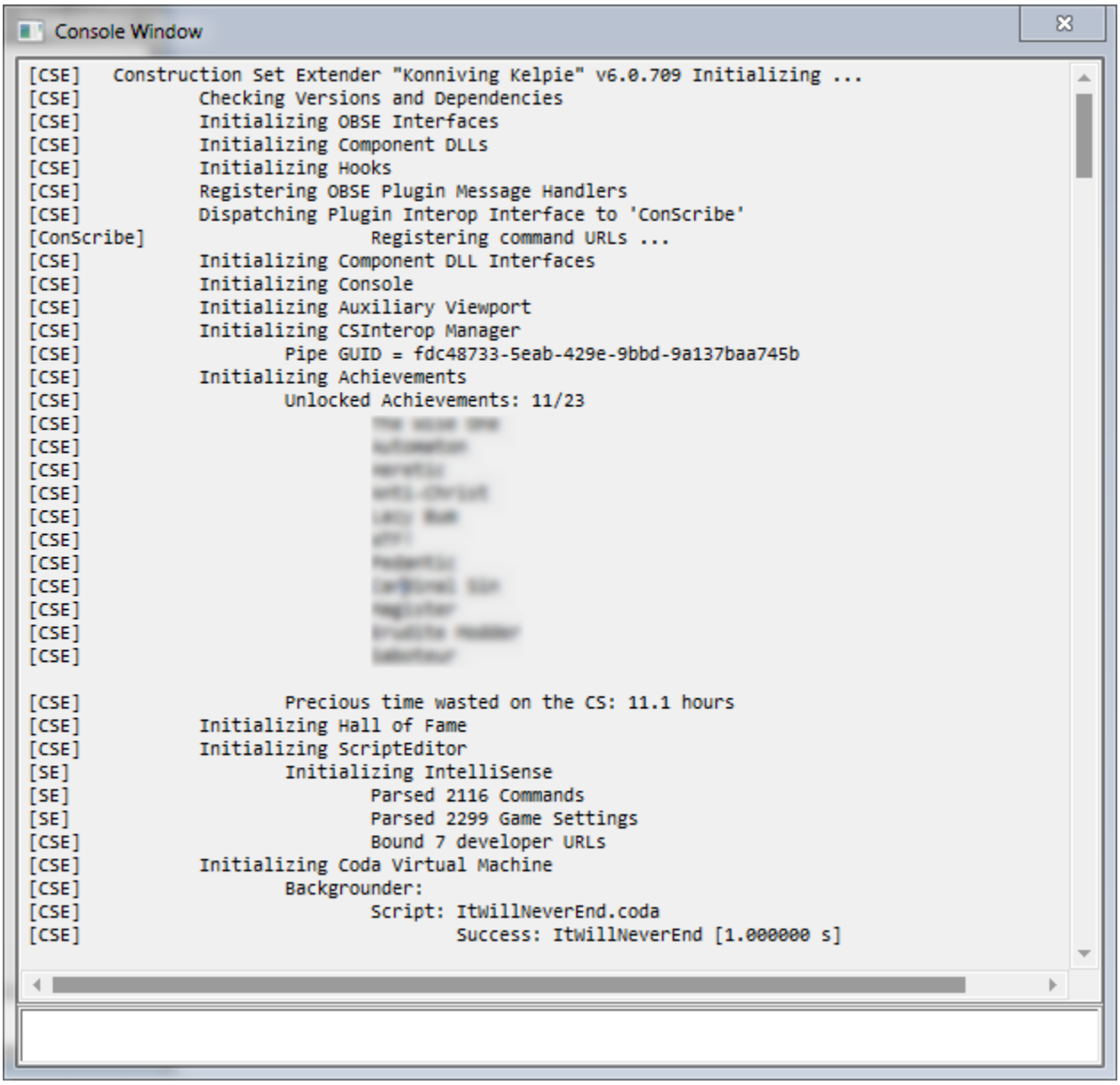
Setting up a Custom Workspace

Select Set Workspace from the File menu. Navigate to the Oblivion game folder. Click the *Make New Folder* button to create a new folder. Click OK. The CSE will reset and a message will pop up notifying that it's using a new workspace. Now the user may open Windows Explorer and copy any needed resources and plugins to the Data directory the CSE has created in the new workspace's folder. The CSE will automatically share master plugins with the workspace. It will also use the resources, like meshes and textures, from the default workspace (the regular Oblivion Data directory).

Console:

The Console Window is the standard output for all of the Construction Set's (and CSE's) output operations. It logs messages from various components of the CS, giving each an identifiable prefix. Its various tools can be accessed from its context menu, which can be

brought up by right clicking anywhere in inside the window. Specific editor warnings can be toggled through the same.



Certain commands can be entered and executed through the command line at the bottom of the window. Default commands:

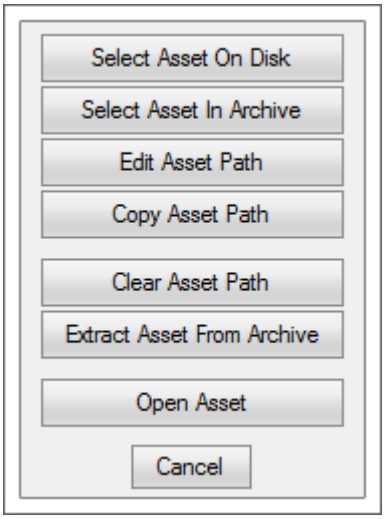
- *LoadPlugin string:<plugin name.extension> bool:setAsActive* – Marks the parameter plugin as loaded and initiates plugin loading.
- *LoadForm string:<editorID>* – Opens the parameter form's dialogue for editing. References will be loaded into the render window.

- *SavePlugin* – Saves the active plugin.
- *AutoSave* – Saves the active plugin to "Data\Backup\" as a copy.
- *Exit* – Closes the CS.

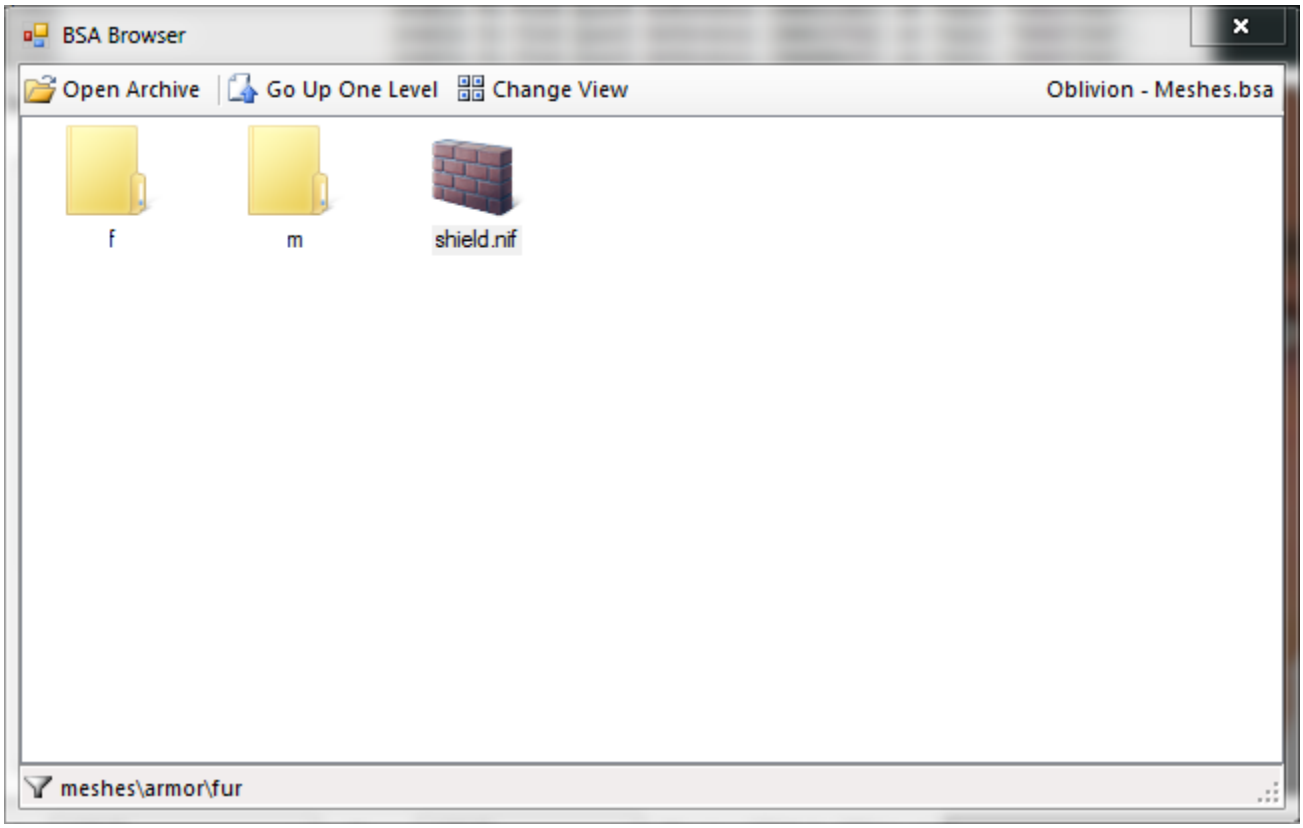
The command history can be walked through with the up and down arrows keys at the command line. This is also where Coda scripts are executed. Refer to the separate Coda documentation for details.

Enhanced Asset Selection:

Asset selection i.e., the selection of textures, meshes, sound files, Speedtree files and animation files, has been overhauled for intuitive access. Clicking on an *Add Asset* button brings up the dialogue shown below.



- Select Asset On Disk – Opens the regular Open File dialogue for disk access. The starting directory automatically defaults to the asset's previous path, if any.
- Select Asset In Archive – Opens the BSA browser, allowing the direct selection on assets inside BSA archives.



- Edit Asset Path – Allows the direct editing of the asset's file path.
- Copy Asset Path – Allows the quick copying of asset file paths between different records.
- Clear Asset Path – Resets the file path.
- Extract Asset From Archive – Searches for the file inside any of the loaded archives and extracts it, if found.
- Open Asset – Opens the asset file in its default viewer.

Additionally, hovering the mouse pointer over an asset button or combo box will display its value in a tooltip.

Context Menu Tools:

CSE adds a number of new tools that can be accessed from any form's context menu. To open the context menu for object references, right-click on an empty area in the Render window or right-click on the object in the Cell View window.

- Set Form ID – Allows the Form ID of a form to be changed.
- Mark As Unmodified – Reverts the "Modified" flag on a form, preventing it from being saved to the active plugin.

- Undelete – Resets the "Deleted" flag on a form.
- Show Override List – Displays a list of all the loaded plugins that modify the form in question.
- Edit Base Form – Only visible for object references. Opens the reference's base form edit dialogue
- Toggle Visibility – Only visible for object references. Toggles the visibility state of the reference.
- Toggle Children Visibility – Only visible for object references. Toggles the visibility state of the reference's linked children.
- Preview – View the form's 3D representation in the Preview window.
- Jump To Central Use Info List – Displays the Centralized Use Info Listing window and scrolls to the form in question.
- Add To Active Tag – Adds the form to the Tag Browser's selected tag, if any.
- Export FaceGen Textures – Only visible for NPC records. Generates and saves the record's facegen textures.
- Copy To Global Clipboard – Copies the form to the Global Clipboard.
- Replace Base Form – Only visible for object references. Changes the base form of the reference.

Batch Copy Eyes/Hair:

Hair and eye records of one race can be copied to another by using the *Copy Hair*/*Copy Eyes* buttons found in the Body Data tab of the Race edit dialogue.

FaceGen Editing:

Dialogs with FaceGen edit controls now update their preview windows in real-time, i.e, the mouse button does not need to be released for the preview to update. Advanced parameter editing has been made easier with a new shortcut – Using the mouse wheel while holding down the right mouse button over the attribute list will automatically move the Value slider.

Rearrange Effect Items:

Magic items with effect lists can have the order of their effect items changed. The CTRL+UP/DOWN key combos are used to move the current selection up and down respectively. Additionally, the sorting of the effect item list view has been disabled to reduce confusion when calculating the indices of the constituent effect items.

LOD Texture Generator:

The following improvements have been made to the LOD texture generator:

- Diffuse map and normal map textures are created with the appropriate MIP map chains.
- Placed object references are rendered to diffuse map textures.
- The resolution of diffuse maps can be customized through the CSE Preferences dialogue, with the new upper-limit being 6144px.
- Performance and stability have been improved significantly.
- Partial textures are deleted after the full LOD map is assembled, as dictated by the relevant setting in the CSE Preferences dialogue

Live Change Log:

The live change log is a tool implemented by the CSE that tracks modifications made to records in real time. Currently, the following actions and events are supported:

- Form Instantiation.
- Editor ID Change.
- Form ID Change.
- Form Active/Modified Flag Change.
- Form Deleted Flag Change.

Tracked changes are logged to the session's log with their timestamps. The session log can be accessed from the Console's Contexts... context menu item (right-click in the console window). If automatic plugin backups are enabled, the change log pertinent to that plugin's session is saved along with it.

Script Compiler:

The following improvements have been made to the script compiler:

- Compiled byte-code size for scripts has been increased to 32KB.
- Compiler errors accumulate, i.e., script compilation will not halt on encountering an error.

Quick Look-up Editor ID:

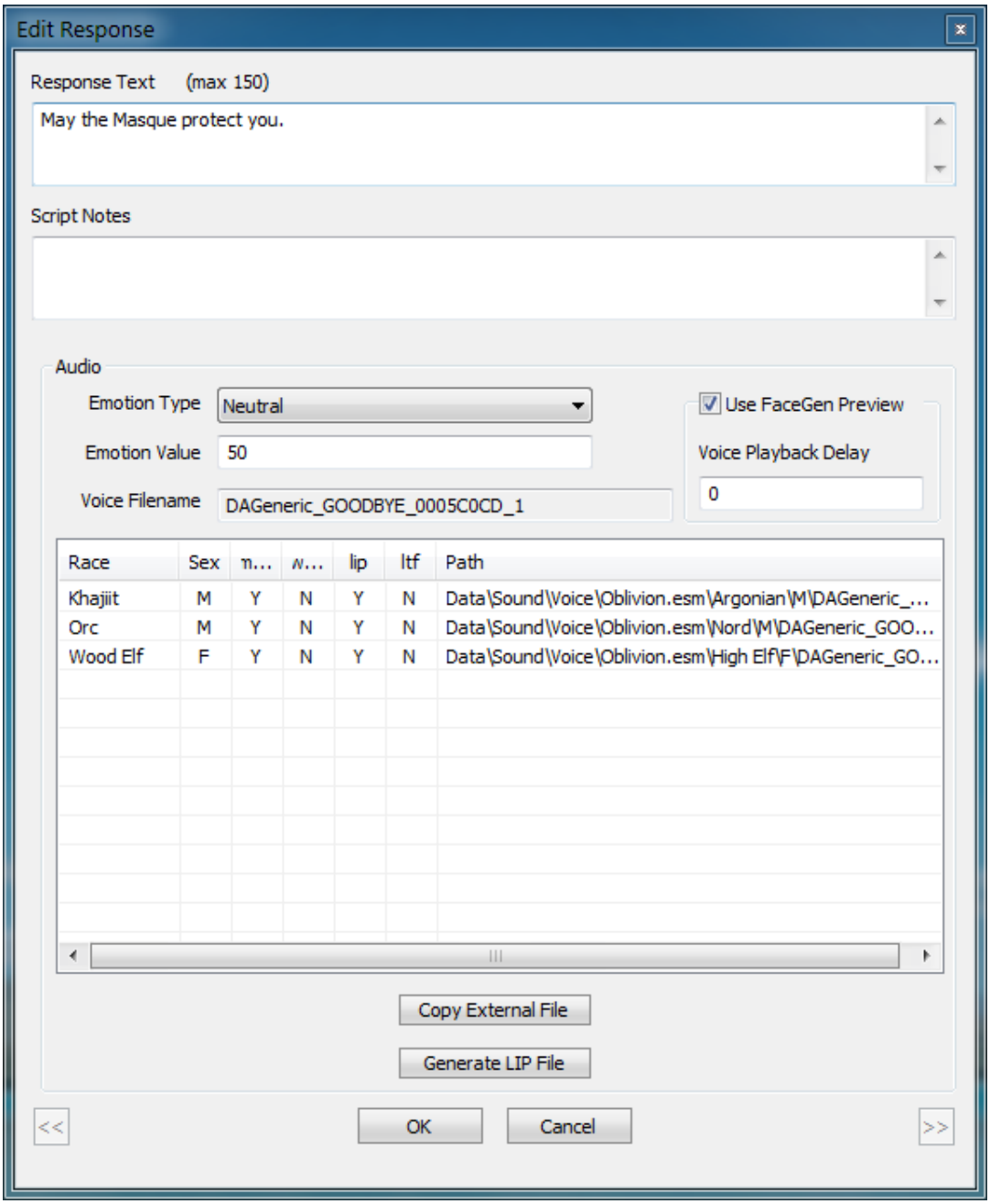
Middle clicking on a dialog control (buttons, text boxes, list-view cells, combo boxes, etc.) whose text states a form's Editor ID will bring up said form's edit dialogue.

Global Script Creation:

This tool allows quest scripts to be quickly created by specifying the editor IDs of both the quest and its script, along with the processing delay time. It can be accessed from the Gameplay menu.

Enhanced Response Editor:

The response editor has been modified to provide a more streamlined interface. The voice recording tool has been removed, given its general buggy'ness and obsolescence in comparison to 3rd party recording tools such as Audacity. A 'Copy External File' tool has been added. It allows the user to move recorded voice files from arbitrary workspaces into the CS workspace. It works on a per-race, per-sex basis – the target voice must be selected from the voiced races list in the editor.



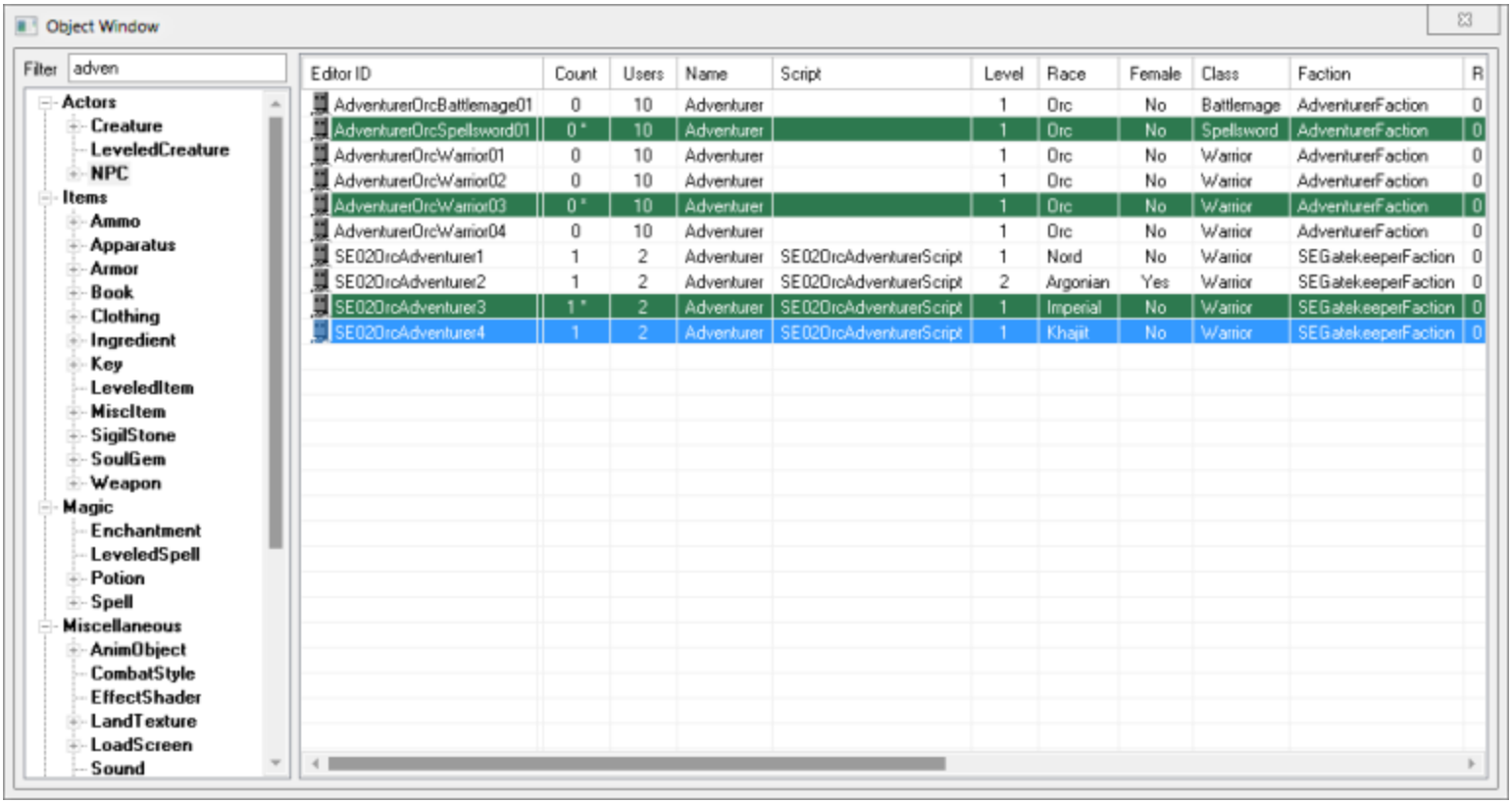
CSE removes the need for switching between different editor versions to generate LIP files for voices – It implements the lip sync generator in the latest version of the Construction Set. LIP files are generated on a per-race, per-sex basis, similar to the 'Copy External File' tool. The lip generator no longer needs a valid WAV file of the recorded voice – it will automatically convert the source MP3 file, if any, to WAV during generation.

CSE also adds a batch LIP generator tool that generates LIP files for all loaded topics. It can be accessed from the Character menu.

Double clicking on an item in the listview will playback the corresponding voice file. If the 'Use FaceGen Preview' checkbox is enabled, the voice file's lip synch animation is played in a NPC edit dialog, if one is open. The playback delay textbox can be used to specify the time delay, in milliseconds, between the voice file's playback and the lip synch animation's. Alternatively, a voice file-lip synch file pair can be directly drag-dropped into a NPC edit dialog's preview control.

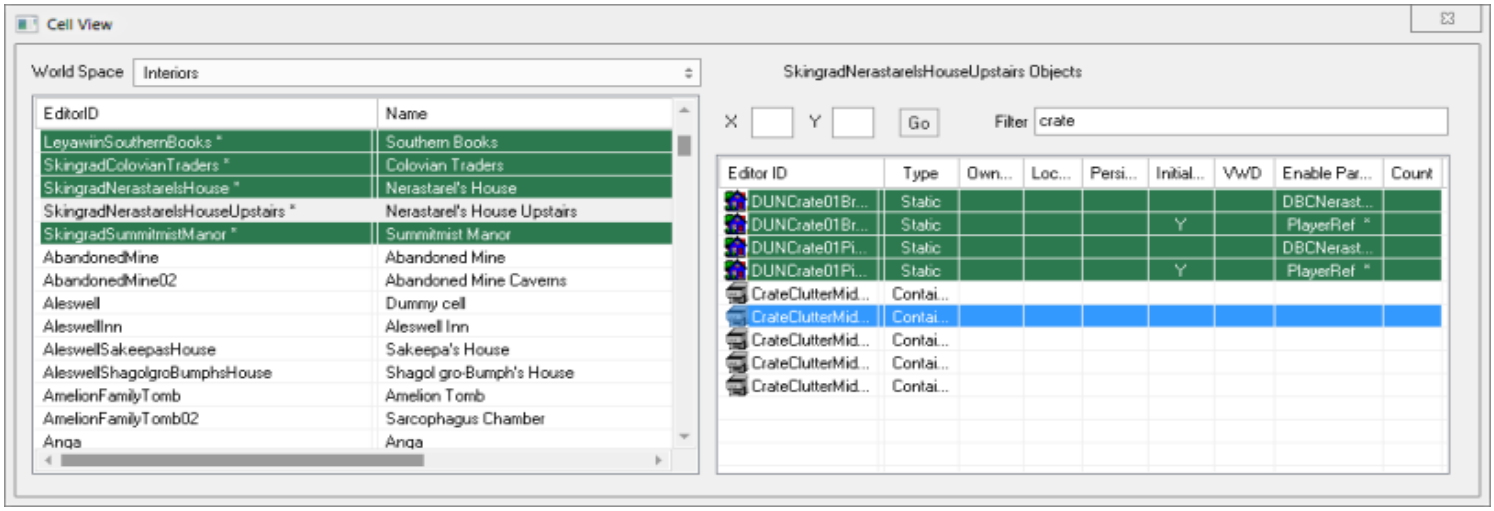
Object Window:

The following improvements have been made to the object window:



- Filter By Editor ID – The object list can now be filtered by a search string. The filtering is performed as a case-insensitive sub-string search in each form's Editor ID, Form ID, full name and description components (if any).
- Note that the active form colourization is enabled but not the active form sorting – see the explanation below for enabling these features.

Cell View Window:

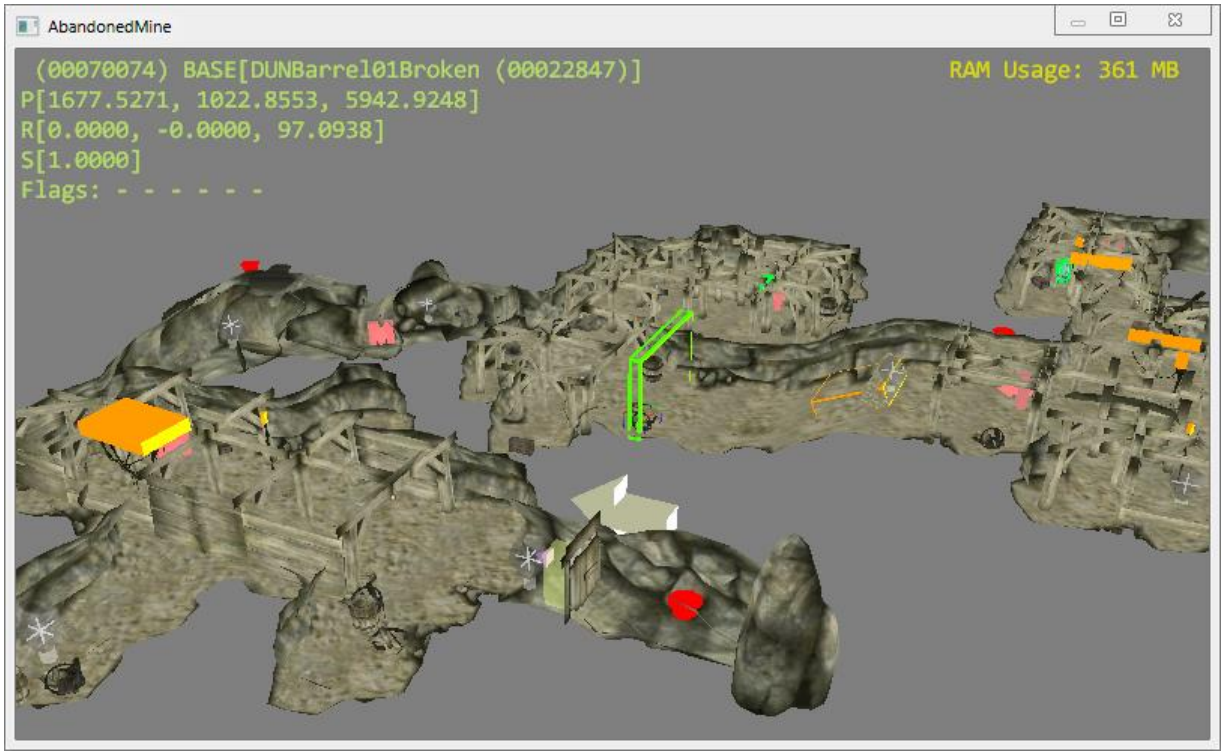


The following improvements have been made to the cell view window:

- Filter References By Editor ID – The references list can be filtered by a search string, similar to the object window. If a reference doesn't have an Editor ID, the filtering is performed on its base form.
- Jump to Exterior Cell – Enter the X and Y coordinates, hit *Go* and wait till a velocity of +88mph is attained.
- New Columns – The following sortable columns have been added to the reference list:
    - Persistent
    - Initially Disabled
    - Visible When Distant (VWD)
    - Enable Parent
    - Count

Render Window:

The following improvements have been made to the render window:



- Increased Responsiveness – The viewport now operates at an average frame rate of 48 FPS.
- On-screen Display – Detailed information on the active selection and the editor's current RAM usage are displayed in the top corners of the window. The details of the object under the mouse pointer are also displayed.
- Door markers Properties – Holding down the Control key and double clicking on door markers will now bring up their reference properties dialogue box.
- Alternate Camera Movement Settings – The render window is now allowed to have a second set of camera/reference movement settings, editable through the CSE Preferences dialogue.
- Initial Camera Placement – On loading an interior cell, the camera is placed at the location of the first reference. If the cell is empty, it is placed at its origin.
- Reference Scaling – Collections of references can be scaled relatively by holding down the ALT modifier key while performing the scaling operation.
- Auxiliary Viewport Window – The CSE adds a secondary viewport that allows the current render window scene to be viewed from a second camera/perspective. The auxiliary viewport's camera can be set directly from the render window, with the "Freeze/Release auxiliary viewport camera" shortcut key mentioned below.

  It can be accessed from the View menu.

- Path Grid Editing Enhancements –
    - Path grid points can be unlinked from their linked references/relinked.
    - Path grid point operations can be undone/redone[1]
- Landscape Editing Enhancements –
    - The active landscape texture can be changed from the Landscape Texture Use dialogue, by double clicking on any of the listed land textures.
    - The upper limit on the landscape edit brush's radius has been increased to 25 units.
- Fly Camera Mode – When enabled, the viewport camera can be controlled with the keyboard and the mouse.
    - WSAD – Directional movement.
    - Space/Ctrl – Rise/Sink.
    - Shift/Alt – Sprint/Crawl.
- Custom FOV – The render window's horizontal field of vision cone can be customised in the INI file.
- Disabled Reference Visibility – The visibility of disabled references and their children can be toggled from the View menu.
- New Context Menu Tools –
    - Invert – Inverts the render window selection.
    - Relative Alignment – Aligns the selection to one of the 3 axes of its first reference.
    - Grouping – Allows multiple references to be grouped as a collective object. Selecting a member of a group using the left mouse button[2] causes all of its members to be selected. This behaviour will persist until the selection group is dissolved. A reference can be a member of only one group at a time.
    - Freezing – Freezes the movement of the selection. Frozen references cannot be selected[2] in any way until they are thawed.
        - Thaw All In Cell – Unfreezes all frozen references in the loaded cell.
        - Freeze All Inactive – When enabled, all unmodified references are implicitly frozen to prevent accidental edits.

---

1 Path grid point deletion operations are only supported to a limited extent. Undoing a delete operation will not restore the state of the linked points.

2 Reference selection behaviour can be temporarily reset by holding down the Alt key while selecting objects.

- Visibility – Toggles the visibility of the selection/selection's linked (enable parent) children.
  - Reveal All In Cell – Resets the visibility state of all the references in the loaded cell.
- Batch Edit – Displays the Batch Reference Editor for the loaded cell.
- Co-Planar Drop New References – When enabled, new references will be placed co-planar with the object at the cursor location.
- Switch C & Y Hot-keys – When enabled, the functionality of the C (Isometric view) and Y (Axis modifier) is switched.
- Unload Current Cell(s) – Unloads the cell(s) loaded into the render window.
- Use Alternate Movement Settings – Toggles between the default and alternate camera movement settings.
- Object-Under-Pointer Display – Toggles the pop-up display.
  - CTRL Key Modified – When enabled, the pop-up only shows when the Control key is held down.
- Save Exterior Cell Snapshot – Saves a DDS snapshot – as seen in the render window – of the current exterior cell (grid-centre) to the Data\Textures\Landscape directory. The resolution of the snapshot can be customized in the CSE Preferences dialogue.

Shortcut Keys

Shortcut Key	Action
F1	Show use info report for selection
F5	Refresh viewport/reveal all invisible references in the current cell
2	Toggle selection visibility
G	Toggle path grid editing mode
SHIFT + CONTROL + Mouse Move	Rotate camera around static pivot
SHIFT + CONTROL + ALT + V	Invert selection
SHIFT + E	Toggle co-planar dropping
SHIFT + Q	Freeze/Release auxiliary viewport camera

CONTROL + Q	Switch between the vanilla and alternate movement settings
CONTROL + R	Unlink path grid points from their linked references
`	Enter/exit fly camera mode
CONTROL + A	Select all references in the current cell

Global Clipboard:

This tool allows easy interoperability between multiple CSE sessions, i.e., it allows records in one extended CS session to be copied to another extender CS session. All form types, apart from the following, are supported by this tool:

- Skill.
- Magic Effect.
- Script.
- Cell.
- Region.
- Path Grid.
- Land.
- Road.
- Topic.
- Topic Info.
- Idle.

Forms can be copied with the 'Copy To Global Clipboard' context menu tool and pasted with the 'Paste From Global Clipboard' Edit menu tool. Different form types cannot be mix-matched when copying multiple forms. Copied forms are assigned new formIDs unless they override existing forms, i.e., use the same editorID. Object references are pasted in the render window's currently loaded cell.

The contents of the global clipboard can be viewed through the View main menu.

Global Undo Stack:

The global undo stack adds the ability to undo (and consequently, redo) changes made to records. All form types, apart from the following, are supported by this tool:

- Topic.
- Topic Info.
- Idle.
- Cell.
- Region.
- Land.
- Reference.
- Path Grid.
- Road.
- Quest.
- Script.

An undoable operation is recorded every time changes are committed to a record, i.e., when record is marked as modified. The 'Global Undo' and 'Global Redo' menu items in the Edit main menu can be used to undo/redo any previously recorded operations.

User Interface Improvements

Active Form Sorting:

Active forms, i.e., modified records, can be sorted to the top of most form lists (a list-view that displays records) that support sorting. This behaviour can be toggled from the View menu.

Active Form Colourization:

The foreground and background colours of active form items in form lists can be changed from their defaults of black and white. This behaviour can be toggled from the View menu and the colours can be changed through the CSE Preferences dialogue.

Override Form Colourization:

Similar to the above but colourizes form items depending on their override level, i.e., how many of the loaded plugins override them. Overrides of up to 4 levels are supported.

Form Enumeration Filters:

The Hide Unmodified Records and Hide Deleted Records menu options can be used to toggle the display state of forms that haven't been modified by the active plugin or have been deleted, respectively. They can be accessed from the View menu.

Main Editor Windows' Visibility:

The object, cell view and render windows, upon hiding, are completely hidden instead of being minimized to the bottom of the desktop. Their visibility state is also preserved between CS sessions.

Taskbar Visibility:

Almost every editor dialogue can be made to show up in the taskbar. This behaviour can be toggled from the CSE Preferences option in the File menu.

Search and Replace:

The Search & Replace dialogue no longer closes itself after a successful replace operation.

Enhanced Find Text:

Entries in the find text dialogue can be invoked directly for editing, i.e., double clicking the results of a search will bring up the corresponding item's dialogue box or load the object into the render window, if it is a reference. Entries can also be drag-dropped into valid destinations like the Render window.

Safer Modification of List View based Records:

Forms of types such as Magic Effect, Race, Eyes, Hair, etc. (with the lone exception of Quests) are displayed as a list when editing them in the CS. The default behaviour of such dialogues leads to many a dirty edit. CSE attempts to fix it by introducing the following changes:

- The *OK* and *Cancel* buttons are changed to *Apply* and *Close* respectively, thereby clarifying the actions they perform, i.e., *Apply* saves any changes made to the currently selected record while *Close* discards them and closes the edit dialogue
- The newly minted *Apply* button no longer closes the dialogue, thereby allowing the user to continue editing after saving.
- When switching to a different record, a Save Changes confirmation is displayed. Changes are saved only when the user selects *Yes*.

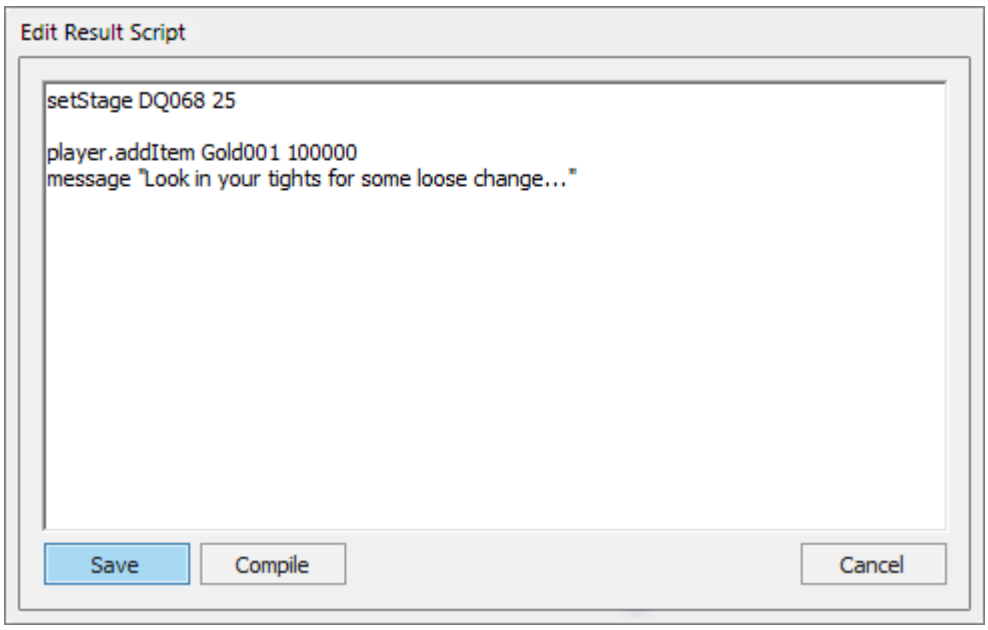
In addition to the above, CSE allows multiple list items to be selected for drag-drop operations. For instance, you could select multiple Hair or Eye records and drop them into a Race record in a single go.

Launch Game:

The Launch Game button, found on the main toolbar, is primarily used to spawn a mad, wild killer bull – cunningly disguised as a bird – that's got winning odds of 80000 to 1 in a cock fight.

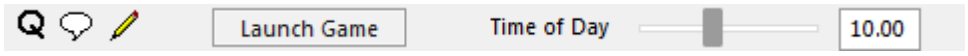
Result Script Editing:

Result scripts can now be edited in a larger dialogue by clicking on the "…" button placed next to them.



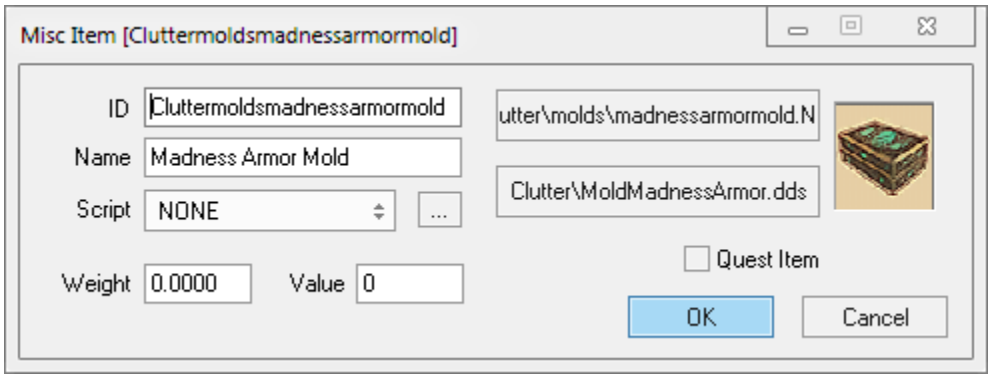
Time of Day Slider:

A Time of Day slider has been added to the main toolbar. It requires sky rendering to be turned on.



Editor IDs in Edit Dialogue Titles:

The editor ID of the form being edited is now displayed in the dialogue window's title bar.



Improved Dialogue UI's:

The following dialogues' UI's have been improved to enhance usability and better use the available screen real estate:

- Filtered Dialogue.
- Quest Editor.
- Dialogue Editor.
- Find Text.
- Idle Animation.
- AI Packages.
- AI Data.
- Select Topic/Quest.
- Data.
- NPC.
- Creature.
- Preview.
- About.
- Region Editor.
- Landscape Edit.
- Climate.
- Worldspace.
- Hair.
- Eye.
- Game Setting.
- Global.
- Birthsign.
- Race.
- Load Screen.
- Static.
- Clothing.
- Armour.
- Ammo.

- Weapon.
- Book.

Trifles

Performance Improvements:

The editor's general performance and responsiveness has been noticeably improved.

Fast Exit:

The editor shuts down in matter of seconds, as opposed to minutes when not using the CSE.

Icons with MIP maps/Texture Size Limitations:

Icons with MIP maps can be previewed correctly and the CS no longer generates errors about the matter. Also, the resolution limitation of 512px for certain textures has been removed.

Auto-loading BSA Archives:

All BSA archives in the Data folder are loaded at start-up, regardless of their connection to an active plugin.

Integer-prefixed Editor IDs:

Editor IDs that start with integers display a warning, reminding the user of the caveats of using such identifiers. This behaviour can be turned off through the CSE Preferences dialogue.

Idle Animation Tree Initialization:

The root nodes of the idle animation tree are automatically initialized on editor start-up, enabling master-less plugins to create IDLE records.

Archived Sound File Sampling:

Sound files, FX and voice files alike, that stored in BSA archives can be sampled from the editor directly without having to manually extract them beforehand.

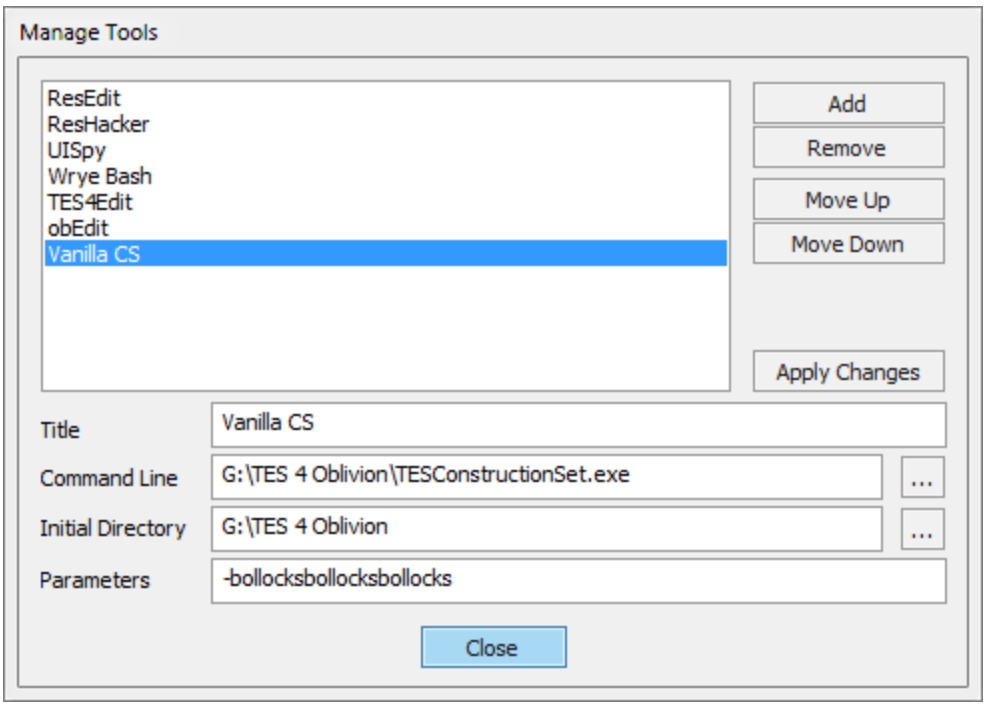
Retroactive "nVidia Fog Fix":

The CSE will automatically correct the 'Near Fog Plane' attribute of delinquent cell records (records with a value less than 0.0001) when loading plugins.

Circular Leveled List Detection:

The CSE will automatically check for circular paths when saving leveled list records.

3rd Party Tool Launcher:



The CSE implements a rudimentary tool manager/launcher, with which 3rd party software can be launched directly from within the editor environment.

Both the manager and the list of registered tools can be accessed from the Tools menu.

Achievements:

The CSE smugly awards achievements to the user – for "doing stuff" – when they least expect it. It also tracks the total amount of time spent using the editor, a metric that is primarily used as an insidious means to instigate an existential crisis in the user.

Last Chance "Panic Save" Handler:

Let's face it – The CSE is an almost-sentient, handsome stallion of a software. But it has "got heart", as film critics like to say. And in that "heart", it houses the trait of humility. While it knows that it's capable of feats beyond the vanilla CS's wildest dreams, it realizes that it isn't omnipotent and cannot predict when its finicky host is about to vomit on the proverbial kettle. But it *can* do the next best thing –Try and shield the active plugin from that malodorous mix of bodily fluids and half-digested breakfast. It does so by attempting to save it on the event of a CTD.

If successful, the plugin is saved to the workspace's "Data\Backup" folder under the name of "PanicSave.bak". The dumped plugin may contain errors and/or lice, so approach with caution.

Vanilla Bug Fixes

The CSE fixes a ton of bugs and quirks present in the vanilla CS. The complete list follows:

Response Editor Microphone	Fix for the CTD that occurs on machines with Realtek sound cards, when the response window is initialized without a microphone plugged in its socket.
Topic Info Data Reset	Fix for the bug that automatically clears result script text and info flags if they are visible when a new topic is added to the topic list.
NPC Editor Face-Gen	Fix for the CTD that occurs due to the improper initialization of the face-gen renderer in NPC and creature dialogues
Identical-To-Master Dialogue And Worldspace Edits	Fix for the version control related bug that makes unnecessary edits to cells, dialogues and worldspaces should one of the plugin's masters have an active record of the same.
Race Description Dirty-Edits	Fix for the bug that copies race description from one race record to another if the latter were to invoke the spell checker.
Code Level Assertions	This bug is deep rooted in the editor code and tends to cause a fairly large number of CTDs for no reason.
Unknown Record And Group Types In Plugins	Fix for the bug that caused a CTD when a plugin containing unknown records, sub records or group types was loaded into the editor.
Plugins With Missing Masters	Fix for the bug that caused the editor to exit when a loaded plugin was found to have a missing master.
Always-On-Top Data Dialogue	Fix for the bug that caused the Data dialogue to stay above all open windows.
Render Window Main Menu Item	Fix for the bug that prevented the Render window for being closed when using the View > Render Window main menu item.
Topic Info Copying	Fix for the bug that caused the wrong topic info record to be flagged as modified during a copy operation.
Lip Sync File Generator	The infamous lip sync tool has been finally fixed! See

	the Enhanced Response Editor section, above, for details.
Variable Declarations In Result Scripts	Fix for the bug that caused a CTD when compiling a result script that had local variable declarations.
New/Duplicate Form Selection	Fix for the bug that prevented newly created/duplicated list view-based forms from being selected after instantiation.
Reference Duplication	Fix for the bug that prevented the complete duplication of extra-data between object references.
Plugin Author/Description Editing	Editing a plugin's author/description field in the Data dialogue sometimes corrupts it, the cause of which appears to be related to file access permissions. CSE attempts to prevent this by pre-emptively check the plugin file's handles before committing such potentially dangerous changes.
Reference Record Serialization	Fix for the bug that prevented the correct serialization of deleted reference records under certain conditions.
Game Setting Clean-up	Fix for the limitation that caused modified game setting records to retain their state between plugin loads.
Form Usage Reference Counting	Fix for the bug that broke usage reference counting when a form was referenced more than once by another.
Anim-Object Creation	Fix for the bug that caused a CTD when an Anim-Object was created or edited when no plugins were loaded.
LOD Diffuse Map File name	Fix for the bug that caused incorrect file names to be given to generated LOD colour maps (which led to pink coloured LOD meshes in-game).
Plugin Save	Fix for the bug that prevented the Save Plugin action from being reused if a previous operation ended prematurely.
Incorrect Script Compiler Errors	Fix for the bug that displayed incorrect line numbers in script compiler error messages.
Light Object Reference First-Time Initialization	Fix for the bug that caused a CTD when a light object was placed in a cell for the first time.

Render Window “Fall” Operation	Fix for the bug that caused occasional CTDs when using the Fall tool.
Quest Stage Result Scripts	Fix for the bug that prevented quest stage result scripts from being cleared.
Reference Editor Z-Order	Fix for the bug occasionally caused reference edit dialogues to spawn underneath the render window.
NPC Editor Inventory Preview	Fix for the bug prevented NPC models from being updated correctly in the preview window after an item was removed from their inventory.
Faction Ownership In Conditions	Fix for the bug that prevented factions from being allowed as arguments to condition script commands that accepted parameters of the type "Owner".
ESP/ESM File Associations	Fix for the bug that trashed ESP/ESM file associations at editor start-up
Path Grid Point Linking	Fix for the bug that invalidated the render window after a path grid point was linked to a reference.
Cell View Window Bounds	Fix for the bug that progressively reduced the size of the cell view window’s controls every time plugins were loaded into the editor.
Theme-Enabled Owner-Drawn Controls	Fix for the bug that caused CTDs when painting theme-enabled owner-drawn preview controls.
Landscape Texture Change	Fix for the bug that caused the render window to lose input focus after switching the active landscape texture.
Path Grid Point Creation	Fix for the bug that caused a CTD when a path grid point was created after every loaded path grid was destroyed.
Path Grid Point Selection	Fix for the bug that prevented the selection rectangle from showing on multiple path grid point selections.
Render Window Reference Duplication	Fix for the bug that reverted the render window’s selection to the original references during a duplication operation.
Non-Standard Line Endings In Scripts	Fix for the bug that prevented the correct parsing of script source code containing non-standard line

	endings.
Dialogue Creation Failure	Fix for the bug that prevented modeless dialogues from being destroyed correctly, eventually exhausting the operating system's global window handle pool.
LOD "Black Texture"	Fix for the bug that caused half the pixels of generated LOD diffuse maps to appear black in colour
Cell Edit Dialogue Dirty Edits	Fix for the bug that caused dirty editors when selecting cells in the Cell edit window.
Superfluous Addition Of Cell Water Data	Fix for the bug that unnecessarily added water extra-data to cells that didn't have any water.
Render Window Exterior Cell Loading	Fix for the bug that caused the render window to flicker intermittently when loading exterior cells.
Interior Cell Duplication	Fix for the bug that prevented lighting data from being copied when an interior cell was duplicated.
Sweeping Path Grid Point Selection	Fix for the bug that caused the render window to select every path grid point in the loaded cell(s) while reclaiming input focus.
Quest Stage Log Entry	Fix for the bug that caused the quest editor to close whenever the Enter key was pressed inside the Log Entry text box.
"Path Grid Edit Mode" Toolbar/Menu Deactivation	Fix for the bug that caused the "Path Grid Edit Mode" main menu/toolbar buttons to relinquish their toggled state occasionally for no reason.
Path Grid Edit Mode Initialization	Fix for the bug that caused path grid points to incorrectly appear preselected when entering the path grid edit mode.
Loading Detached References	Fix for the bug that caused a CTD when a detached reference (a reference with no parent cell) was loaded into the render window.
Reference Variables In Compound Expressions	Fix for the bug that caused a CTD when a reference variable was used as a calling reference in a compound script expression.
Mismatching Parentheses In Set Expressions	Fix for the bug that prevented the script compiler from complaining about mismatching parentheses in Set expressions.

"Dirty Flag" Reset	Fix for the bug that caused the editor to reset its "Unsaved changes" flag when the Data dialogue was closed.
Weather Sounds List-view Sorting	Fix for the bug that caused a CTD when the sounds list-view in the weather edit dialogue was sorted.
Render Window Axis-Modifier Hot keys	Fix for the bug that caused the render window's axis modifier keys (Z, X and Y) to work incorrectly.
Recursive Loading Of Plugin Master Files	Fix for the bug that prevented the recursive loading of the active plugin's master files during a plugin load operation.
Color Control Blackout	Fix for the bug that caused the entire desktop to blackout when working with color picker controls.
Landscape Edit Window Crash	Fix for the bug that caused a CTD when the landscape edit window was opened and no cell was loaded in the render window.
AI Package Rename	Fix for the bug that caused the AI Packages dialog to close itself on renaming an existing AI package.
Region Data Copy	Fix for the bug that caused a CTD when a data copy operation was performed on a newly created region record.
Actor AI List-view Reset	Fix for the bug that caused the packages list-view in AI data dialog to reset its columns whenever it was modified.
Previewing Textures Without Mip-Maps	Fix for the bug that occasionally caused a CTD when previewing textures without mip-maps.
NPC Face Texture Exporting	Fix for the "bug" that prevented NPC records in ESP files from having their facegen textures exported.
Cell Testing	Fix for the bug that caused a CTD when testing cells.
Cell View Window List-View Sorting	Fix for the bug that caused the cell view window's object list-view to lose its sorting whenever a cell was selected.
Interior Cell Near Fog Plane	Fix for the bug that caused interior cells to blackout when their near fog plane distance was too low.
Invalid Script Condition Expression	Fix for the bug that caused script compilation to silently fail whenever an invalid condition expression was encountered.
Light Falloff Exponent	Fix for the bug that prevented the falloff exponent

	text field in Light object edit dialogs from being clamped correctly.
GMST Renaming	Fix for the bug that allowed game setting records to be renamed.
GMST Copying	Fix for the bug triggered an assertion when copying game setting records.

New tools

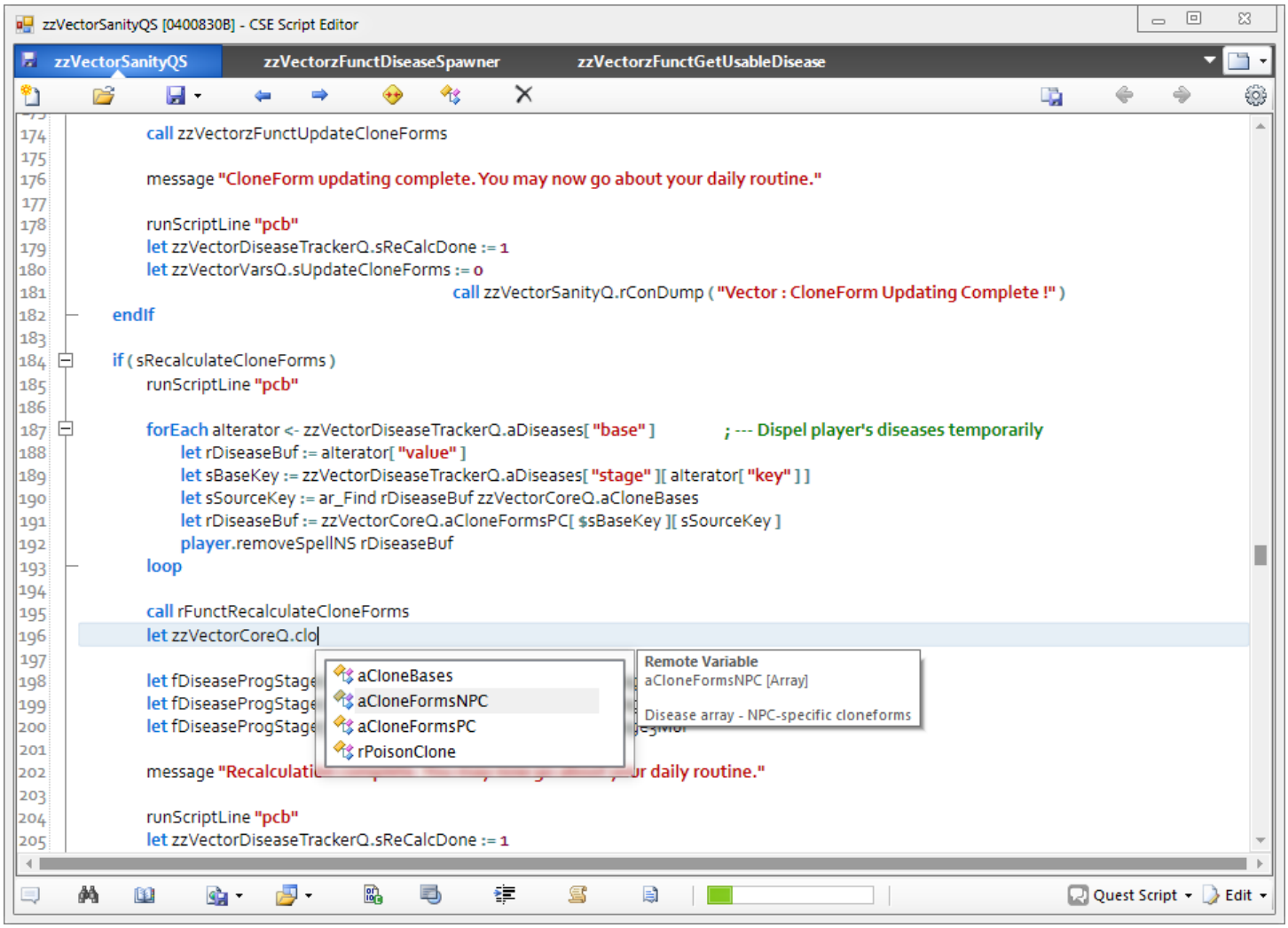
Coda Script

CSE introduces a new scripting language that allows the user to automate operations inside the editor environment itself. The complete documentation is bundled separately.

Script Editor

The CSE Script Editor is a complete replacement for the vanilla script editor. It has been (badly) written from scratch and is basically superior to the vanilla in every way. Its intuitive design allows scripters, old and new, to quickly acclimatize themselves with its many advanced features.

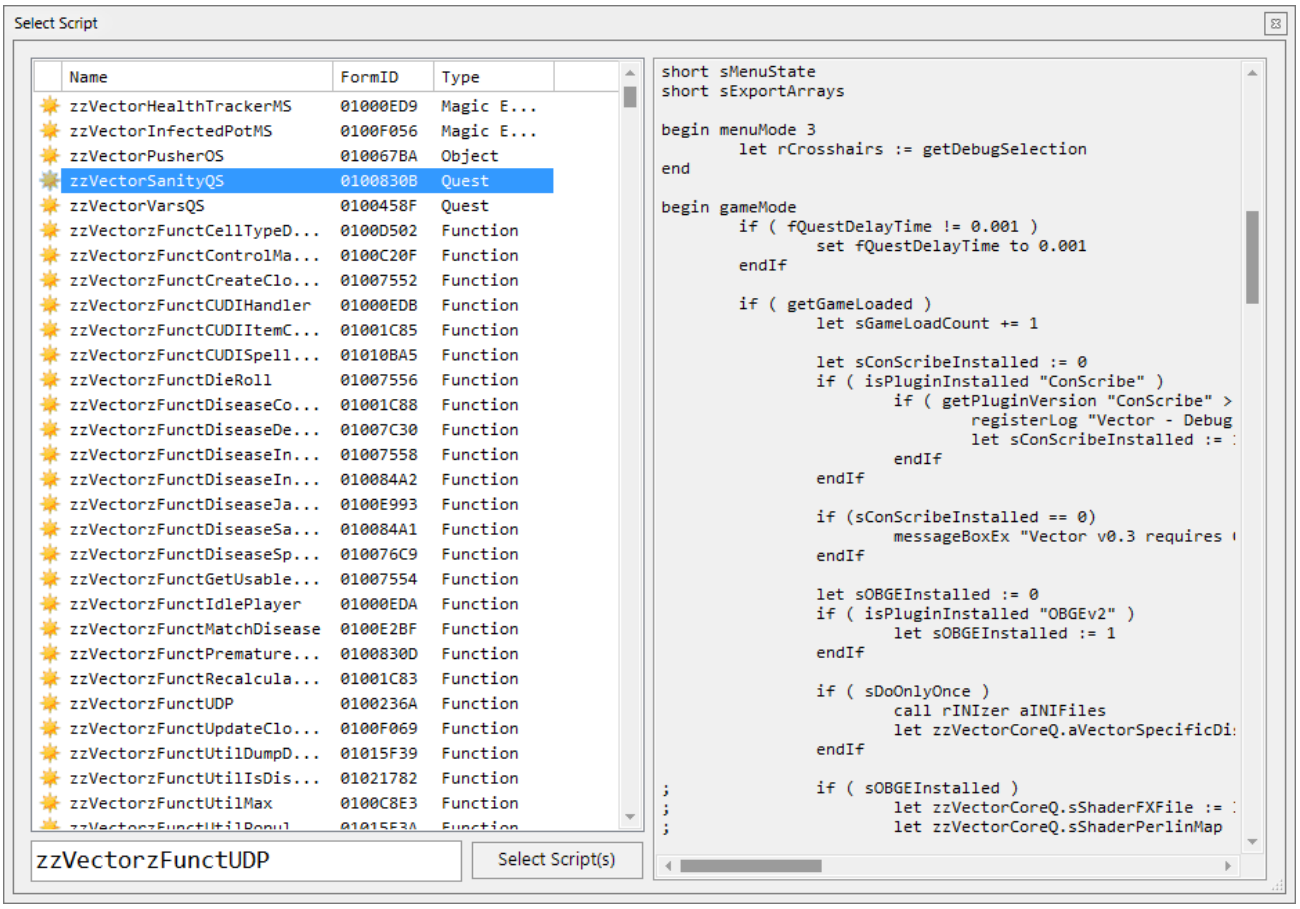
To begin with, the CSE Script Editor is a tabbed code editor. It can hold an arbitrary number of workspaces and allows operations such as tab rearranging and tab tearing. New workspaces can be instantiated with the *New Tab* button present in the tab strip's control box. Open workspaces can be sorted alphabetically from the control box's drop down menu.



Workspaces can be rearranged by dragging their tab across the tab strip. They can also be "torn" out of their tab strips and dropped onto either another editor window's tab strip or the desktop. The former relocates the workspace to the destination window, while the latter spawns a new editor at the drop location and moves the workspace into it.

The following is the list of the buttons/controls in a workspace in the order of appearance (Moving from top-left to bottom-right).

- New – Creates a new script.
- Open – Opens the Select Script dialogue for script selection.



The list view displays all the loaded scripts and the text box to the right shows a preview of the selected script. The textbox at the bottom can be used to select a particular script by its Editor ID or Form ID. Multiple scripts may be selected for opening. The list view can be sorted by each column. The first column denotes the state of each script – A golden star denotes that the script is from an active plugin and an "X" mark denotes that the script has its "Deleted" flag set. The list view is status-sorted by default.

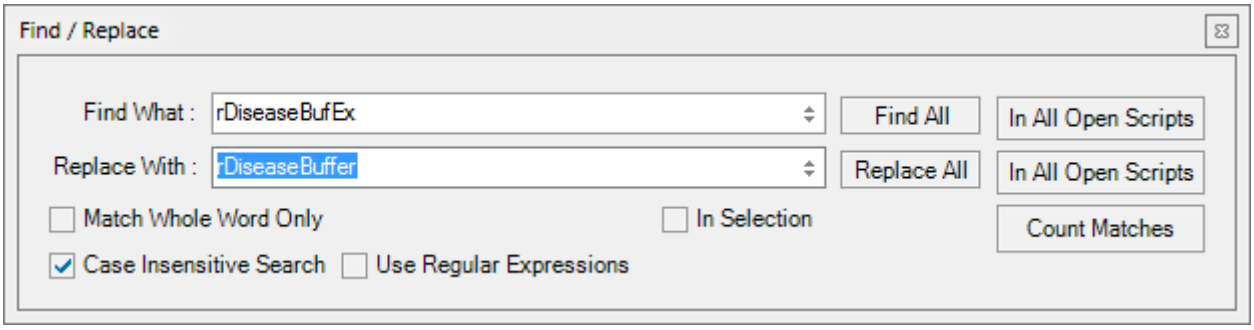
- Save – Attempts to compile and save the loaded script. This button has a drop down:
    - Save Script but Do Not Compile – Saves the script text without compiling it to bytecode. On loading a non-compiled script, the editor will warn the user about the script's status.
    - Save Script and Active Plugin – Attempts to compile and save the script, but saves the active plugin regardless of the compilation result.
- Previous – Loads the previous script, if any.
- Next – Loads the next script, if any.
- Recompile Active Scripts – Attempts to compile and save every script in the active plugin. Compilation results are logged to the console.

- Recompile Script Dependencies – Attempts to compile and save any scripts (regular and result scripts) that might reference the current script and prints a detailed report to the console.
- Delete – Opens the Select Script dialogue for script deletion.
- Save All Open Scripts – Attempts to compile and save all open workspaces.
- Navigate Backward – Jump back in the navigation stack (more on this later).
- Navigate Forward – Jumps forward in the navigation stack.
- Preferences – Opens the Preferences window. Some changes may require a restart of the script editor.

  A floppy-disc icon in a workspace's tab indicates that its script has unsaved changes.

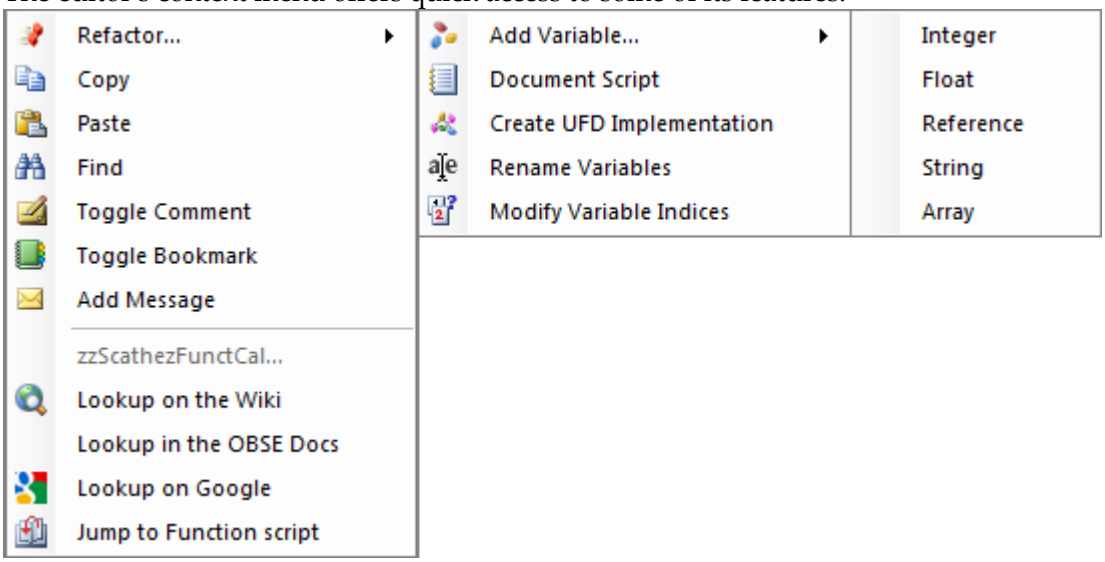
  The bottom toolbar is actually a splitter bar which can be moved to resize the editor area and show the controls beneath it.
- Message List – Displays output messages from the script validator, pre-processor and the compiler. Custom messages are also displayed here. Double clicking on an item will either move the caret to the appropriate line or remove it from the list depending upon the message type.
- Find Results – Displays the last executed Find/Replace operation's results. Double clicking on an item will move the caret to the appropriate line.
- Bookmark List – Displays stored bookmarks for the loaded script, if any. Double clicking on an item will move the caret to the appropriate line.
- Dump Script – Saves the loaded script as a file of arbitrary type in a selected folder.
  - Dump All Tabs – Saves all open workspaces to a selected folder as text files.
- Load Script – Loads a plain text type file from disk into the workspace. Replaces any existing content.
  - Load Multiple Scripts Into Tabs – Loads multiple scripts into a workspace of their own.
- Fetch Variable Indices – Enumerates every variable in the script, their type and index. Indices can be edited by double clicking on the desired cell.
- Update Variable Indices – To be used in conjunction with the above tool. Updates the script with the modified variable indices, if any. This is an advanced tool and must be used with care as it can easily break scripts.
- Toggle Offset Viewer – Displays line offsets in place of line numbers. Useful when debugging OBSE errors as they only mention offsets into script bytecode. This tool may only be used with compiled scripts.
- Toggle Pre-processed Text Viewer – Pre-processes the script text and displays the result in a separate text viewer.
- Sanitize Script Text – Performs various operations, as set in the Preferences window, on the script text to make it more legible. The following operations are supported:

  - Anneal Identifier Casing – Corrects the case of identifiers (Editor IDs, variable names, command names, etc.).
  - Indent Script Lines – Indents script lines according to block structure.
  - Eval'ify If/ElseIf Statements – Adds the "Eval" keyword to all If/ElseIf statements.
  - Apply Compiler Override To Script Blocks – Prefixes script blocks with the compiler override specifier – "_".
- Bind Script – Allows the current script to be bound to a new or an existing scriptable object.
- Code Snippet Manager – Brings up the Code Snippet Manager dialogue (more on this later).
- Progress Bar – Indicates the compiled bytecode size of the current script.
- Script Type Menu – Specifies the current script's type.
- Edit Menu –
  - Find/Replace – Displays the Find/Replace dialogue.



  - Goto Line – Jumps to the given line number. This tool cannot be used in the offset viewer.
  - Goto Offset – Jumps to the given script offset. This tool can only be used in the offset viewer.

The editor's context menu offers quick access to some of its features:



- Refactor Menu –
    - Add Variable – Appends a new variable to the current script's variable declaration block. If a string token is present at the context menu's location, it is used as the name of the new variable.
    - Document Script – Inserts comment-based documentation into the script text.
    - Create UDF Implementation – Context sensitive. The tool is used to quickly create a template implementation of a user-defined function. Only displayed when the context menu's opened at a user-defined function call site and the string token at the menu's location isn't a known UDF's identifier.
    - Rename Variables – Allows the current script's variables to be renamed without losing their indices.
- Copy – Copies the string token at the menu's location.
- Paste – Pastes the contents of the clipboard at the caret's location.
- Find – Displays the Find/Replace dialogue for the string token at the menu's location.
- Toggle Comment – Toggles the comment status of the selection/current line.
- Toggle Bookmark – Sets bookmark on the current line. Bookmarks are saved with the script text, as metadata.
- Add Message – Adds a notification message for the current script. Messages are saved with the script text and displayed every time their parent script is loaded into a workspace. They can be deleted by double clicking on their respective items in the Message List list-view.
- Look-up on the Wiki – Searches for the string token on the Elder Scrolls Construction Set Wiki.
- Look-up in the OBSE Docs – Searches for the string token in the OBSE Command Documentation.
- Look-up on Google – Searches for the string token on Google.

- Developer Page – Context specific. Opens a developer specified link, if any, in the default web browser. Only displayed for identifiers of script commands from 3rd party OBSE plugins that interoperate with the CSE.
- Jump to Object/Quest/Function Script – Context specific. Opens the script associated with the identifier at the menu's location, if any. Only displayed for identifiers of scriptable objects and scripts themselves.

  The *Navigate Forward/Backward* buttons in the main toolbar can be used to switch between workspaces after successive jumps.
- Open Import File – Context specific. Opens the pre-processor import file. Only displayed for the IMPORT directive.

And it keeps coming! The code editor offers even more:

- Syntax Highlighting.
- Code Folding.
- Brace Matching.
- IntelliSense.
- Code Snippet Manager.
- Script Validation.
- Auto-Recovery.
- A Pre-processor.

IntelliSense

IntelliSense is the name given to the script editor's implementation of auto-completion. It provides a convenient method to access script commands, local variables, remote scripts and their variables, user defined functions, game settings, global variables, quests and pretty much every other object one can create in the CS.

IntelliSense displays its pop-up list as one types in code, filtering its items to reflect the changes made to the token being typed. Once displayed, the Up and Down arrow keys can be used to navigate the suggestion list. The currently selected suggestion can be inserted at the caret location by pressing the Tab key, while the Escape key closes the pop-up.

The suggestion list may contain items of the following types:

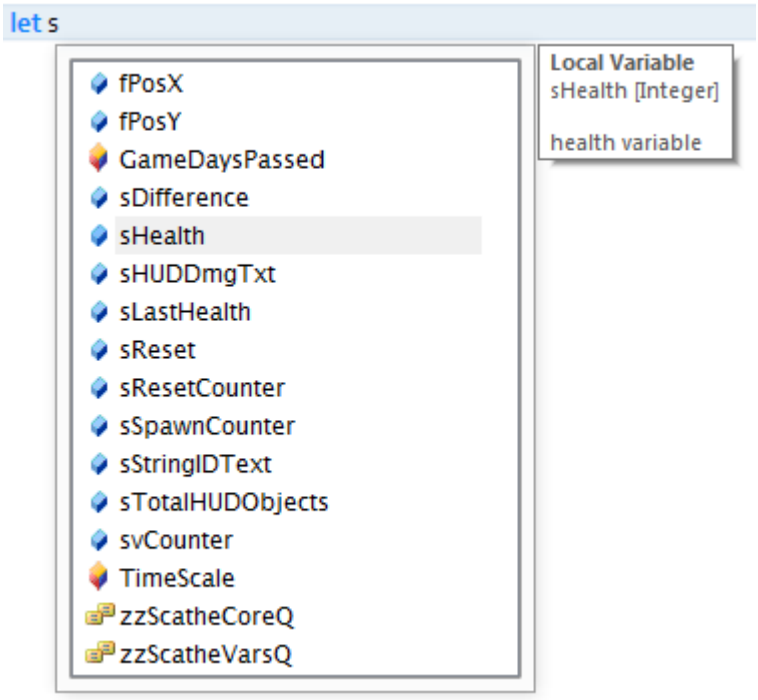
- Commands – Descriptions include command alias, description, number of parameters, command source and return type.
- Variables – Descriptions include their type and any comments that following their declaration.
- Quests – Descriptions include the name field of the quest and the editorID of the quest script, if any.
- User Defined Functions – Consider the following UDF script:

```
Scn SampleUDFScript

; this is an UDF script
; some text – foo
; more foo

Short sArg1         ; Some arg
Float fArg2         ; Another arg

Begin Function {sArg1 fArg2}
     Let sArg1 := 111
     SetFunctionValue sArg1
End
```

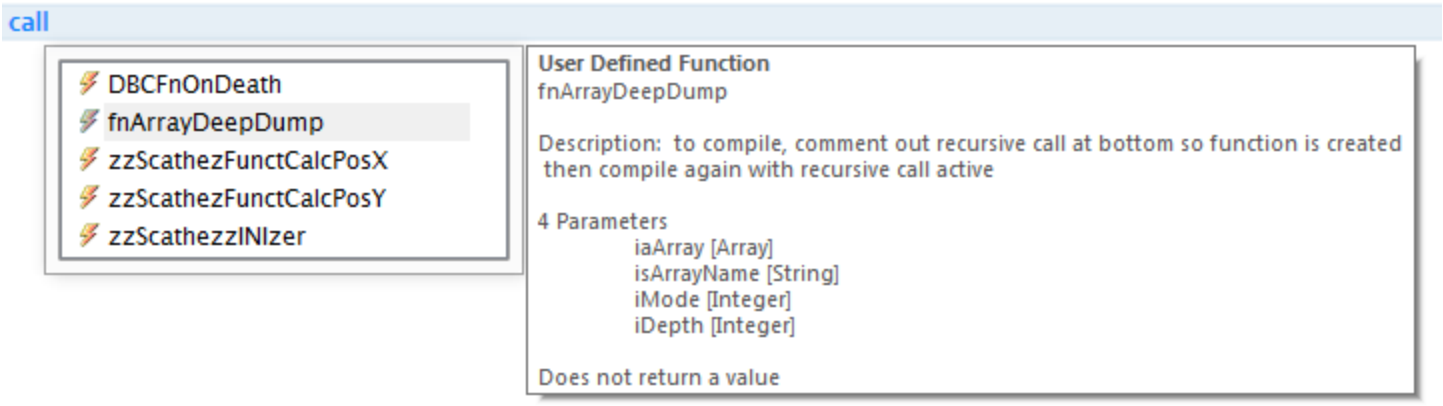
The description will include the comment text between the script name declaration and the first local variable's. Arguments are treated as variables and enumerated. And finally, the return type of the UDF is stated.

IntelliSense supports the following context specific triggers:

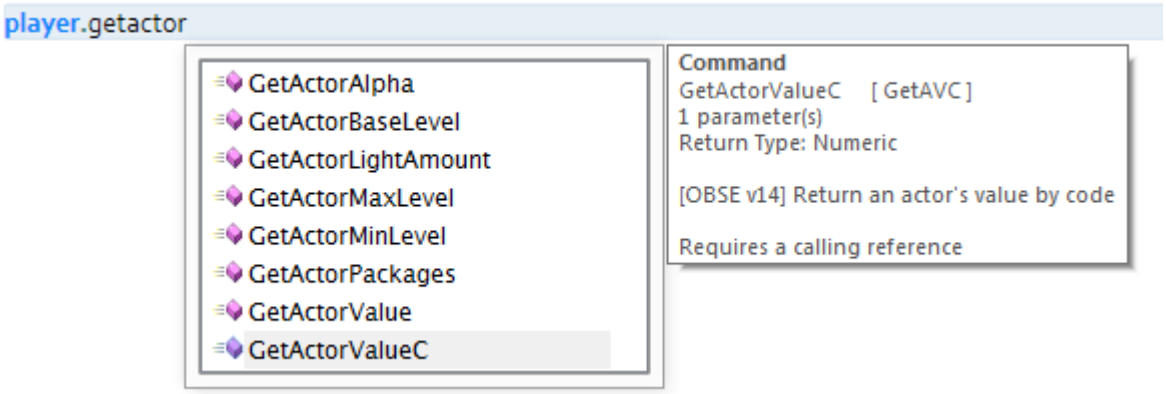
- Set or Let – Suggests local variables, global variables and quests.



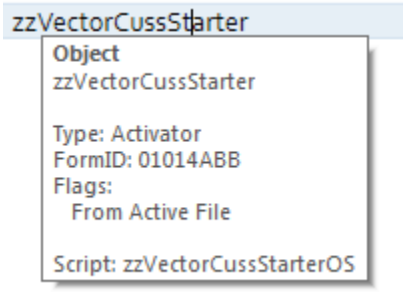
- Call – Suggests user defined functions.



- Dot (.) – Suggests remote variables (variables in the script attached to the first operand) and script commands that require a calling reference.

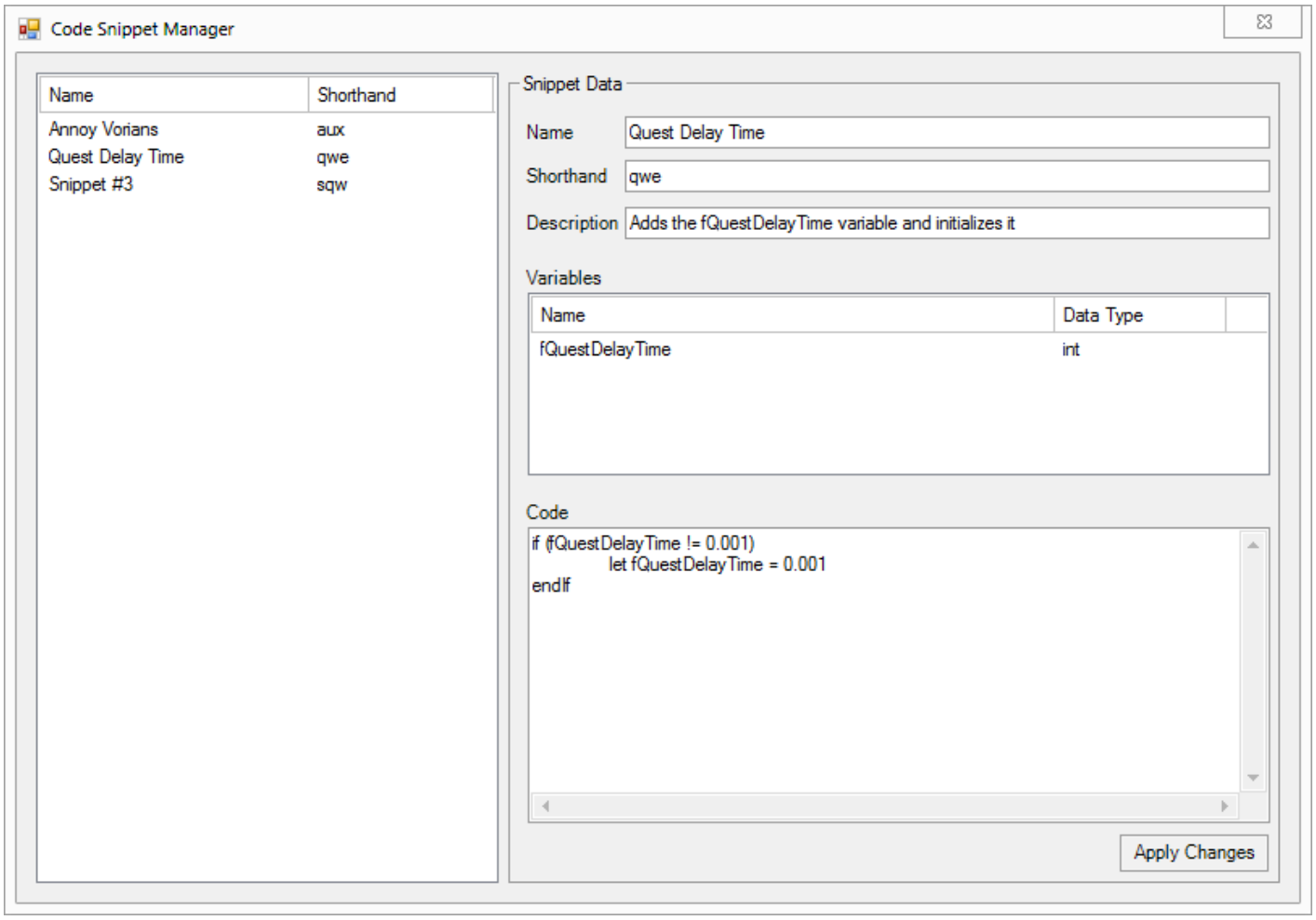


IntelliSense also allows for quick access to an object's properties in the form of tool-tips. Hovering the mouse



pointer over a valid identifier will bring up a tool-tip describing the object using it.

Code Snippet Manager

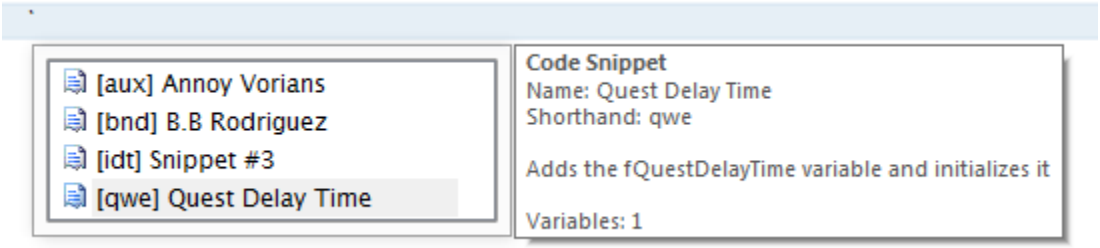


The Code Snippet Manager, as its name suggests, is used to manage create, edit and manage various snippets of code that can be directly inserted into a script, at caret location. Snippets have the following fields:

- Name – The name given to the snippet.
- Shorthand – The snippet's alias.
- Description – The snippet's description. This field is optional.
- Variables – A list of variables that the snippet may use.
- Code – The actual code that is to be inserted.

Right clicking on the snippet list-view brings up its context menu which can be used to add new snippets or remove existing ones. Similarly, the variable list-view's context menu can be used to add/remove variables from the active snippet.

Inside the code editor, the tilde (`) key is used to bring up the list of available snippets.



Script Validator

The script validator catches errors that the vanilla script compiler doesn't, namely:

- Invalid block types for non-object scripts.
- Script name re-declarations.
- Superfluous expressions in commands.
- Nested variable declarations.
- Variable re-declarations.
- Unreferenced variables.

The token parser expects operators, operands and function arguments to be delimited by one of the following characters: `., (){}[]\t`

Pre-processor

The script editor implements a pre-processor engine that allows users to use various pre-processor directives, not unlike the C pre-processor. All directive declarations or definitions need to be represented as comments. Pre-processor directives are grouped in two: single and multi-line directives. Single line directives do not exceed a line of code in the text editor. Such directives use the '#' character as their prefix. Multi-line directives, on the other hand, encompass multiple lines of code and must be prefixed with the '@' character. The multi-line argument/value needs to be enclosed in curly braces. Some directives support no more than one encoding type.

For example:

```
;#DEFINE MACRO_FOO "FOO~POO"

;@IF (MACRO_FOO != 123.222 || (MACRO_FOO < 10 && MACRO_FOO > 4.2))
;{
;   PRINT "MACRO CONDITION EVALUATED AS TRUE!"
;}
```

Define – Defines a pre-processor macro, similar to C's. Macro identifiers can only contain alpha-numeric characters and underscores, and are case sensitive. They must be delimited with one of the following chars to be recognized: `., (){}[]\t.` Macro values themselves can contain any character. They can be used in

any context as the pre-processor simply replaces the macro identifier with its value before compilation. For instance,

```
;#define _DEBUG 1

if _DEBUG
print "This message will be printed if _DEBUG is set to a non-zero value"
endIf

;@define PrintMESSAGEString
;{
; print “MessageOne!”
; print “MessageTwo!”
; ; comment
;}

if zzQuest.Var == 1
PrintMESSAGEString
endif
```

The pre-processor also allows the use of accessory operators during macro expansion. These operators are placed before macro identifiers and perform special operations on the values of macros. The following are the supported accessory operators:

- Stringize (#) – Wraps the macro’s value in double quotes.

```
;#define STRIZE Help
print #STRIZE                          ; expands to “Help”
```

Import – Allows external text to be inserted into scripts, similar to #include in C’s. The text files to be inserted must be placed inside the script editor’s pre-processor resource folder (detailed further below). Consider the following example,

```
<Pre-processor Resource Directory>\TestSnip.txt
    float  fquestdelaytime
    short  doonce
    long   goldvalue

Regular Script zzTestQS
    scn zzTestQS

    ;#import “TestSnip”

    begin gamemode
           print “foo”
    end
```

```
Pre-processed Script zzTestQS
    scn zzTestQS

    float  fquestdelaytime
    short  doonce
    long   goldvalue

    begin gamemode
           print “foo”
    end
```

The Import directive is recursive, so imported scripts and snippets can have their own preprocessor directives. It does not support multi-line encoding.

Enum – Defines an enumeration (enum for short). An enumeration is basically a single line definition that allows multiple macros to be defined in order. Enum items can only have numeric values. They need not be continuous i.e., an item may be declared without an initialization value, in which case it will be assigned one more than the value of its predecessor. The default value starts with 0. The syntax for an enumeration is as follows:

```
;#ENUM ENUM_NAME {ITEMA=VALUE ITEMB=VALUE ...}
;@ENUM ENUM_FOO
;{
; ITEMA=VALUE
; ITEMB
;}
```

Enum items can be used like any other macro, by their identifier.

If – Controls compilation of portions of the script. If the expression written (after the directive identifier) evaluates as true, the code group following the directive is retained in the translation unit.

```
;#define DebugLevel 1
;#define foo “String”
;#define bar 4.5

;@if DebugLevel > 1 && DebugLevel < 3
;{
;   print “Log Level A: Debug Message”
;}

;@if ((DebugLevel <= 12) || ((foo == “String”) && foo != 4.25))
```

```
;{
;   print "Log Level X: Debug Message"
;   if eval (Octopi.tentacles == "CSE > Skyrim")
;          player.kill
;   endif
;}
```

The condition expression can only include macro identifiers and constants or literals. The directive supports the following relational operators, which are evaluated in their default order of their precedence.

- Equality [==]
- Less than or equal [<=]
- Greater than or equal [>=]
- Inequality [!=]
- Greater than [>]
- Less than [<]

In addition to the above, the logical operators AND [&&] and OR [||] are allowed in expressions. Parentheses may be used to override the default precedence. A macro's definition can be tested by using it in the condition expression without any operators.

```
;#define bar 1

;@if bar
;{
;   print "This line will be parsed by the script compiler"
;}

;@if foo
;{
;   print "This line will not be parsed as 'foo' has not been defined"
;}
```

Any text files placed inside the standard directives folder will be parsed before each preprocessor operation. Multi-line blocks can contain directive declarations – they will be expanded automatically when the parent directive is. The number of passes the preprocessor makes can be configured from the Preferences window.

Shortcut Keys

The CSE Editor adds a number of counter-intuitive shortcut keys for its various functions to aid the lazy scripter. And tejon.

Shortcut Key	Action

CONTROL + T	New workspace
Middle Mouse Click on a Tab	Close workspace
CONTROL + TAB CONTROL + PAGE DOWN	Switch to the next workspace
CONTROL + SHIFT + TAB CONTROL + PAGE UP	Switch to the previous workspace
CONTROL + 1…9	Switch to the nth workspace
CONTROL + New Button	Open a new workspace and initialize a new script
SHIFT + New Button	New editor window
CONTROL + Open Button	Open a new workspace and display the Open Script dialogue
CONTROL + Q	Toggle comment
CONTROL + O	Open script
CONTROL + S	Compile and save script
CONTROL + SHIFT + S	Save all open workspaces
CONTROL + D	Delete script
CONTROL + ALT + LEFT	Previous script
CONTROL + ALT + RIGHT	Next script
CONTROL + N	New script
CONTROL + F4	Close script
CONTROL + B	Toggle bookmark
CONTROL + ENTER	Show IntelliSense interface
ESCAPE	Hide IntelliSense interface Clear find result indicators

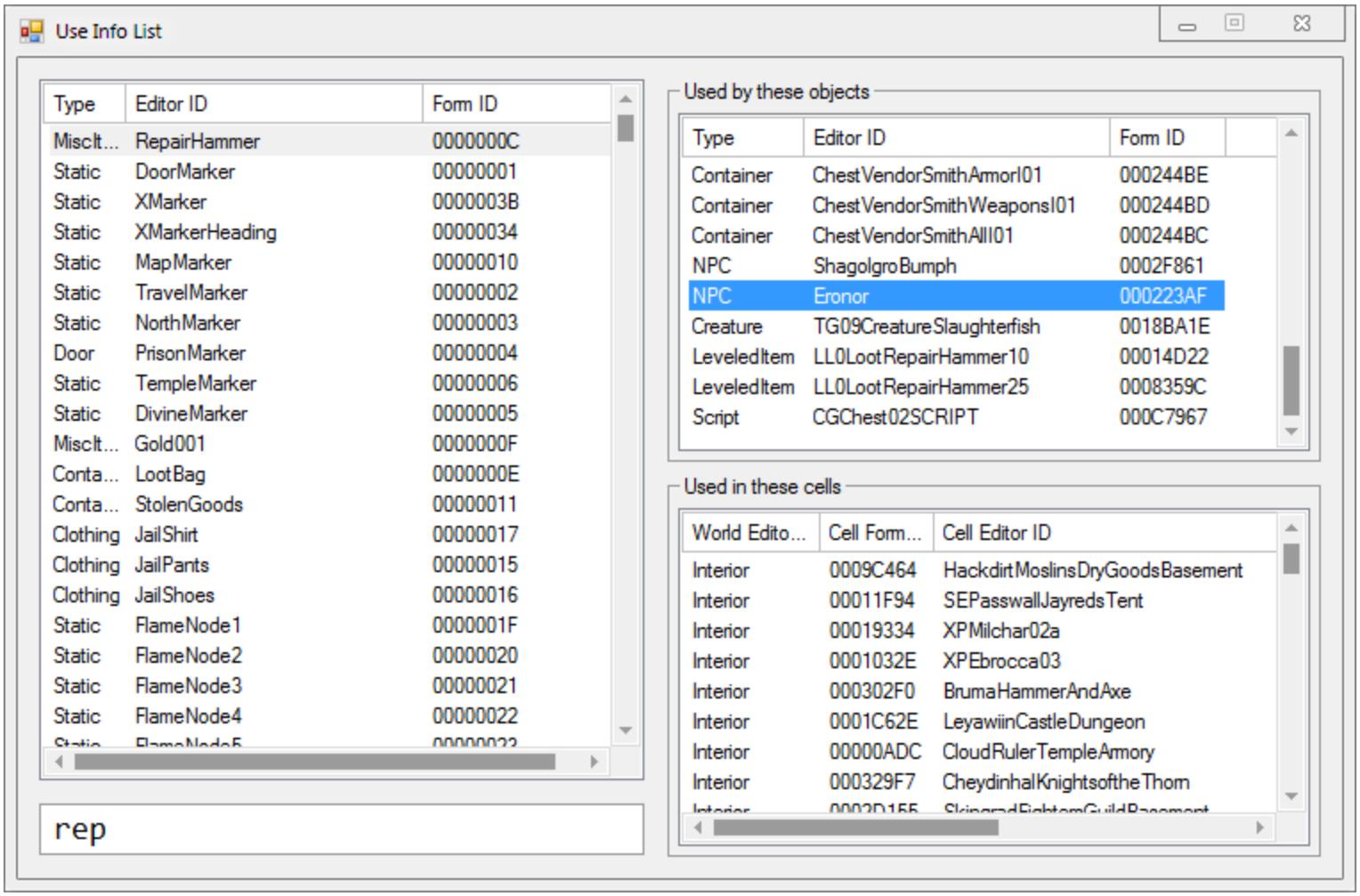
CONTROL + UP	Move current line up
CONTROL + DOWN	Move current line down
CONTROL + F	Find
CONTROL + H	Replace
CONTROL + G	Go to line
CONTROL + E	Go to offset
CONTROL + Left Mouse Click on Scriptable Object Identifier	Jump to Script
F1 (In the Select Script dialogue)	Use info report for the selected script

Resource Location

As with all Bethesda Game Studios Editor Extender-related resources, the script editor's resources are to be saved inside the "Data\BGSEE" directory.

- Data\BGSEE\Script Editor\Preprocessor – Preprocessor resources such as importable snippets are saved in this folder.
- Data\BGSEE\Script Editor\Preprocessor\STD – Standard preprocessor directives are saved in this folder.
- Data\BGSEE\Script Editor\Snippets – Code snippets are saved to this folder.

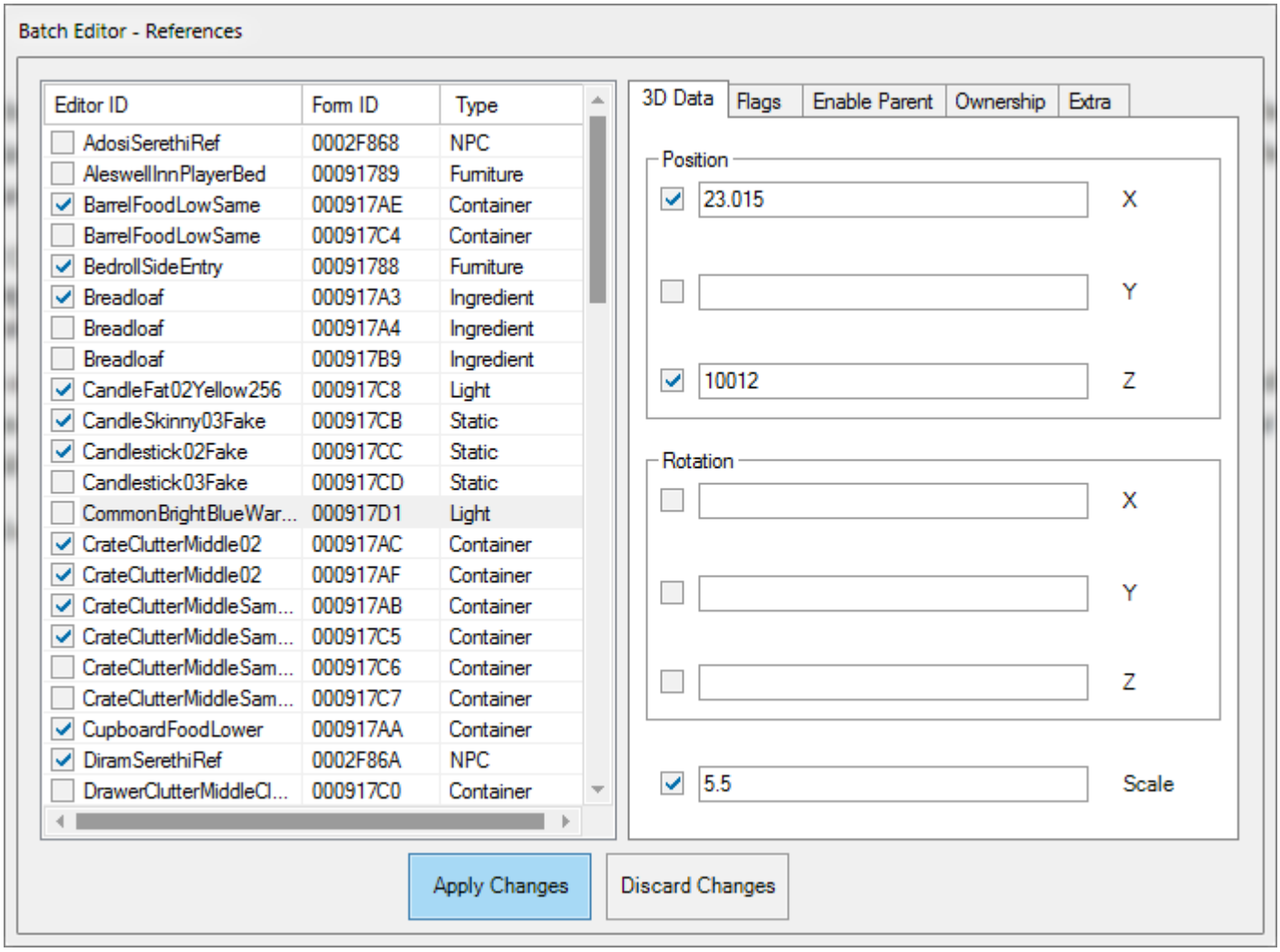
Centralized Use Info Listing



The Use Info listing tool is basically a conglomeration of the use reports of every loaded record in the CS. It allows easy look up of cell and object use lists through its centralized listing. Furthermore, every item in the list can be edited directly by double clicking it. The textbox at the bottom is used to filter the form list by Editor ID and Form ID. Every form type, save MGEF and GMST, are listed and tracked.

It can be accessed from the “View” menu.

Batch Reference Editor



The Batch Editor for references lives up to its name in most areas of the batch editing of references. Editable references must be present in the loaded cell. When initialized, the selected objects will automatically be checked in the editor's object list – only checked objects will be modified. The editor attempts to emulate the vanilla reference properties dialogue as seen above. It can edit attributes of the following groups:

- 3D Data – Includes position, rotation and scale data.
- Flags – Includes flags for persistence, initially disabled state and visible when distant [VWD].
- Enable Parent, Ownership and Extra – Similar to the vanilla reference property dialogues.

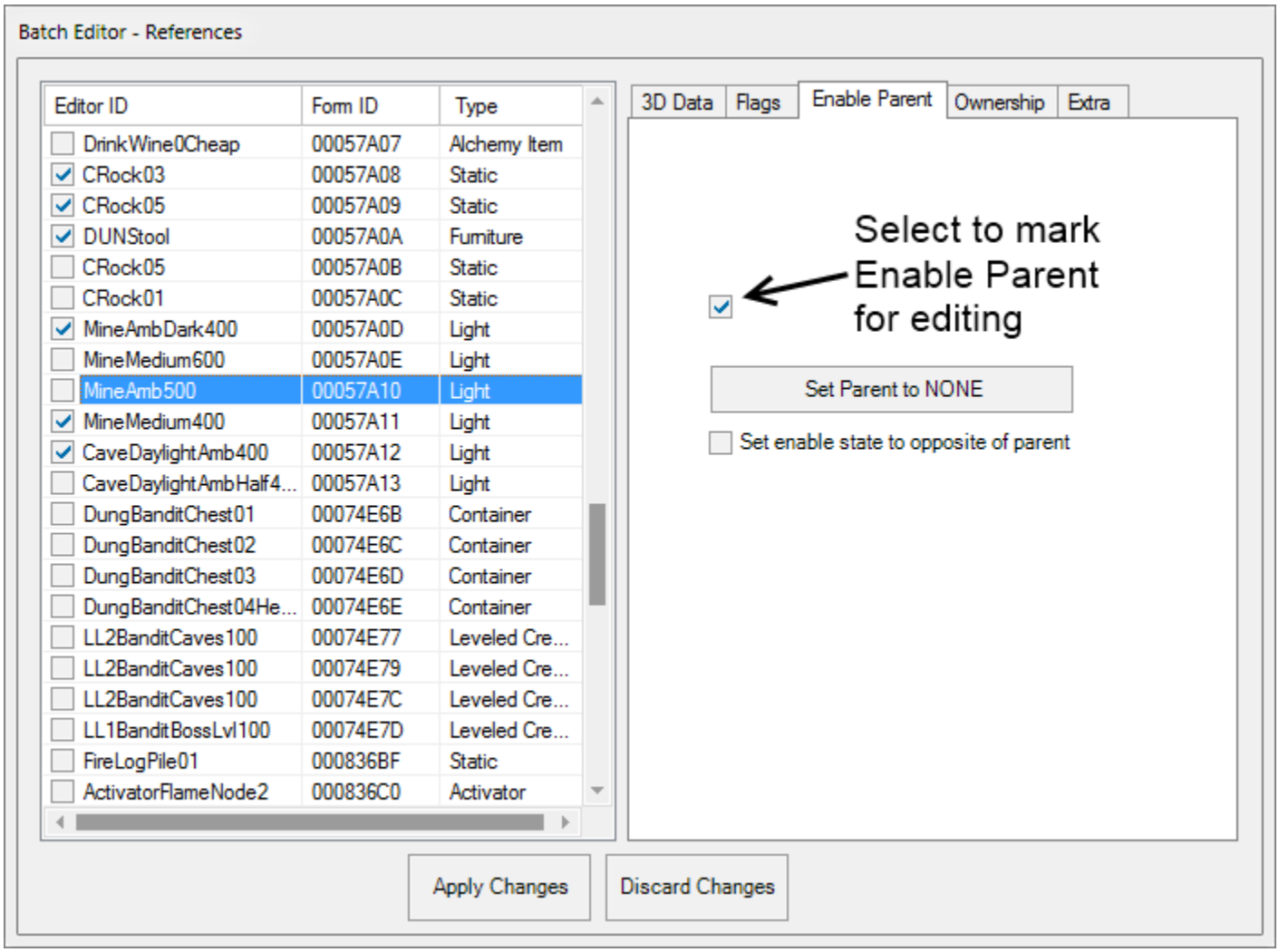
The last 3 groups are applied selectively, to the objects that have the respective attributes and extra-data. Each modifiable attribute has a check-box next to it which indicates the enabled state of the corresponding attribute – The attributes modified are those with an 'Enable' check in their companion check-box

It can be accessed from the render window's context menu.

Usage

The Batch Editor operates on the selected references. References may be selected in the Render Window first and those references will be marked for editing when the Batch Editor window is opened. Alternatively, open the Batch Editor and select the references for editing by clicking on them in the left panel.

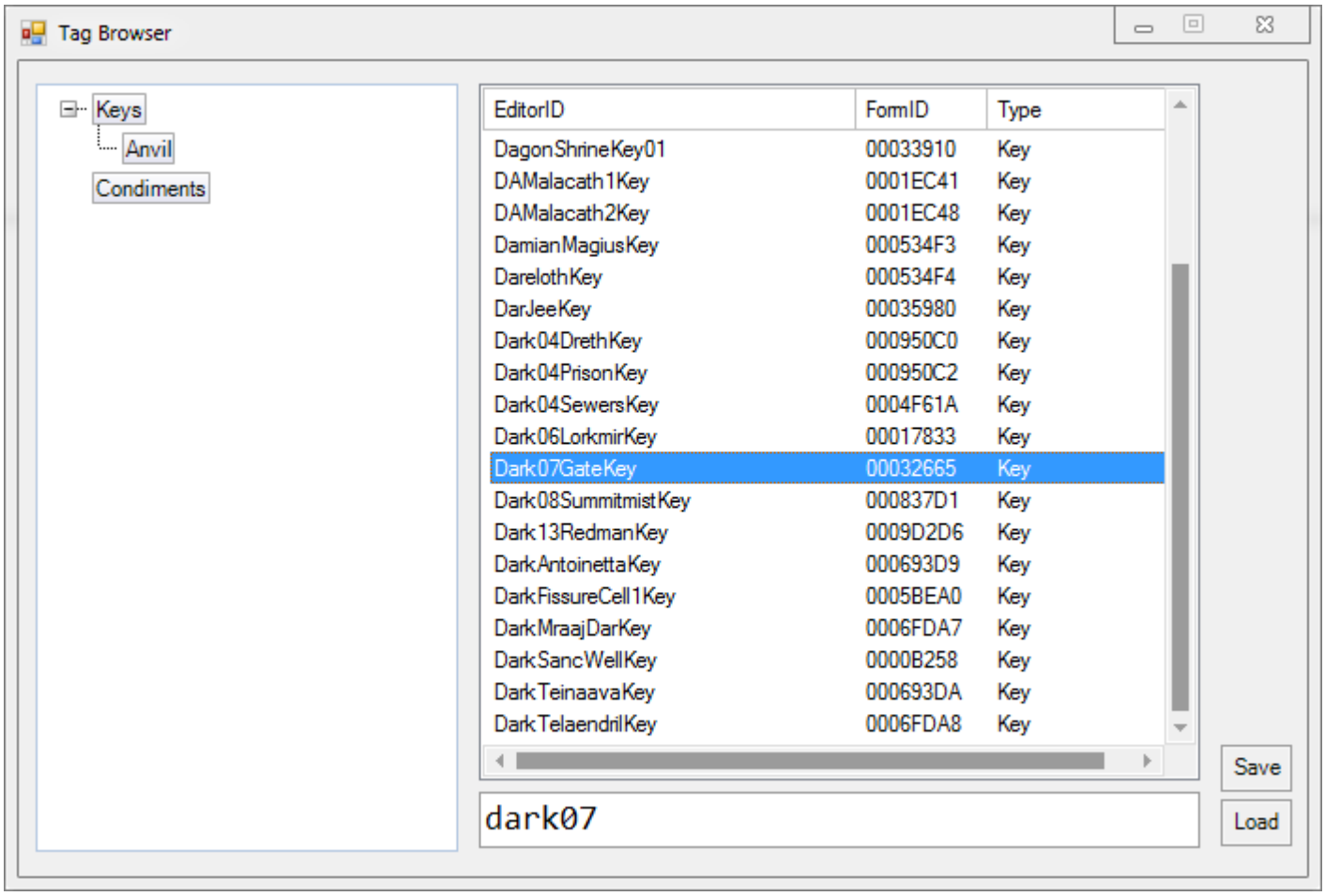
Once you have selected the objects, you will find the various editing options organized via tabs in the right panel. There are two components for every option: you must select the option for editing and then input the data value or select the value. It has been organized this way so that clearing the previous setting is also an option. The Enable Parent tab is a prime example. If you wish to edit this setting, you need to select the checkbox above the settings.



If you wish to clear the parent setting, leave the button as Set Parent to NONE and click *Apply Changes*. If you wish to set the parent, then click on the *Set Parent to NONE* button. It will open the Choose Reference dialogue. Select a parent reference. Now the text of the button will change to the selected reference. Click

the *Apply Changes* button to change the parent. Click the *Discard Changes* button to close the Batch Editor without making any changes.

Tag Browser



The Tag Browser allows the user to attach arbitrary tags to any record. Tags can be nested with the use of drag-drop operations, renamed by (slowly) double clicking on their nodes. A record can be allocated multiple tags, but each tag may only contain a single instance. Records can be drag-dropped into the record list of the active tag. The record list behaves similar to the object window – records can be double clicked for editing and drag-dropped into the Render Window for reference instantiation or indeed any other target location that allows dragging and dropping. The textbox can be used to search for specific items in a tag's record list. Tags and tagged records may be added or removed through the context menu. Tag hierarchies can be saved to disk using the Save option; the Load option is subsequently used to load a saved hierarchy. Invalid or non-existent records will be removed during a load operation. The textbox in the bottom can be used to find items in the record list.

It can be accessed from the View menu.

Usage

To use the Tag Browser, start by defining a tag in the Tag Browser window, accessed from the View menu. You may add several tags if desired. Now click on a tag to make it active. It will have a thicker border when active. Close the Browser window. You may save your tags to a file, but that is not necessary.

Now find an object that you wish to tag. Right-click on it and select Add to Active Tag from the context menu. You can also tag an object by dragging it to the left pane of the Tag Browser window, as mentioned above.

Plugin Inter-Op API

The Construction Set Extender provides a public API for 3rd party OBSE plugins that modify the editor. The latter can avail some of the new features the CSE introduces to the Construction Set.

The API and related documentation can be found in the CSE's source code repository, saved inside the 'CSEInterfaceAPI.h' header file. A link to the aforementioned repository can be found in the readMe file.