

Construction Set Extender

shadeMe

Version 10.0

Introduction.....	3
Enhancements	3
Handling of Plugins and Masters.....	3
Creation and Modification of Master Files:	3
Modification of Master File Header Data:	3
Removal of the Need for Mod De-isolation:.....	3
Saving Plugins as ESM Files:	4
Save As Option:	4
Timestamp Preservation:.....	4
Automatic Backup Creation:.....	4
Work-flow Improvements.....	4
Start-up Options:	4
Workspaces:	4
Setting up a Custom Workspace.....	5
Console:	5
Enhanced Asset Selection:.....	7
Context Menu Tools:.....	8
Batch Copy Eyes/Hair:.....	9
FaceGen Editing:.....	9
Rearrange Effect Items:.....	9
LOD Texture Generator:	9
Live Change Log:.....	10
Script Compiler:.....	10
Quick Look-up Editor ID:	10
Multiple Preview Windows:	10
Filterable Object/Form Lists:.....	11
Global Script Creation:	11
Enhanced Response Editor:	11
Object Window:	13
Cell View Window:	13
Render Window:.....	14
On-Screen Display:.....	14

Key Bindings:	15
New Tools:	15
Tweaks:	17
Render Window Context Menu:	18
Global Clipboard:	18
Global Undo Stack:	19
User Interface Improvements	19
Active Form Sorting:	19
Active Form Colourization:	20
Override Form Colourization:	20
Form Enumeration Filters:	20
Main Editor Windows' Visibility:	20
Taskbar Visibility:	20
Search and Replace:	20
Enhanced Find Text:	20
Safer Modification of List View based Records:	20
Launch Game:	21
Result Script Editing:	21
Time of Day Slider:	21
Editor IDs in Edit Dialogue Titles:	21
Improved Dialogue UI's:	22
Custom Color Theme/Dark Mode:	23
Trifles	24
Performance Improvements:	24
Fast Exit:	24
Icons with MIP maps/Texture Size Limitations:	24
Auto-loading BSA Archives:	24
Integer-prefixed Editor IDs:	24
Idle Animation Tree Initialization:	24
Archived Sound File Sampling:	24
Retroactive "nVidia Fog Fix":	24
Circular Leveled List Detection:	25
3 rd Party Tool Launcher:	25
Achievements:	25
Last Chance "Panic Save" Handler:	25
Persistent Window Locations:	26
Purge Loaded Resources:	26
Vanilla Bug Fixes	26

New tools	32
Coda Script	32
Script Editor.....	32
IntelliSense.....	38
Code Snippet Manager	41
Script Validator	42
Pre-processor.....	42
Sync Scripts To Disk.....	46
Shortcut Keys	47
Resource Location	49
Centralized Use Info Listing	50
Tag Browser	50
Usage	52
Object Palette.....	53
Working with OPALs.....	54
Object Prefabs.....	56
Plugin Inter-Op API.....	57

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. The CSE Preferences dialog can be accessed from the File main menu.

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).

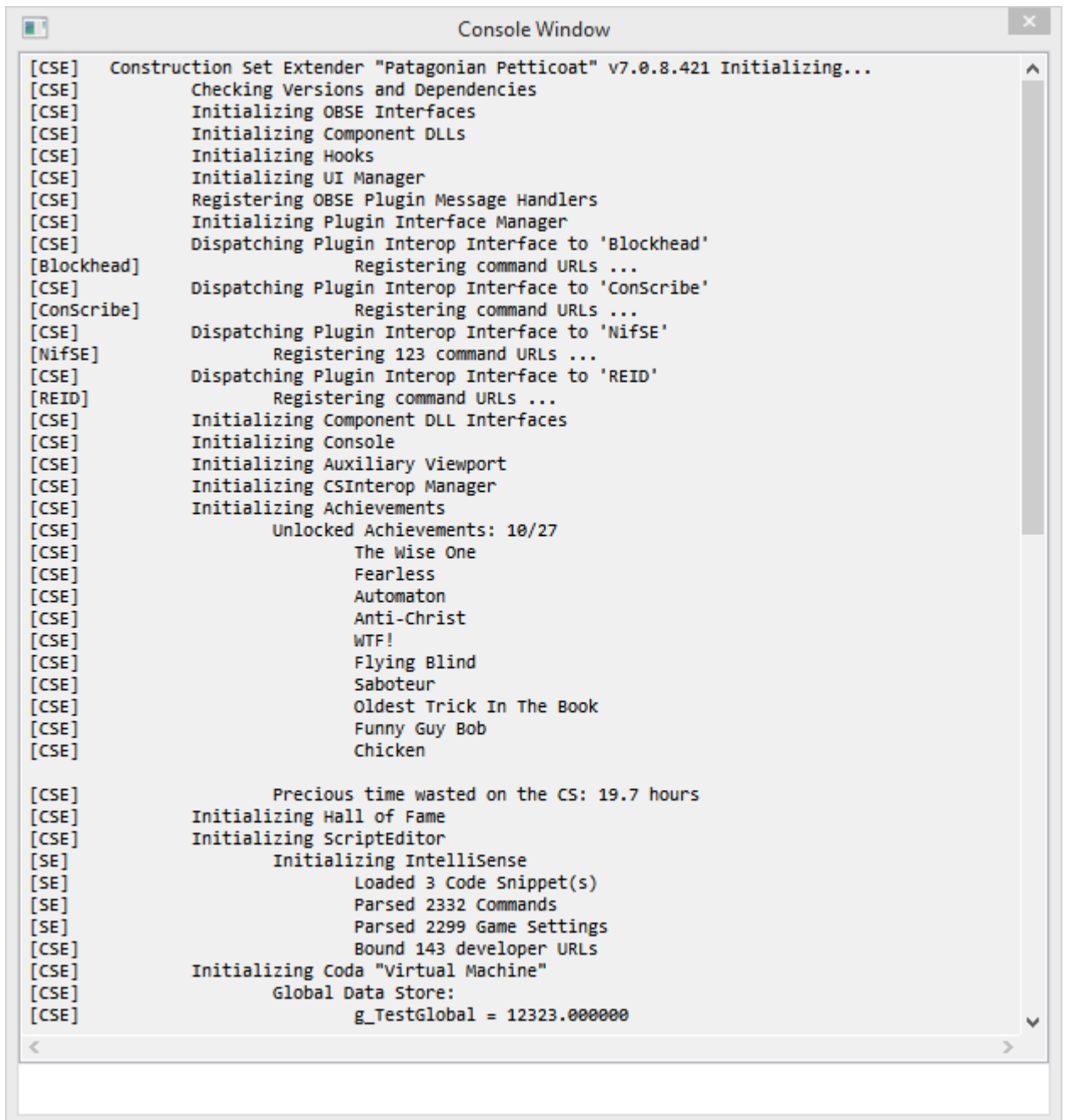
Console:

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.



```

Console Window

[CSE] Construction Set Extender "Patagonian Petticoat" v7.0.8.421 Initializing...
[CSE] Checking Versions and Dependencies
[CSE] Initializing OBSE Interfaces
[CSE] Initializing Component DLLs
[CSE] Initializing Hooks
[CSE] Initializing UI Manager
[CSE] Registering OBSE Plugin Message Handlers
[CSE] Initializing Plugin Interface Manager
[CSE] Dispatching Plugin Interop Interface to 'Blockhead'
[Blockhead] Registering command URLs ...
[CSE] Dispatching Plugin Interop Interface to 'ConScribe'
[ConScribe] Registering command URLs ...
[CSE] Dispatching Plugin Interop Interface to 'NifSE'
[NifSE] Registering 123 command URLs ...
[CSE] Dispatching Plugin Interop Interface to 'REID'
[REID] Registering command URLs ...
[CSE] Initializing Component DLL Interfaces
[CSE] Initializing Console
[CSE] Initializing Auxiliary Viewport
[CSE] Initializing CSInterop Manager
[CSE] Initializing Achievements
[CSE] Unlocked Achievements: 10/27
[CSE] The Wise One
[CSE] Fearless
[CSE] Automaton
[CSE] Anti-Christ
[CSE] WTF!
[CSE] Flying Blind
[CSE] Saboteur
[CSE] Oldest Trick In The Book
[CSE] Funny Guy Bob
[CSE] Chicken

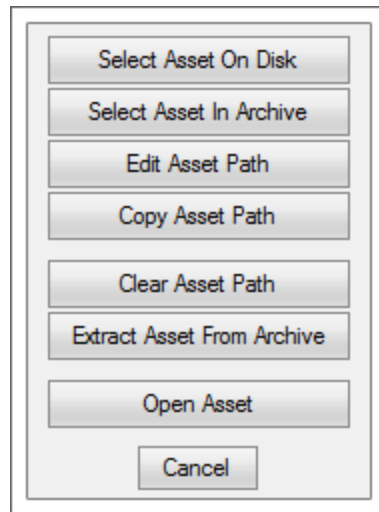
[CSE] Precious time wasted on the CS: 19.7 hours
[CSE] Initializing Hall of Fame
[CSE] Initializing ScriptEditor
[SE] Initializing IntelliSense
[SE] Loaded 3 Code Snippet(s)
[SE] Parsed 2332 Commands
[SE] Parsed 2299 Game Settings
[CSE] Bound 143 developer URLs
[CSE] Initializing Coda "Virtual Machine"
[CSE] Global Data Store:
[CSE] g_TestGlobal = 12323.000000

```

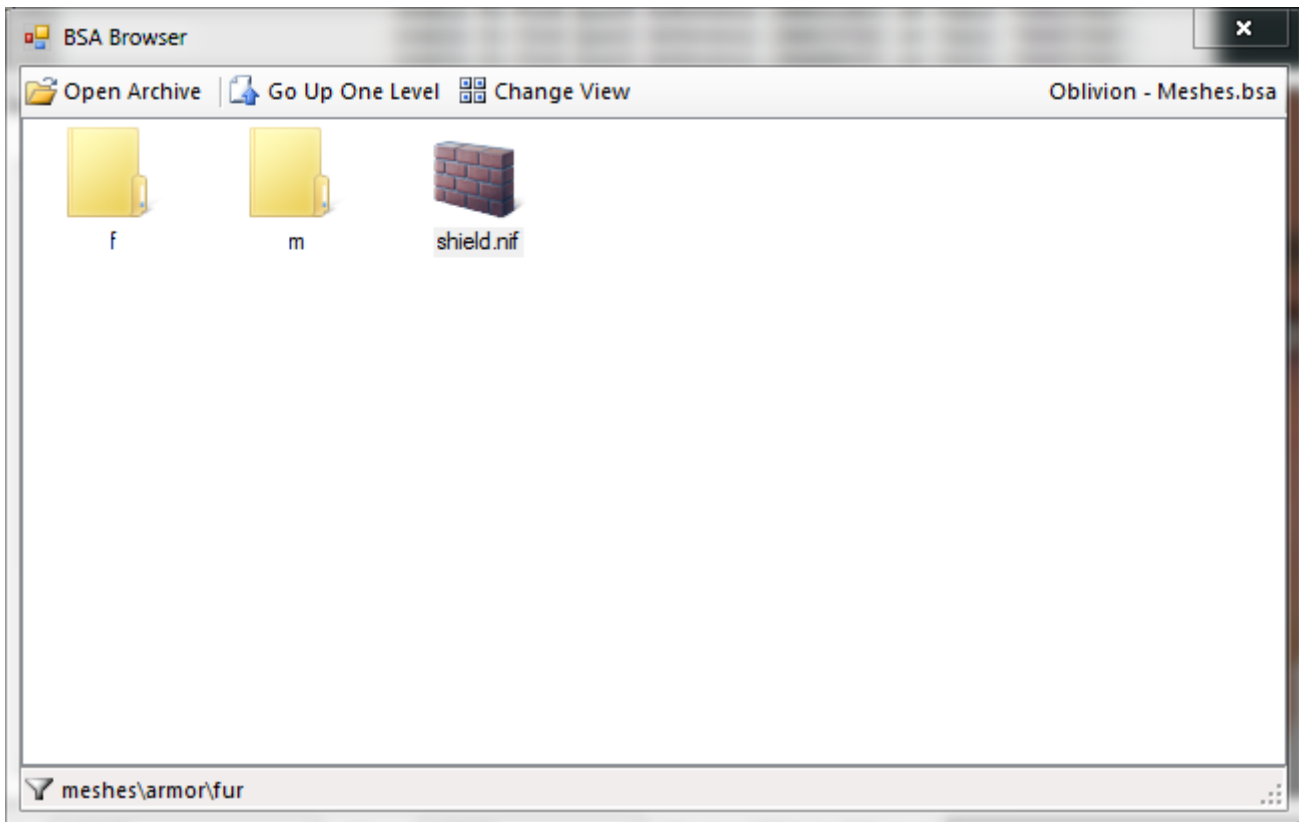
The console has the option to bundle messages that fall under pre-defined categories. While the majority of informational and debug output is directly shown in the console's main/default message channel, certain tools implemented by the CSE can choose to route their messages to a specific message channel to keep from cluttering the main channel. These channels can be accessed through the console's context menu under the 'Contexts...' submenu.

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 viewed in the live change log console message channel. If automatic plugin backups are enabled, the change log pertinent to that plugin's session is saved along with it. Logging is disabled by default and can be enabled in the CSE Preferences dialog.

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.

Multiple Preview Windows:

The editor now supports previewing objects in more than one preview window. This option can be disabled in the View menu.

Filterable Object/Form Lists:

A “Filter” field has been added to most dialogs that contain a listview of forms/records. It can be used to filter the listview’s contents by EditorID, FormID, Name or Description. Search strings can either be regular expressions or simple substrings. Right clicking on the “Filter” label displays options to control the search behaviour.

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.

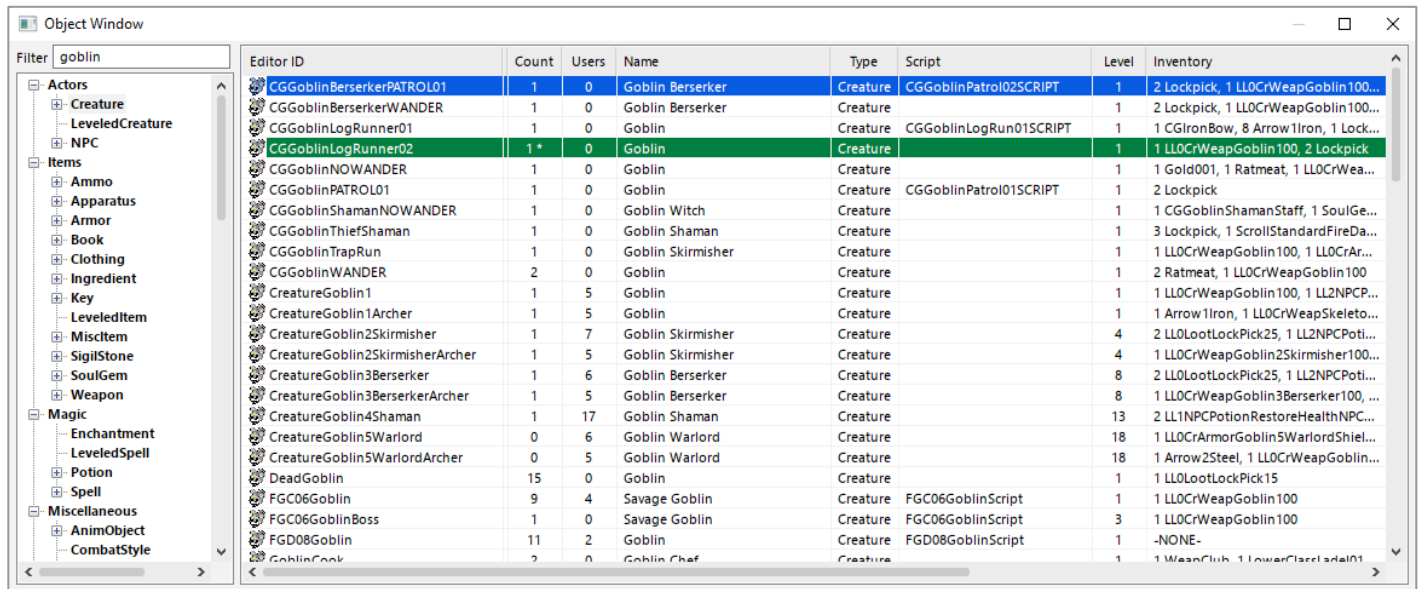
Enhanced Response Editor:

The response editor has been modified to provide a more streamlined interface. The voice recording tool has been removed, given its general bugginess 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.

Object Window:

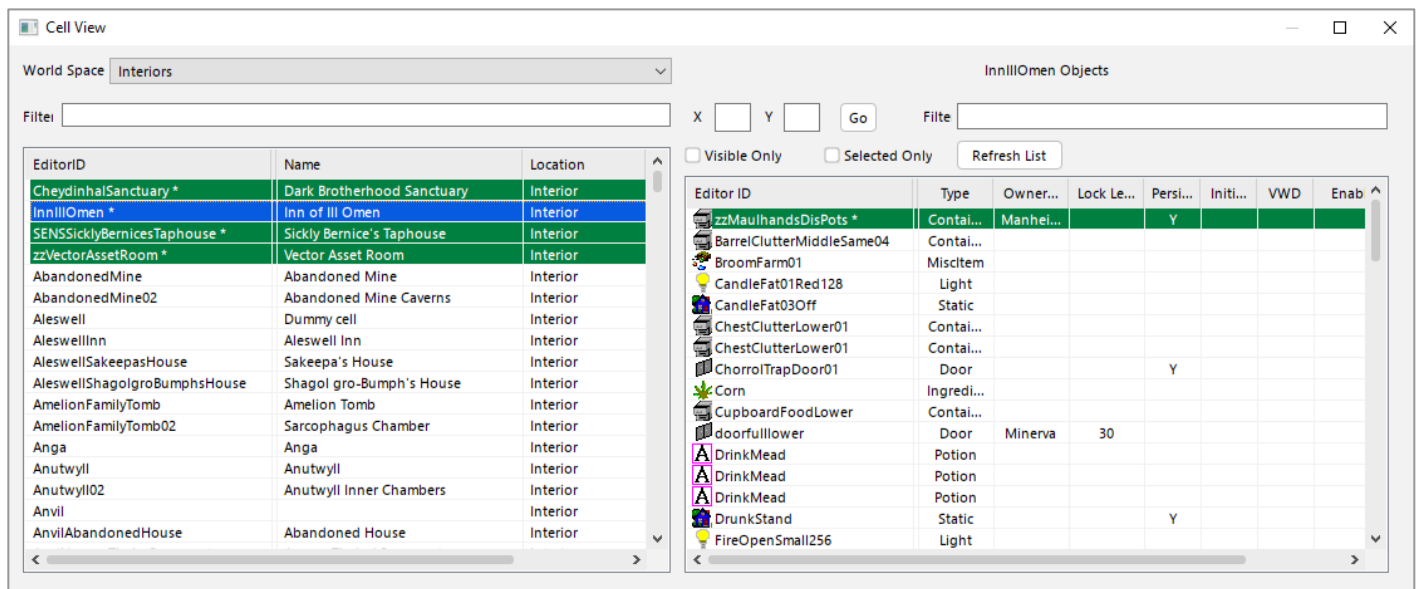
The following improvements have been made to the object window:



- Filter Objects – The object list can now be filtered with a search string.
- Multiple Instances – Additional Object Windows can be opened from the View main menu (Spawn Extra Object Window).

Cell View Window:

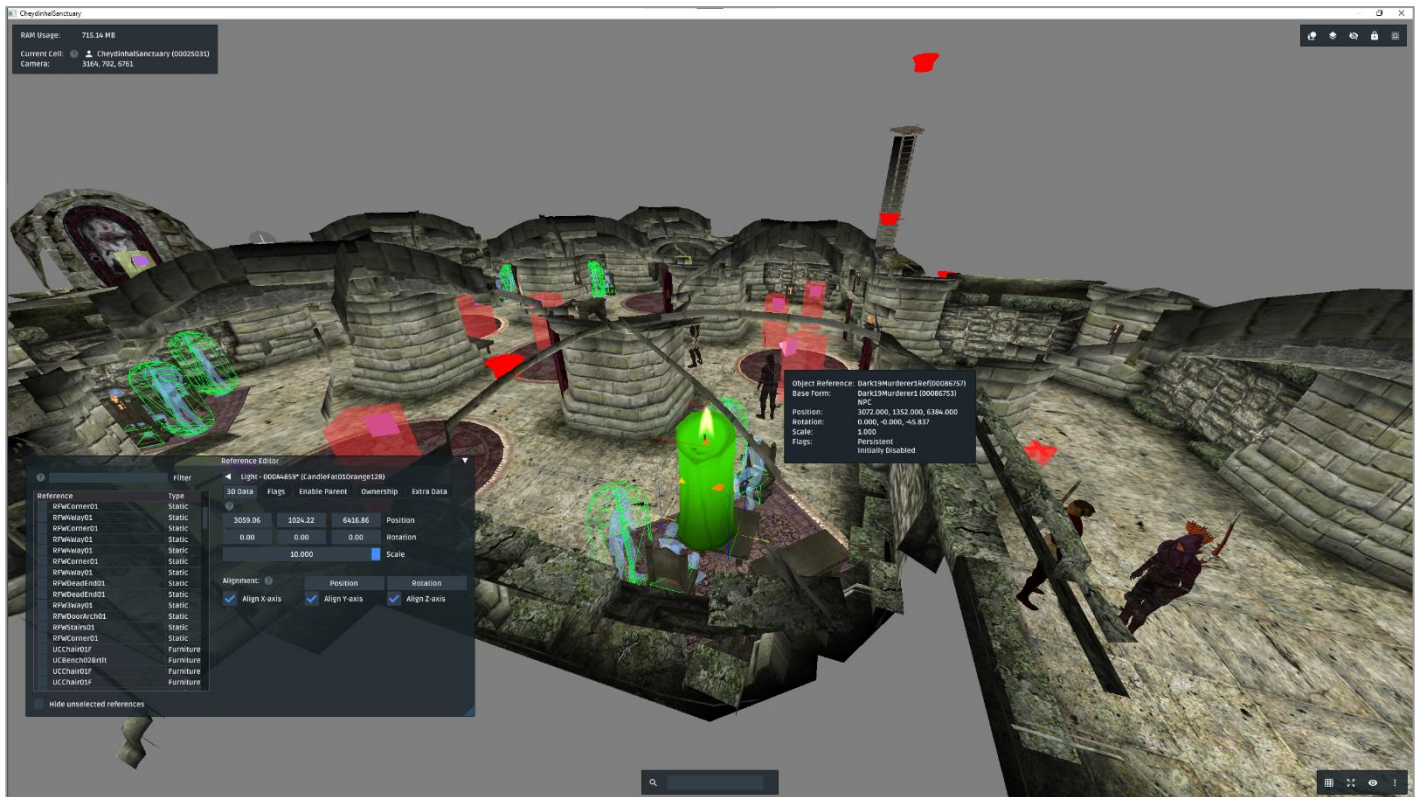
The following improvements have been made to the cell view window:



- Filter Cells and References – The cell and references lists can be filtered with a search string. References can additionally be filtered by their visibility and selection.

- 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 render window has been tweaked and overhauled to improve the general modding experience.

On-Screen Display:

The most visible change to the render window comes in the form of an on-screen display. It acts as a unified interface for the new render window tools added by the CSE, in addition to providing an overview of the current scene.

The OSD consists of two types of elements: Overlays and Panes. Overlays are static elements that display specific information. They have little to no user-interactable elements. The OSD supports the following overlays:

- Info Overlay – Positioned at the top-left corner of the viewport, the info overlay displays salient information about the current scene and RAM usage.
- Mouse Tooltip – Positioned next to the mouse pointer, it displays information about the reference or pathgrid point that is currently under the pointer.

Panes are windows that contain interactive widgets. They provide a user-interface for certain render window tools. The OSD supports the following panes:

- Reference Editor – Provides an interface to modify the basic properties of the current render window reference selection. Functions as a batch reference editor. This window can be moved around by dragging it.
- Search – Allows the user to cycle through references in the current scene that match the search/filter string. The filter string can either be the editorID or the formID of the reference or the editorID of the reference's base form. The TAB and SHIFT+TAB keys are used to cycle through the matches.
- Top Toolbar – Provides an interface for the following tools: Recent Cells & Bookmarks, Layers, Invisible References, Frozen References, Reference Groups.
- Bottom Toolbar – Provides an interface for the following tools: Snap Controls, Movement Controls, Visibility Toggles, Time of Day, FOV, selection mask.

Hovering the mouse over icons and help elements (?) will display contextual information. All OSD elements can be disabled through the render window context menu. Some OSD elements can be toggled using dedicated hotkeys.

The OSD's color theme can be modified independently from the editor's (custom) color theme by invoking the on-screen color theme editor through the render window context menu.

Key Bindings:

The render window now supports custom keyboard bindings for its many actions, allowing the user to remap the key combinations of both vanilla and newly-added tools. The complete list of remappable/bindable actions can be accessed from the render window context menu.

New Tools:

Refer to the render window key bindings for hotkey information. Some tools add options to the render window context menu too. The following list is not exhaustive:

- Reference Grouping – References can be grouped together so that selecting any reference in a particular group will result in all the members of the group being selected. Groups are identified by a unique name and are saved separately to disk along with the active plugin. Dissolving a group will remove all references from it. A reference can be a

member of only one group at a time. References can be “orphanized” (removed from their parent group, if any).

- Reference Visibility – The visibility of references – and their children, additionally - can be toggled individually.
- Reference Freezing – References can be frozen so that they cannot be selected, essentially preventing them from being modified by rogue edits. Additionally, references that aren’t modified by the active plugin can be frozen at a global, editor-wide scale.
- Recent Cells & Bookmarks – Displays the list of recently visited cells (from the Cell View window). Cells can be bookmarked for quick access through the context menu (and consequently unbookmarked through the same). Bookmarks are saved to disk separately when the active plugin is saved. Double clicking on a cell will open it.
- Layers – Conceptually similar to layers in image editing programs, layers are unique objects, not tied to a specific cell/area, to which references are “added”. Layers can be hidden and frozen. The Current Layers list will populate with layers used by references in the loaded area. The current selection can be added to a layer through its context menu. Newly added references automatically get added to the Active Layer, which can be set through the context menu.

The Default Layer is essentially the "not in a layer" layer. To remove something from a layer, simply add it to this layer. Layers are saved to disk separately when the active plugin is saved.

- Selection Painting – Holding down CTRL+ALT and then the left mouse button enables the reference selection painting mode. This lets the user “paint” the selection along several references. If the first reference in the painting sequence is isn’t selected, then painted references will be selected. If it’s selected, the painting sequence will unselect references.
- Auxiliary Viewport – 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 controlled in the same way as the default render window camera. The TAB key can be used to reset the camera’s position and rotation to the render window camera’s. It can be accessed from the View menu.
- Fly Camera Mode – Pressing the tilde (~) key (remappable) toggles the fly camera mode, allowing the viewport camera to be controlled with the keyboard and the mouse.
 - WSAD – Directional movement.
 - Space/Ctrl – Rise/Sink.

- Shift/Alt – Sprint/Crawl.

Tweaks:

Some of the following tweaks can be modified/toggled either through the render window context menu or the CSE Preferences dialog.

- Increased Responsiveness – The viewport updates at a framerate of 60 FPS. Mouse and keyboard responsiveness have been improved.
- Unrestricted Mouse Movement – The mouse pointer is no longer limited by the screen bounds when moving the camera or transforming references/pathgrid points.
- 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.
- Door markers Properties – Holding down the Control key and double clicking on door markers will now bring up their reference properties dialogue box.
- Reference Scaling – Collections of references can be scaled relatively by holding down the ALT modifier key while performing the scaling operation.
- Path Grid Editing Enhancements –
 - Path grid points can be unlinked from their linked references/relinked.
 - Linked reference indicators can be toggled in the View menu.
 - Path grid point operations can be undone/redone. 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.
- 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.
 - Areas with potential grass can be highlighted by enabling the grass overlay visibility toggle in the OSD. The overlay text must be configured through the CSE Preferences dialog.
- Alternate Camera Movement Settings – The render window is now allowed to have a second set of camera/reference movement settings. They can be modified/toggled from the OSD.
- Reference Selection – Holding CTRL and then SHIFT while drag-selecting references will remove references from the current selection. Reference selection behaviour can be

temporarily reset by holding down the ALT key while selecting objects – This allows otherwise unselectable references to be selected.

Render Window Context Menu:

Some of the newly-added context menu tools:

- Invert – Inverts the render window selection.
- Co-Planar Drop New References – When enabled, new references will be placed co-planar with the object at the cursor location.
- Offset Duplicated Refs in the Z-Axis – When enabled, duplicated references are slightly offset in the Z-axis to improve their visibility.
- Unload Current Cell(s) – Unloads the cell(s) loaded into the render window.
- On-Screen Display – Toggles for the OSD overlays and panes and OSD color theme editor.
- Render Window Key Bindings – Key bindings editor for render window actions.
- 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.

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' context 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. Multiple scripts can be opened for editing using the "Open in Script Editor" context menu tool.

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*.

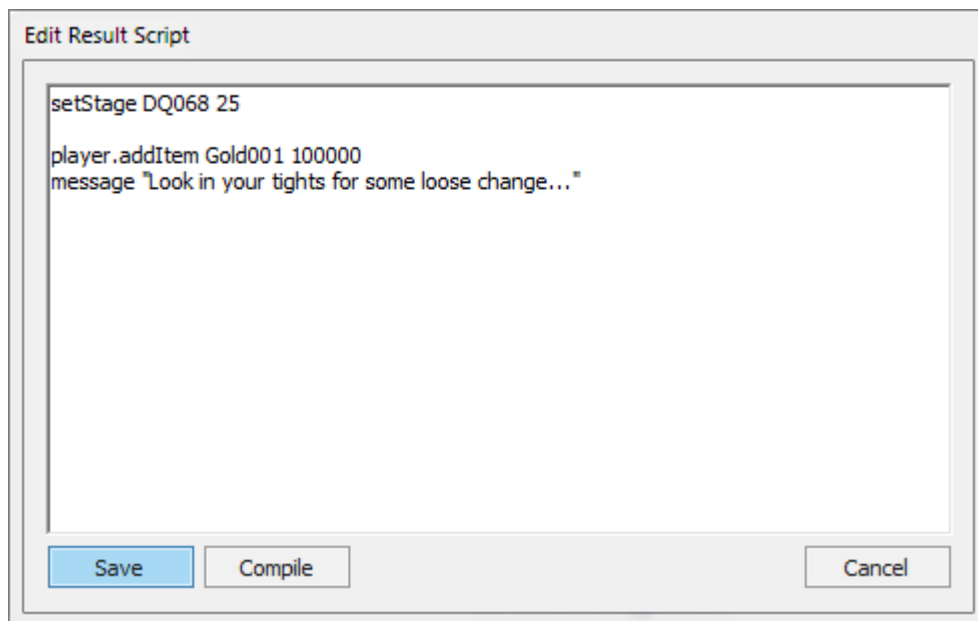
In addition to the above, CSE allows multiple list items to be selected for drag-drop operations. For instance, you could select multiple Hair or Eye records and drop them into a Race record in a single go.

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.

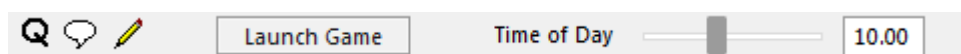
Result Script Editing:

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



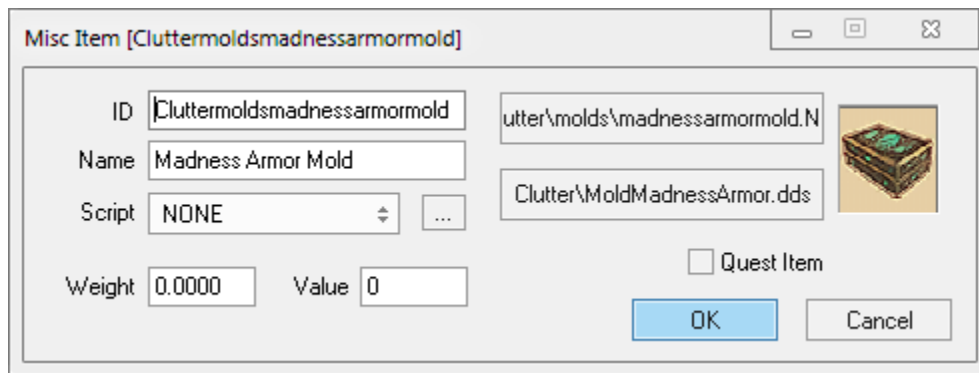
Time of Day Slider:

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:

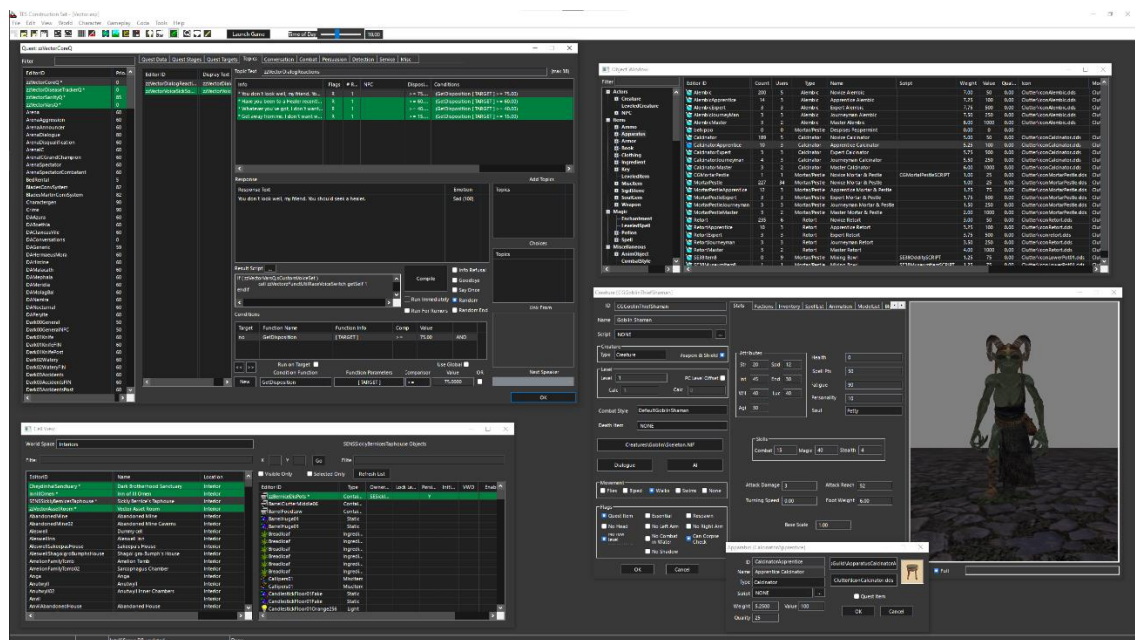
The following dialogues' UIs – amongst others - have been improved to enhance usability and better use the available screen real estate:

- 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.

Custom Color Theme/Dark Mode:

The CSE adds a custom color mode that lets you fully customise the colors of all UI widgets and elements used by the editor dialogues. This can be accessed from the View main menu. The color parameters can be customised through the CSE Preferences dialogue. By default, they are set up with a dark mode theme.



While the custom color theme can be toggled from the main menu at any given time, certain UI elements might not correctly update their colors until either their dialogues are closed and reopened or until the editor is restarted.

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:

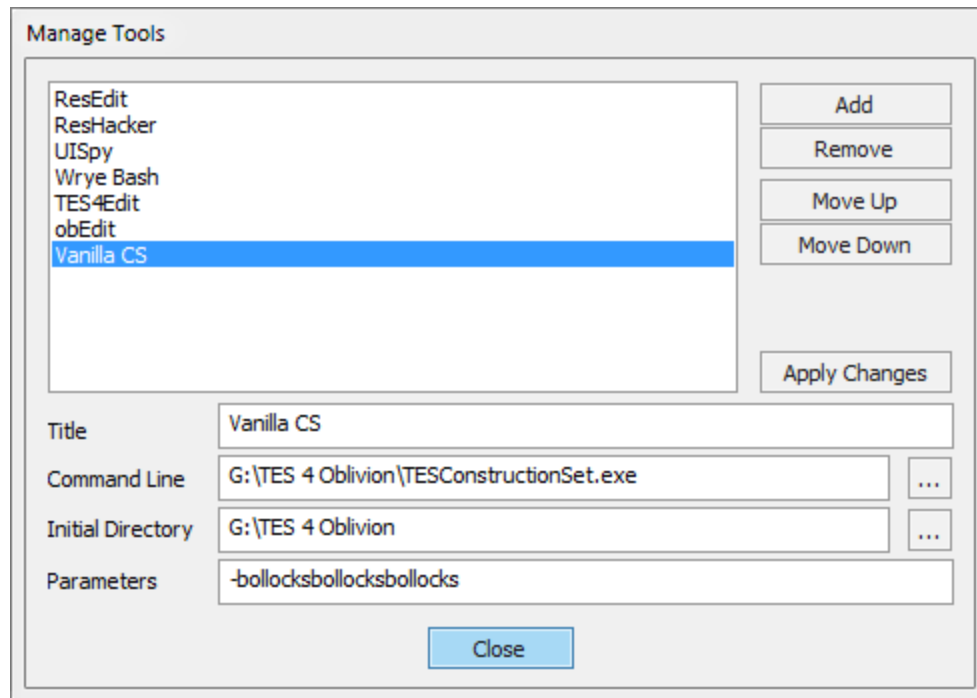
Sound files, FX and voice files alike, that stored in BSA archives can be sampled from the editor directly without having to manually extract them beforehand.

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.

Persistent Window Locations:

Editor dialogs are automatically restored to their previous position upon opening.

Purge Loaded Resources:

**** Experimental ****

The editor's in-memory textures and meshes can be reset by invoking the "Purge Loaded Resources" tool, found in the View main menu. This will cause assets to be loaded anew from disk.

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	Fix for the bug that caused a CTD when a plugin

Plugins	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 preemptively 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.
Script-Magic Item Cross Reference	Allow for use info cross-references between scripts and magic items.
Render Window “Fall” Operation on Linked Path Grid Points	Fix for the bug that caused the “Fall/Floor” render window tool to displace path grid points with linked references in increment of 65 units
Render Window Center/Top Camera	Fix for the bug that occasionally caused a CTD when centering/topping the render window camera
Render Window Rotation in Local Axes	Fix for the bug that prevented the rotation of a multi-reference selection in their local axes
Render Window Undo/Redo Crash	Fix for the bug that occasionally caused a CTD when undoing/redoin an action
Render Window Reference Movement Crash	Fix for the bug that caused a CTD when references were moved too far away from the camera
Render Window Panning Mouse Capture	Fix for the bug that didn’t release mouse capture when the SPACE key was released
Object Window Drag-Drop from Windows Explorer	Fix for the bug that broke drag-and-drop operations when running the editor with elevated privileges

Data Dialog Save Corruption

Fix for the bug that could potentially cause save corruption if the loaded state of the currently active plugin's dependencies was reset in the data dialog

Multiple Editors Crash

Fix for the bug that caused a crash that was triggered by the following conditions: multiple editor instances are disabled in the editor's INI, a primary instance of the editor has an open dialog with a preview control, and a second instance of the editor is launched

New tools

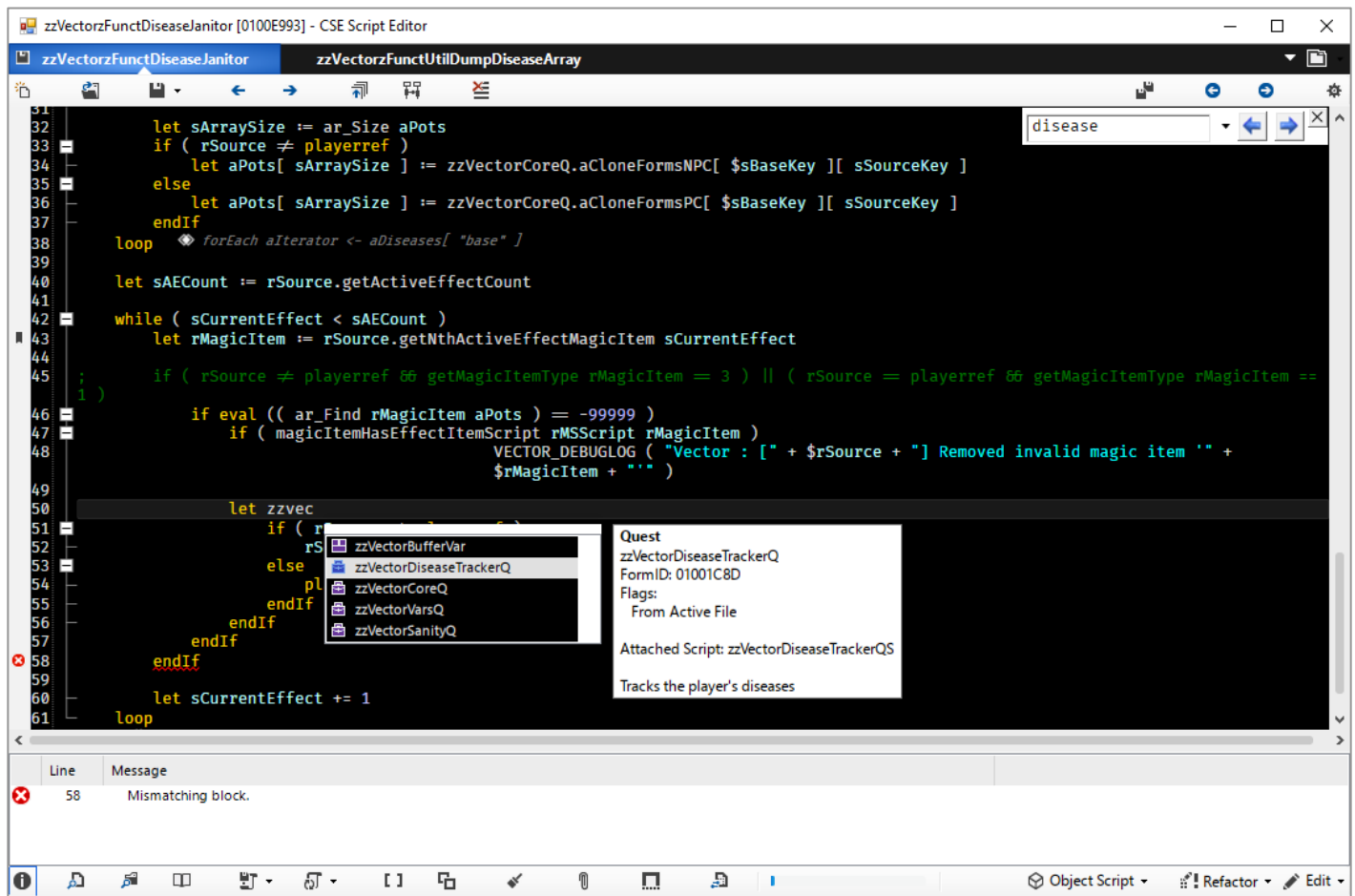
Coda Script

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 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.

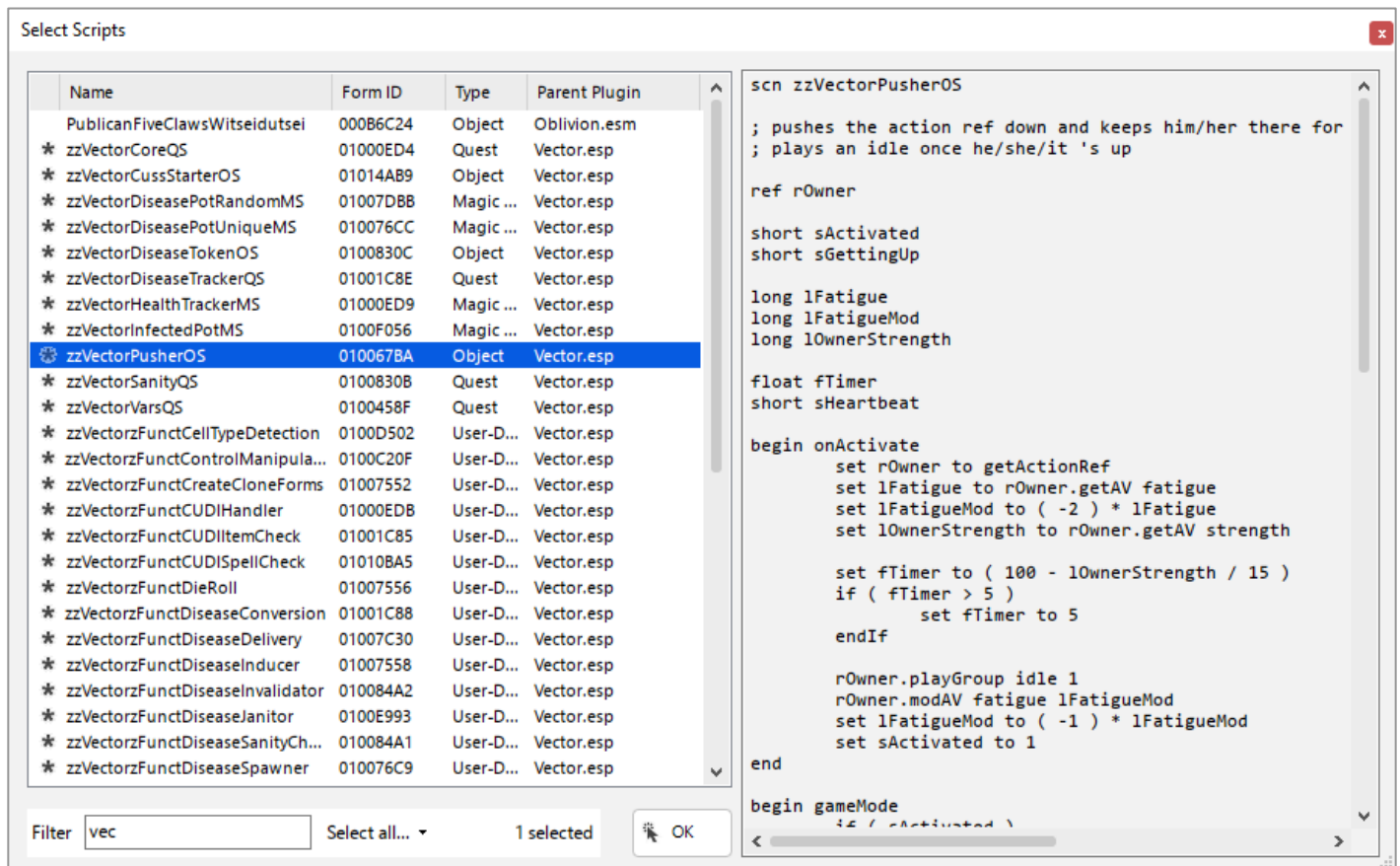


Workspaces can be rearranged by dragging their tab across the tab strip. They can also be "torn" out of their tab strips and dropped onto either another editor window's tab strip or the desktop. The former relocates the workspace to the destination window, while the latter spawns a new editor at the drop location and moves the workspace into it. A floppy-disc icon in a workspace's tab indicates that its script has unsaved changes.

To quickly switch between tabs, the editor implements a [tab filter](#) that allows the user to filter the list of open workspaces by the editorID of their script.

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

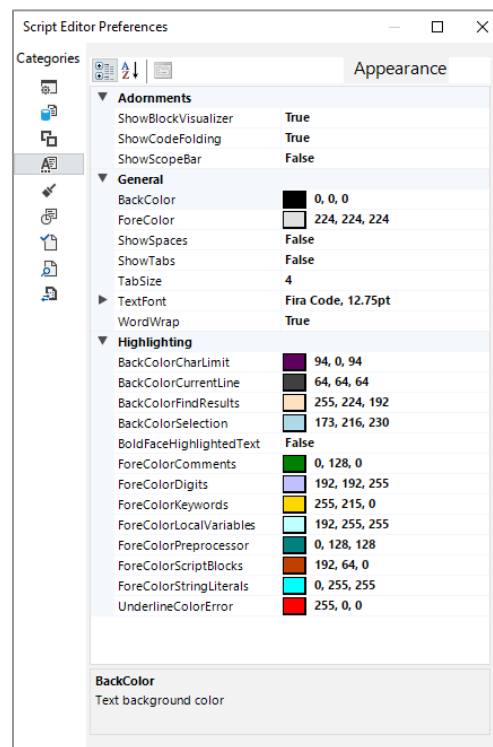
- New – Creates a new script.
- Open – Opens the Select Script dialogue for script selection.



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 – An asterisk denotes a script is from the active plugin, an "X" mark denotes a deleted script, a Warning icon denotes an uncompiled script, and a double-arrow icon indicates a script being synced to disk in the background.

- 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. Any number of scripts may be selected for deletion.
- Save All Open Scripts – Attempts to compile and save all open workspaces.
- Navigate Backward – Jump back in the tab navigation stack. The stack is populated every time the user switches between tabs.
- Navigate Forward – Jumps forward in the navigation stack.
- Preferences – Opens the Preferences window. Some changes may require a restart of the script editor.



The bottom toolbar is actually a splitter bar which can be moved to resize the editor area and show the controls beneath it.

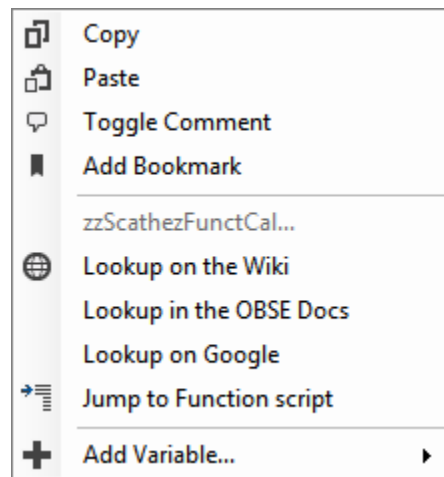
- Messages – Displays output messages from the script validator, pre-processor and the compiler. Double clicking on an item will move the caret to the appropriate line.
- Find Results – Displays the last executed Find/Replace operation's results. Double clicking on an item will move the caret to the appropriate line.
- Global Find Result List – Displays the results of Find/Replace operations performed on all global scope, i.e., on all open workspaces.
- Bookmarks – Displays stored bookmarks for the loaded script, if any. Double clicking on an item will move the caret to the appropriate line. Existing bookmarks can be deleted by selecting them in the list and pressing the Delete key.
- 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.
- 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 attached to a new or an existing scriptable object.
- Code Snippet Manager – Brings up the [Code Snippet Manager](#) dialogue.
- Sync Scripts To Disk – Brings up the [Sync Scripts To Disk](#) dialogue.
- Progress Bar – Indicates the compiled bytecode size of the current script.
- Script Type Menu – Specifies the current script's type.
- Refactor Menu –
 - Document Script - Inserts comment-based documentation into the script text.
 - Rename Variables – Allows the current script's variables to be renamed without losing their indices. Double click on a variable in the list to modify its name.
 - Modify Variable Indices – Allows the modification of the current script's variable indices. Double click on a variable in the list to modify its index.
- 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:



- 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.
- Add Bookmark – Adds a bookmark to the current line. Bookmarks are saved with the script text as metadata and indicated by an icon next to the line number column. Hovering the mouse over the icon displays its description.
- 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.
- 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 attached 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.
- **Open Import File** – Context specific. Opens the pre-processor import file. Only displayed for the IMPORT directive.
- **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.

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

- **Syntax Highlighting** – Local variables, keywords, string and numeric literals, etc. can be assigned specific colors in the Preferences window.
- **Code Folding** – The editor allows script blocks to be collapsed and expanded arbitrarily.
- **Brace Matching** – The editor highlights parentheses and braces contextually.
- **Block Visualizer** – The editor displays visual adornments next to block end specifiers when their starting lines are not inside the viewable area.
- **Outline View** – The editor displays a tree-view of the script's structure that allows for easy navigation within the script.
- **IntelliSense** – Intelligent auto-completion.
- **Code Snippet Manager** – Allows frequently-used code snippets to be quickly inserted.
- **Script Validation** – More fine-grained error processing.
- **Auto-Recovery** – The editor periodically saves a backup of the active workspace and restores them in case of a crash/unexpected shutdown.
- **A Pre-processor** – A C'ish pre-processor for ObScript/Legacy.

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 either the Tab or Enter key, while the Escape key closes the pop-up. Holding down the Ctrl key will temporarily dim the pop-up. The auto-popup behaviour can be turned off in the Preferences window.

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

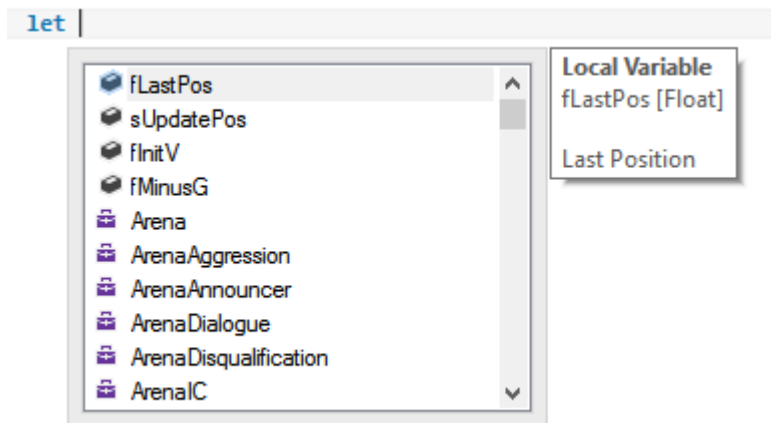
Short sArg1      ; Some arg
Float fArg2      ; Another arg

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

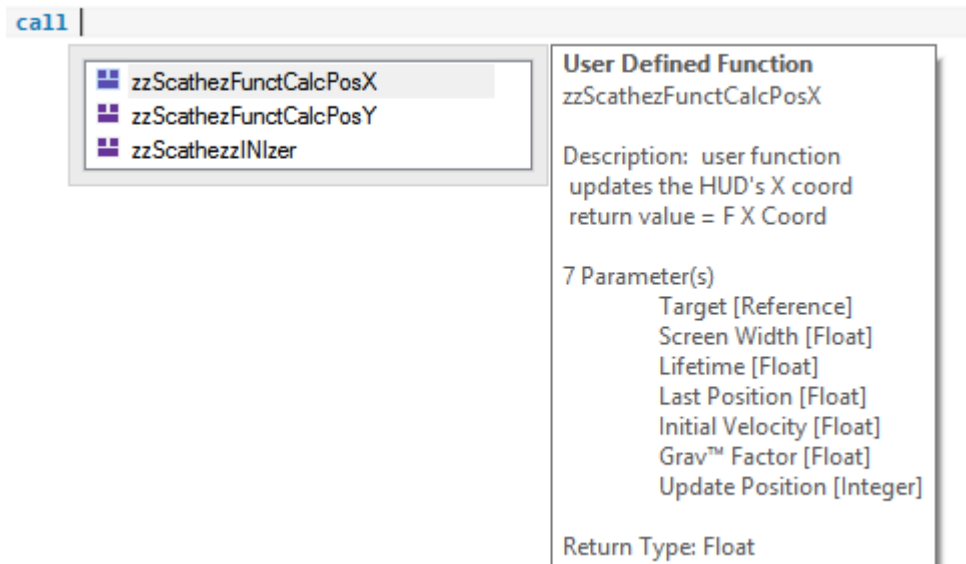
The description will include the comment text between the script name declaration and the first local variable's. Arguments are treated as variables and enumerated. And finally, the return type of the UDF is stated.

IntelliSense supports the following context specific triggers:

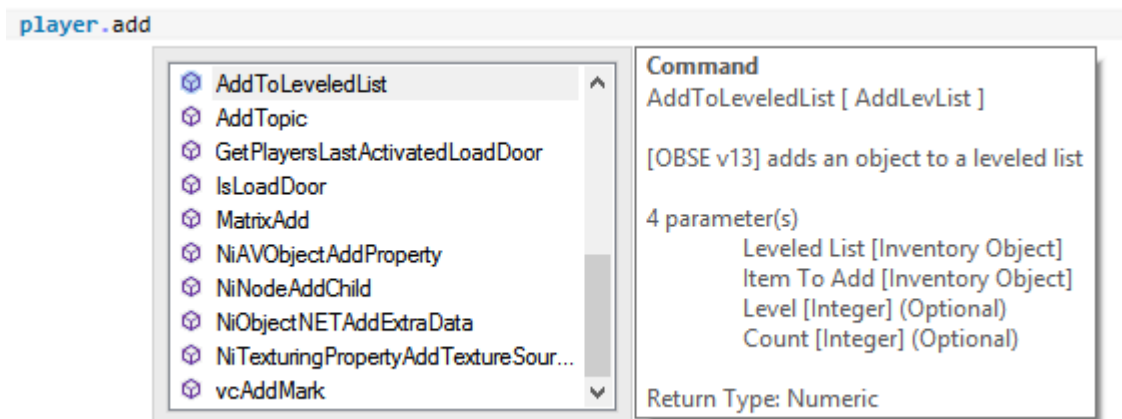
- Set or Let – Suggests local variables, global variables and quests.



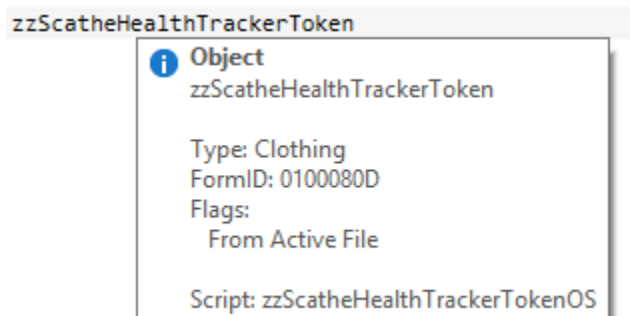
- Call – Suggests user defined functions.



- 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

Name	Shorthand
Annoy Vorians	aux
Quest Delay Time	qwe
Snippet #3	sqw

Snippet Data

Name:

Shorthand:

Description:

Variables

Name	Data Type
fQuestDelayTime	int

Code

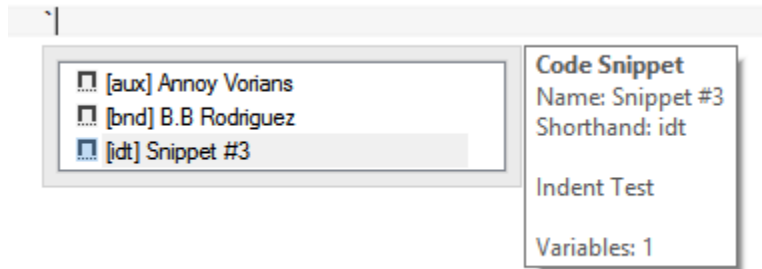
```
if (fQuestDelayTime != 0.001)
    let fQuestDelayTime = 0.001
endif
```

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.
- Invalid operators.
- 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

Warning messages are indicated by an exclamation mark next to the corresponding line number.

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"
;}

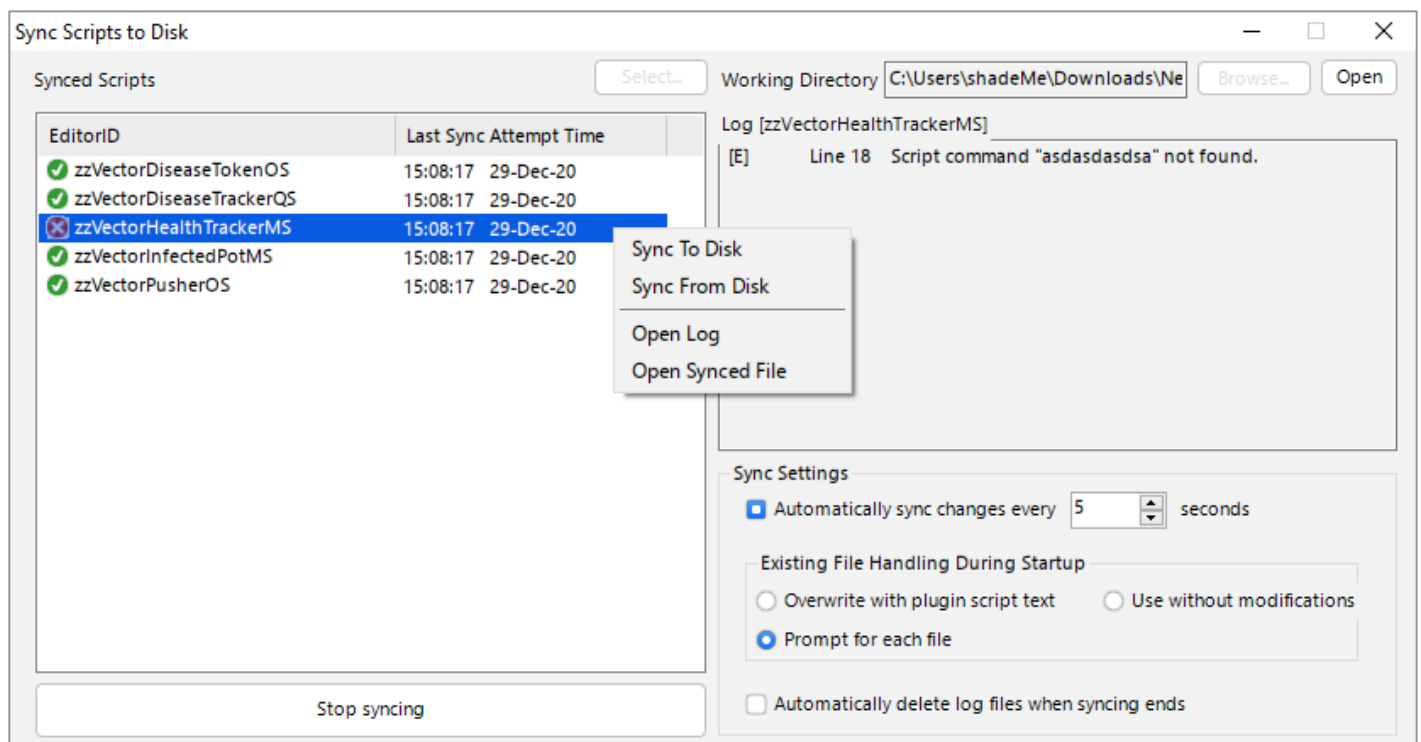
@if foo
;{
;   print "This line will not be parsed as 'foo' has not been defined"
;}

```

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.

Sync Scripts To Disk

While the new script editor is a significant step-up from the vanilla script editor (which was little more than a textbox) that offers many quality-of-life features to improve productivity, it is still limited by the fact that it is essentially a new piece of software that one needs to get used to. Furthermore, it's not as powerful as some other general purpose code editors available on the Internet such as [Sublime Text](#) or [Visual Studio Code](#). With the [Papyrus scripting language](#) in TES 5: Skyrim, the game designers at Bethesda implemented a new build system for their scripting engine that no longer relies on scripts and their bytecode being embedded in plugin (ESP/ESM) files. Instead, the script text (source code) and the compiled code (bytecode) are saved as individual files on disk which are then consumed by the script compiler (also external to the TES 5 editor, the Creation Kit) and the game respectively. This has multiple advantages: it allows modders to create and modify scripts using external text editors, and version control is now possible for individual scripts. The CSE implements a new tool in the script editor that enables a similar workflow in the Construction Set as well. The 'Sync Scripts To Disk' tool can be activated either through the Gameplay main menu or the bottom toolbar of a script editor window.



Firstly, you select the scripts you'd like to synchronise to disk. There are no limits on the number of scripts that can be synced, nor must they be limited to a single source plugin file. Then, you select a working directory on your local drive that will act as the staging area for the synchronisation process. Finally, you click on the 'Start syncing' button to begin the process.

The CSE will then attempt to create two files for each of the selected scripts in the working directory: one with the actual script text and another that will act as the log file for compiler messages. Both files will have the same name as the parent script's editorID, and the default file extension of the script text file is '.obscript' (can be modified in the script editor's preferences) while the log file's is '.log'. If a file already exists at the script text file's path, it can either be overwritten with the script text as found in its parent plugin file or used without any modifications. Depending on your settings, this can be configured to execute automatically or prompt for each file. Log files are always overwritten.

Once the startup process is completed, the synced scripts listview will be updated with the synchronisation status and timestamps of each synchronised script. If automatic synchronisation of changes is enabled, any changes made to the local file will be (periodically) sent to the editor for compilation. If the script compiles successfully, the script text in the parent plugin is updated with the contents of the local file, and a success message is written to the corresponding log file. If compilation fails, the script text in the parent plugin is left unmodified, and the compiler output is written to the log file in the following format:

[<message_type:E or I>]\tLine <line_number>\t<message>. This output can potentially be parsed by an external editor to display error messages.

If automatic synchronisation is disabled, individual or multiple scripts can be manually synchronised by selecting them in the listview and using the 'Sync From Disk' context menu item. If you need to revert the local on-disk file to the state found in the script's parent plugin, you can use the 'Sync To Disk' context menu item. The 'Script Sync' console message channel contains a log of all sync operations performed during a given sync session.

To end the sync process, click on the 'Stop syncing' button. This will forcefully synchronise the contents of all the synced scripts from disk one final time before ending the synchronisation process and updating the listview with the results. The contents of the log file can be displayed for any failed operations regardless of whether they were set to be automatically deleted at the end of the sync process. Closing the dialogue will end any active sync process.

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 n th workspace
CONTROL + SPACE	Show open tab filter
CONTROL + ALT + SPACE	Show outline view
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	Add comment (for multiple lines) Toggle comment (for single lines)
CONTROL + W	Remove comment (for multiple lines) Toggle comment (for single lines)
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 (+ SHIFT) + F	Show inline Find tool/Find-Replace dialog Specified in the Preferences window
CONTROL + H	Show Find/Replace dialog
CONTROL + G	Go to line
CONTROL + E	Go to offset
CONTROL + Left Mouse Click on Scriptable Object Identifier	Jump to script
CONTROL + (Pipe)	Jump to script at caret
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

Centralized Use Info Listing

Type	Editor ID	Form ID	Parent Plugin
Static	HorseMarker	00000012	Oblivion.esm
MiscItem	VarlaStone	00000194	Oblivion.esm
MiscItem	WelkyndStone	00000191	Oblivion.esm
SoulGem	BlackSoulGem	00000192	Oblivion.esm
SoulGem	AzurasStar	00000193	Oblivion.esm
MiscItem	Lockpick	0000000A	Oblivion.esm
MiscItem	DASkeletonKey	00000008	Oblivion.esm
MiscItem	RepairHammer	0000000C	Oblivion.esm
Static	DoorMarker	00000001	Oblivion.esm
Static	XMarker	0000003B	Oblivion.esm
Static	XMarkerHeading	00000034	Oblivion.esm
Static	MapMarker	00000010	Oblivion.esm
Static	TravelMarker	00000002	Oblivion.esm
Static	NorthMarker	00000003	Oblivion.esm
Door	PrisonMarker	00000004	Oblivion.esm
Static	TempleMarker	00000006	Oblivion.esm
Static	DivineMarker	00000005	Oblivion.esm
MiscItem	Gold001	0000000F	Oblivion.esm
Container	LootBag	0000000E	Oblivion.esm
Container	StolenGoods	00000011	Oblivion.esm
Clothing	JailShirt	00000017	Oblivion.esm
Clothing	JailPants	00000015	Oblivion.esm
Clothing	JailShoes	00000016	Oblivion.esm
Static	FlameNode1	0000001F	Oblivion.esm
Static	FlameNode2	00000020	Oblivion.esm
Static	FlameNode3	00000021	Oblivion.esm
Static	FlameNode4	00000022	Oblivion.esm
Static	FlameNode5	00000023	Oblivion.esm
Static	FlameNode6	00000024	Oblivion.esm
Static	FlameNode7	00000025	Oblivion.esm
Static	FlameNode8	00000026	Oblivion.esm
Static	FlameNode9	00000027	Oblivion.esm
Static	FlameNode0	0000001E	Oblivion.esm
Static	FlameNode10	00000028	Oblivion.esm
Static	FlameNode11	00000029	Oblivion.esm

Filter

Used By These Objects

Type	Editor ID	Form ID	Parent Plugin
Idle Animation	SE03ResonatorBossPreachIdl...	00079147	Oblivion.esm
Idle Animation	SE03ResonatorBossPreachIdl...	00079148	Oblivion.esm
Idle Animation	SE03ResonatorPrayIdle01	0007914A	Oblivion.esm
Idle Animation	SE03ResonatorPrayIdle02	0007914B	Oblivion.esm

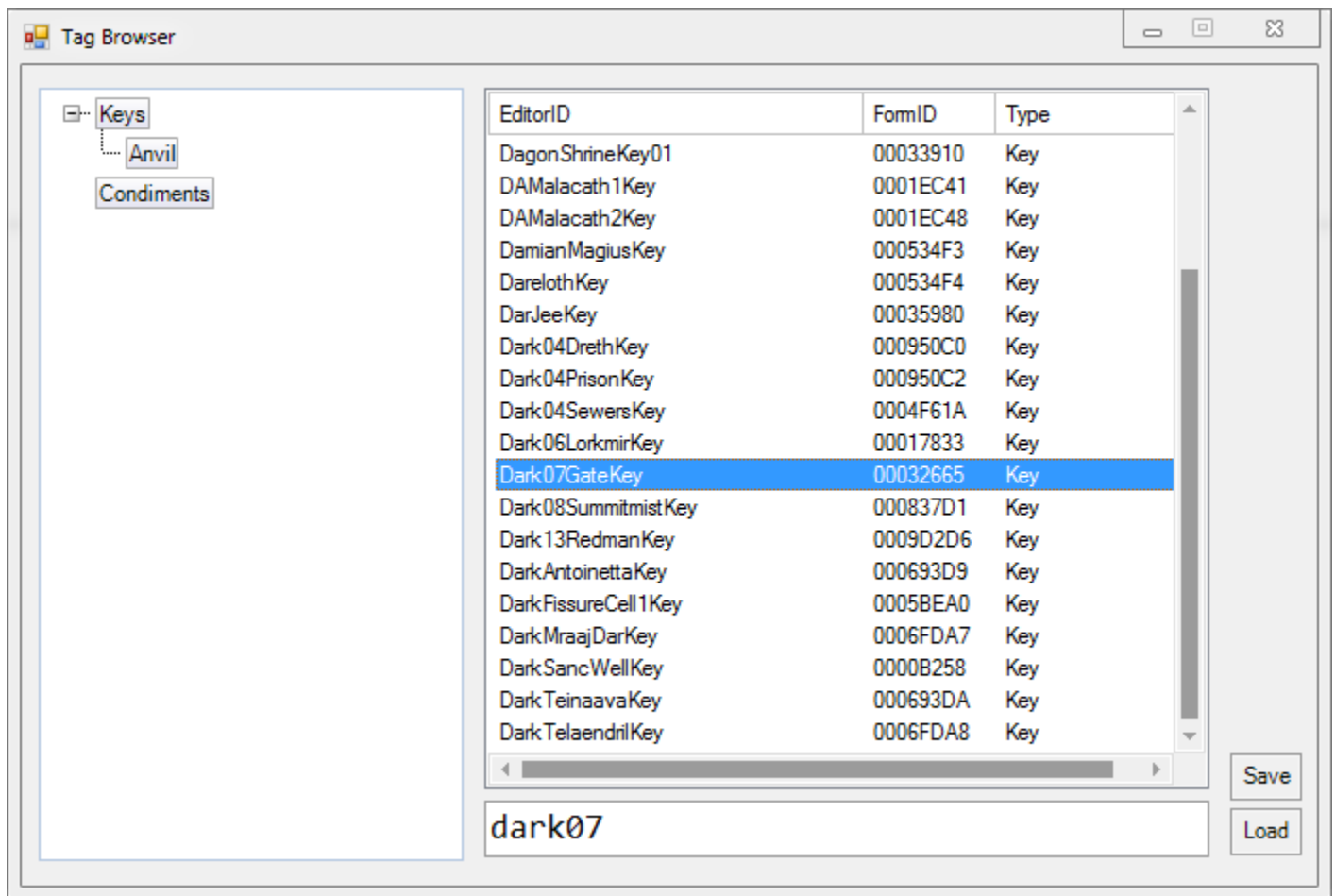
Used In These Cells

World	Cell	Grid	Use C...	First Reference
Interior	XPAichan01		4	0005F2F5
Interior	ChorrolMarkTest		1	00062D03
Interior	XPGloomstonePassage02		1	000171CC
Interior	OblivionRDCavesMiddleA05		2	000A4DC1
Interior	SEPasswallJayredsTent		1	00091D67
Interior	SENSGreenmoteSilo		2	00016803
Interior	LeyawiinFiveClawsLodge		1	000A565F
Interior	BravilCryptOfTheNightMo...		5	000096DA
Interior	ICArenaSpectators		2	0003D6D9
Interior	XPGloomstonePassage03		1	000171CE
Interior	SkingradCastleDungeon		3	00002DE0
Interior	LeyawiinMagesGuild		1	000025FF
Interior	XPEbrocca03		1	0005A2BA
Interior	TestJbrowneXPGloomston...		2	00079A5A
Interior	AnvilCastleGreatHall		7	0000C0F5
Interior	LeyawiinCastleDungeon		3	00035E96
Interior	CheydinhalCastleDungeon		2	000C451E

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.

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.

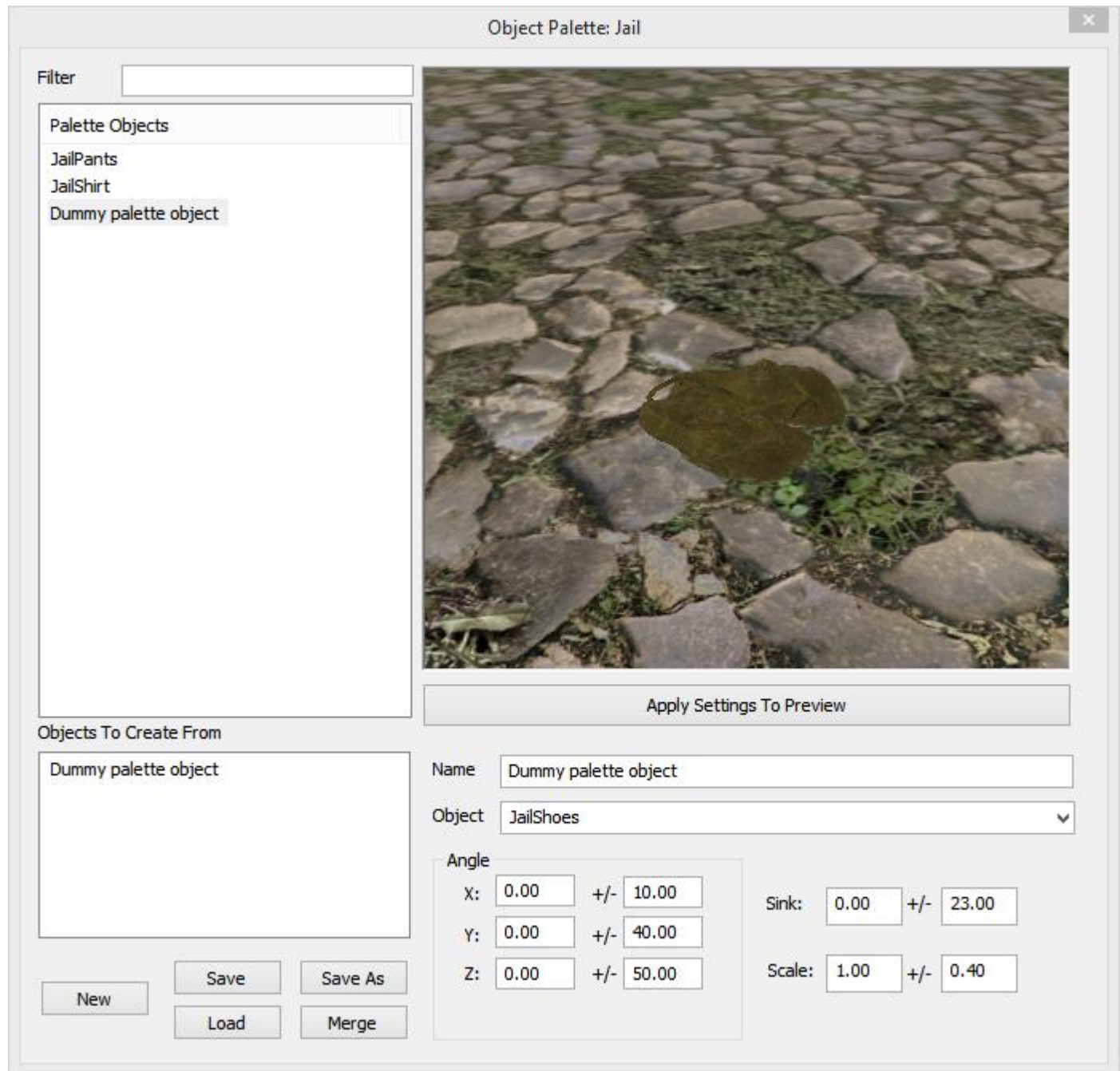
It can be accessed from the View menu.

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.

Object Palette



Object Palettes (OPALs) are user-created collections of placeable objects. Once you've created an OPAL, you can use it to quickly find, view, and place objects in your levels. It allows you to:

- Create a group of any type of objects
- Quickly place objects on the surface that you click

- Semi-randomly rotate, scale and translate objects as you place them
- Preview objects before you place them
- Rename objects so they make sense to you
- Save/Load different palettes

Working with OPALs

The Object Palette window can be accessed from the World menu.

- The **Filter** allows you to filter for a specific subset of objects in the palette. For example, if you enter '08', only NorRubblePile08 will be shown.
- The **Palette Objects** list shows all of the objects in the palette, in alphabetical order.
 - To add items to this list, drag them in from the Object Window.
 - To remove items, click on an item in this list and press the Delete key.
 - If you select any of these objects, a preview of the object automatically appears in the window at right. This window can be manipulated just like the Preview Window, allowing you to pan, rotate, and zoom around the object for a better look.

To select multiple objects, click on an item while holding down the Control key.

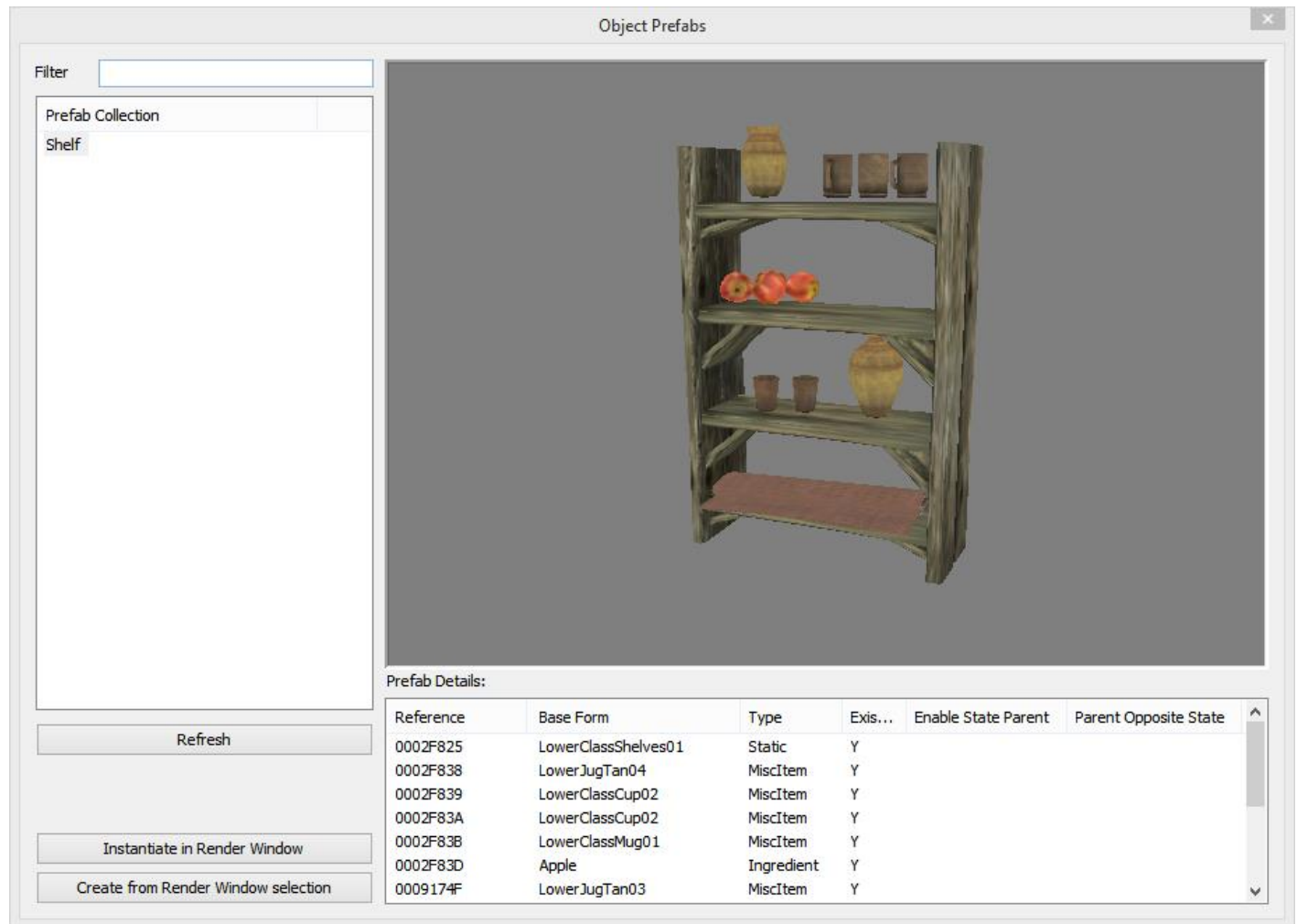
- The **Objects to Create From** list shows which objects the Palette will place in the world when you use it. This is almost always just a single object (the one you have selected), though if you select multiple objects at the same time, they'll all appear here, and the palette will pick from them at random.
- Below that are the file options. You can create a **New** (empty) Object Palette, **Save** or **Save As** the current Palette, **Load** another Palette, or **Merge** this Palette with another.
- The **Name** field allows you to rename the object in this Palette only. This can be helpful if you have trouble remembering the name of an object.
- The **Object** field lets you change what object the palette entry is pointing to.
- The **Angle** allows you to specify the initial X, Y, or Z rotation the object will have when you place it.
 - The first column allows you to specify a specific rotation.
 - For example, all books are lying on their side by default. If you plan to place a large number of books standing on end, giving them an Angle X: 90 will allow them to start standing up, which may make this faster.
 - The second column allows you to specify a restricted random rotation (within +/- degrees).

- When placing minor items like spoons or cups, this may give you some quick variety.
- The **Sink** field is a Z-offset: it causes the item to be placed higher or lower than the point the Creation Kit calculates when you click on a point.
- The **Scale** field automatically scales the item by a fixed or random amount. This can be useful for things like hay or rugs, but shouldn't be used on furniture or items.
- The **Apply Settings to Preview** button will allow you to quickly see what your Angle, Sink, or Scale changes will mean for the object.

To use the OPAL, select one or more objects in the Palette Objects list. Then, in the Render Window, hold CTRL+ALT and Right-Click. The rubble appears where you clicked, and starts selected, allowing you to rotate or reposition it as you like. It may be helpful to enable “Havok Settle” mode while placing new objects.

OPAL files are saved to the “Data\BGSEE\OPAL” folder with the extension “cseopal”.

Object Prefabs



Object Prefabrications (or Object Prefabs) are collections of object references that can be placed in cells. They enable the rapid prototyping of commonly used object arrangements/setups. For instance, one could put together a reusable bedroom for an inn, add the relevant clutter to the room and export it quick reuse in a different mod. Similarly, multi-part traps and puzzles can be rigged up appropriately and reused in multiple dungeons.

The Object Prefabs window can be accessed from the World menu.

- The **Prefab Collection** list displays all the prefabs found in the current workspace.
 - To remove a prefab file, click on its item in this list and press the Delete key.
- The **Filter** field allows you to filter for a subset of prefabs according to their names. Prefab names contain their relative file path without the extension.
- The **Refresh** button reloads the prefab files in the current workspace.

- The **Instantiate in Render Window** button places a copy of the currently selected prefab in the active cell and adds them to the selection. The new references are placed at the camera's location.
- The **Create from Render Window selection** button exports the current Render Window selection and saves it as a prefab file.
- The **Prefab Details** list displays information about the currently selected prefab's references. Selecting a reference will highlight it in the preview.

Prefabs preserve the state of their constituent object references, i.e., their position, rotation, scale, editorID, etc. Non-reference extra data like count and charge is also preserved. However, reference extra data like merchant container, XMarker, etc. (with the exception of enable state parent) are not. In addition to the above, the base records of the references are exported as a shallow copy, i.e., any records they reference (like scripts, inventory items, etc.) will not be exported. If the base record of a prefab's reference isn't found at the time of instantiation, it will be automatically created.

Prefab files are saved to the "Data\BGSEE\Object Prefabs" folder with the extension "cseprefab". Subfolders can be created as required.

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.