## Script Functions

**AI**
AiActivate , ObjectID, [reset]
AiEscort, ActorID, duration, x, y, z, [reset]
AiEscortCell, ActorID, CellID, duration, x, y, z, [reset]
AiFollow, ActorID, duration, x, y, z, [reset]
AiFollowCell, ActorID, CellID, duration, x, y, z, [reset]
AiTravel, x, y, z, [reset]
AiWander, range, duration, time, [idle2], [idle3], …[idle9], [reset]
GetCurrentAIPackage ;returns the current package the NPC is doing (0 through 4)
GetAIPackageDone ;returns true if current package has finished.
GetDetected, ActorID Returns true if object can detect ActorID. Slow function, do not call a lot.
ForceSneak Make the npc sneak if he wanted to run he will only walk
ClearForceSneak
GetForceSneak

**Animation**
PlayGroup, GroupName, [Flags]
LoopGroup, GroupName, Number, [Flags]
SkipAnim

**Collision**
GetCollidingPC Object returns true if PC is colliding with it.
GetCollidingActor Object returns true if ANY actor (including PC) is colliding on it.
HurtCollidingActor, Val Damages colliding actor Val health per second
HurtStandingActor, float value is per second and should be negative to hurt him

**Combat**
GetAttacked return 0 if the actor has never been attacked and 1 if he has ever been attacked
GetTarget, ActorID returns 1 if object's combat target is ActorID
HitAttemptOnMe, ObjectID returns true if a hit on the calling Actor is attempted by objectID in melee (such as hammer01)
HitOnMe, ObjectID returns true if calling Actor is hit by objectID in melee (such as hammer01)
OnDeath Returns true for one frame when Actor is killed.
OnKockout Returns true for one frame when Actor is knocked out.
OnMurder Returns true for one frame when Actor is murdered.
OnPCHitMe NOT A FUNCTION!, but used as a short var that gets set when object is hit.
StartCombat, ActorID
StopCombat

**Dialogue**
AddTopic, TopicID Adds TopicID to PC's list of known topics
ClearInfoActor Makes this info NOT appear in the journal.
ForceGreeting makes the NPC start dialogue with the PC. Does NOT matter where the NPC is.
GetJournalIndex, JournalID
Goodbye Ends current dialogue, PC can only press "Bye"
Journal, ID, Index Adds the index entry into the journal for the journal ID

Choice, “Button1”, …[“Button5”] similar to "messagebox", but for dialogue choices.
SetJournalIndex JournalID index Sets the Journal to that index. Can move up or down.

**Faction**
GetPCRank, [FactionID] Returns PC's rank in faction. This will default to the actor's faction if FactionID is not defined. Returns 0-9 and -1 if not a member.
GetPCFacRep, [FactionID]
GetRace, “RaceID” Returns 1 if the object is of RaceID
LowerRank Lowers the object's rank in its current faction.
ModPCFacRep, var, [FactionID]
ModFactionReaction, factionID1, factionID2, var
PCClearExpelled Clears currently expelled flag.
PCExpell [FactionID] Expells PC from NPCs faction.
PCExpelled [factionID] Returns 1 if PC has been expelled once from calling object (NPC) Faction, or a faction can be defined to get a specific one.
PCJoinFaction [FactionID] FactionID is optional if it is not added it will use the faction of the npc who called the function
PCLowerRank
PCRaiseRank Raises the PC 1 rank in the NPCs faction. If pc is not part of the faction, makes rank 1
RaiseRank Raises the object's rank in its current faction.
SameFaction Returns 1 if PC is in the faction of the calling object (NPC).
SetPCFacRep, var, [FactionID]
SetFactionReaction, factionID1, factionID2, var

**Item/Object**
AddItem, ObjectID, count adds item to calling objects inventory
Activate activates the item (uses it's default activation)
Drop, ObjectID, count calling Actor drops item into world at his feet.
Equip, ObjectID equips the calling item on its owner
GetItemCount, ObjectID returns the count on the objectID in the calling object
OnActivate returns true if calling object is activated
OnRepair returns true if calling object is repaired at all
RemoveItem, objectID, count removes the item from the calling object. Does NOT drop it.
RepairedOnMe, objectID returns true if calling object is repaired by objected
UsedOnMe, objectID returns true if objectID is used on calling object

GetWeaponType returns a number (1-long blade one hand; 2-long blade two hand, 3 - blunt, etc.) based on the class of weapon the actor has equipped.
GetArmorType
HasItemEquipped

**See also:**
Inventory Scripts

**Magic**
AddSoulgem, CreatureID, SoulgemID
AddSpell, SpellID adds spell item to calling object
Cast, SpellID, TargetID calling object casts spell onto target, spells only.
DropSoulgem, CreatureID
DisableLevitation Disables levitation magic.
DisableTeleporting
EnableLevitation
EnableTeleporting

GetBlightDisease Returns 1 if has BlightDisease
GetCommonDisease Returns 1 if has CommonDisease
GetEffect, Effect Returns 1 if Actor is being effected by effect
GetSpell, SpellID Returns true if object has SpellID in inventory
GetSpellEffects, SpellID Returns true if calling object is being effected by spellID
HasSoulgem, CreatureID
RemoveSoulgem, CreatureID
RemoveSpell, SpellID remove spell item from calling objects inventory
RemoveSpellEffects, SpellID remove spell effects of SpellID from effecting the calling object
RemoveEffects, Effect removes all spells with the EffectEnum from effecting the calling object

**Miscellaneous**

Enable Makes the object visible and processed.
DontSaveObject Call when you do not want object change in save game.
Disable Makes the object invisible and it will not be processed.
FadeIn time time is > 0 and <= 10.0
FadeOut time same as fade in
FadeTo alpha speed FadeTo 50 2.0 (Fades screen to 50% in 2 seconds). 0 is full transparency. 100 is black.
GetButtonPressed
GetDisabled Returns true if object is disabled.
GetLocked returns 1 or 0
GetPCCell, Cell ID Returns true if PC is in CellID named. Works like dialogue in that "Vivec" returns true for both "Vivec" and "Vivec, Fred's House".
GotoJail Sends PC to Prison.
GetMasserPhase
GetSecundusPhase
GetStandingPC Object returns true if PC is standing on it.
GetStandingActor Object returns true if ANY actor (including PC) is standing on it.
Lock, Var sets the door or container to the specified lock level.
MenuMode
MessageBox, "Message", [var1], [var2], ["button1"], ["button2"]
PayFine call after paying a crime fee to clean AI
PlayBink "filename" flag Pauses game and plays video. Set Flag to true if player can escape movie.
Random, Value
ShowMap cellID show all the cells that start with this string on the world map
Unlock makes object unlocked, doors and containers only
Xbox returns true if running Xbox version, used for button messages

SetWaterLevel, Val Sets water level at a certain height for an interior
GetWaterLevel
ModWaterLevel, Val

**Movement**

CellChanged ;returns 1 for one frame when player changes cells
CellUpdate
GetPos, axis
GetAngle , axis
GetLOS, ObjectID ;returns true if object has a line of sight to other object(ObjectID)
GetDistance, ObjectID ;returns the distance the object has to ObjectID.
GetStartingPos , axis
GetStartingAngle , axis

Move, axis, units/sec
MoveWorld, axis, units/sec
PlaceAtPC, ObjectID, count, distance, direction
PlaceItem, ObjectID,  x, y, z, rot
PlaceItemCell, ObjectID, cellID x, y, z, rot
Rotate , axis, angle/sec
RotateWorld, axis, angle/sec
Position, x, y, z, zRot
PositionCell, x, y, z, zRot, “cellID”
SetAtStart
SetAngle, axis, angle
SetPos, axis, pos

| | |
|---|---|
| ForceRun | Make the npc run |
| ClearForceRun | |
| GetForceRun | |
| | |
| ForceJump | Make the npc jump like crazy |
| ClearForceJump | |
| GetForceJump | |
| | |
| ForceMoveJump | Make the npc jump when he is moving |
| ClearForceMoveJump | |
| GetForceMoveJump | |
| | |
| GetPCSneaking | Returns 1 if player is in Sneak Mode |
| GetPCRunning | Returns 1 if player is running |
| GetPCJumping | Returns 1 ever time the player starts a jump |

**Player Controls**

| | |
|---|---|
| DisablePlayerControls | Can only look around with mouse or use options menu, nothing else and menus disappear. |
| DisablePlayerFighting | |
| DisablePlayerJumping | |
| DisablePlayerLooking | |
| DisablePlayerMagic | |
| DisablePlayerViewSwitch | |
| DisableVanityMode | |
| EnableLevelUpMenu | |
| EnablePlayerControls | Enables the controls and menus. |
| EnablePlayerJumping | |
| EnablePlayerFighting | |
| EnablePlayerLooking | |
| EnablePlayerMagic | |
| EnablePlayerViewSwitch | |
| EnableRest | |
| EnableVanityMode | |
| GetPlayerControlsDisabled | |
| GetPlayerFightingDisabled | |
| GetPlayerJumpingDisabled | |
| GetPlayerMagicDisabled | |
| GetPlayerLookingDisabled | |
| GetPlayerViewSwitch | |
| GetVanityModeDisabled | |

PCGet3rdPerson returns 1 if in 3rd person mode
PCForce3rdPerson queue the change to 3rd person mode (this may have to wait for the animation to finish)
PCForce1stPerson same as above but 1st person mode

**Player Sleeping**
GetPCSleep Returns true if pc is sleeping.
ShowRestMenu Brings up rest menu, for beds in illegal cels
WakeUpPC Wakes up PC

**Time**
GetSecondsPassed

**Script**
ScriptRunning, ScriptName
StartScript, ScriptName
StopScript, ScriptName

**Sound**
GetSoundPlaying soundID
PlayLoopSound3D, “soundID”
PlayLoopSound3DVP, “soundID”, volume, pitch
PlaySound, “soundID”
PlaySoundVP,“soundID”, volume, pitch
PlaySound3D, “soundID”
PlaySound3DVP, “soundID”, volume, pitch
Say, “file name”, “text” make subject say wav file, only works on animating objects
SayDone returns true if the object is not saying anything.
StreamMusic, “filename.ext”
StopSound, SoundID stops the soundID if it is currently playing in the object.

**Stats**
GetDeadCount, ObjectID ; returns how many of this objID have been killed
GetStat
SetStat, NewValue
ModStat, Mod
Resurrect Brings Actor back to life

**Weather**
ChangeWeather, RegionID, TypeEnum
GetCurrentWeather
ModRegion, RegionID, clear_var, cloudy_var, foggy_var, overcast_var, rain_var, thunder_var, ash_var, blight_var

**Console (in game only commands)**
CenterOnCell (coc), CellID Places the PC in the named cell.
CenterOnExterior (coe), X, Y Places the PC in the exterior cell grid.
CreateMaps "Filename.esp" Creates map image file for Xbox
FillJournal add all entries to journal, takes a long time
FillMap show all the towns on the full map
FixMe Jump 128 units away from where I am now.
GetFactionReaction factionID factionID The faction ids are not optional, works in Console window only
Help Shows shorthand for most commands
Show

ShowVars (sv)
StopCellTest (sct)
TestCells (tc)
TestInteriorCells (tic)
TestModels ( t3d)
ToggleAI (ta)
ToggleBorders (tb)
ToggleCombatStats (tcs)
ToggleCollision (tcl)
ToggleCollisionBoxes (tcb)
ToggleCollisionGrid (tcg)
ToggleDebugText (tdt)
ToggleDialogueStats (tds)
ToggleFogOfWar (tfow)
ToggleFullHelp (tfh) show you ownership and script name
ToggleGodMode (tgm)
ToggleGrid (tg)
ToggleKillStats (tks)
ToggleLoadFade
ToggleMagicStats (tms)
ToggleMenus (tm)
ToggleScripts
ToggleStats (tst)
ToggleSky (ts)
ToggleTextureString (tts)
ToggleWorld (tw)
ToggleWireframe (twf)
TPG Toggle path grid display
SG Show selected actor's group members
ST Show selected actor's target group members.
ShowScenegraph (ssg)

**Moon Phases**

GetMasserPhase
GetSecundusPhase

They both return these values:

0 = MOON_PHASE_NEW (this is the default)
1 = MOON_PHASE_WAXING_CRESCENT or MOON_PHASE_WANING_CRESCENT:
2 = MOON_PHASE_WAXING_HALF or MOON_PHASE_WANING_HALF:
3 = MOON_PHASE_WAXING_GIBBOUS or MOON_PHASE_WANING_GIBBOUS:
4 = MOON_PHASE_FULL

## Inventory Scripts

Certain inventory activities can be checked through scripts using local variables. These local variables are automatically set when the pc does the action. They are NOT functions, but variables that are true or false.

| | |
|---|---|
| OnPCEquip | The PC has the object equipped (remains true while object is equipped) |
| OnPCAdd | The PC added the object to inventory |
| OnPCDrop | The PC dropped the object |
| OnPCRepair | The PC repairs the object |
| OnPCSoulGemUse | the Object is a soulgem and it has been used in either recharging or itemmaking |
| PCSkipEquip | Set this to 1 to skip equipping object. Good for popping up messages for breaking seals on books and such. |

Example script in a sword that gives a message when equipped

```
Begin MyScript

Short OnPCEquip

If ( OnPCEquip == 1)
   MessageBox " You hear the sword in your mind: 'Feed me the souls of the living' "
   Set OnPCEquip to 0

endif

End MyScript
```

## PlaceAtPC

PlaceAtPC, ItemID, count, distance, direction
PlaceAtPC, “Ninja Man”, 1, 256, 1

Places the object at the PC, in the direction you specify and the distance. If that location is not safe (in the air, in a wall, etc), the object will be placed at one of the other axis or at the player’s exact location (feet).

direction is:

```
0 = front
1 = back
2 = left
3 = right
```

## ChangeWeather

ChangeWeather, RegionID, TypeEnum

ChangeWeather, “West Gash”, 4

Changes the weather in the specified region right now. The weather will begin transitioning immediately. It will reselect a new weather on the next weather update (12 hours).

The weather type enum are as follows:

```
0  Clear
1  Cloudy
2  Foggy
3  Overcast
4  Rain
5  Thunder
6  Ash
7  Blight
```

**See also:**
ModRegion, RegionID, clear_var, cloudy_var, foggy_var, overcast_var, rain_var, thunder_var, ash_var, blight_var
GetCurrentWeather

## ModRegion

ModRegion, RegionID, clear_var, cloudy_var, foggy_var, overcast_var, rain_var, thunder_var, ash_var, blight_var

ModRegion, “West Gash”, 10, 20, 10, 5, 5, 40, 10, 0

Changes the weather chances for the RegionID. Used to get rid of, or add weathers to an area permanently. The values must add up to 100 or you will get odd results.

**See also:**
ChangeWeather, RegionID, TypeEnum
GetCurrentWeather

## GetCurrentPackage

GetCurrentPackage

If ( GetCurrentPackage == 2 )
endif

Used for checking the current AI package an Actor is performing.

The return values are:

```
None = -1
Wander = 0
Travel = 1
Escort = 2
Follow = 3
Activate = 4
Pursue = 5
```

## Stat Script Functions

GetHealth
SetHealth, Var
ModHealth, Var

Each of the above can use any stat available to the player or actor (NPCs and creatures). See the list below. “Scale” can be used on any object.

GetStat
Returns the current value of the stat.

SetStat, var
SetHealth, 90
Sets the stat to the new value, var. Note that this will also change the stats maximum value.

ModStat, var
ModHealth, 10
Modifies the stat by var.

Stats that can be used for “Stat”:

```
Strength
Intelligence
Willpower
Agility
Speed
Endurance
Personality
Luck
Block
Armorer
MediumArmor
HeavyArmor
BluntWeapon
LongBlade
Axe
Spear
Athletics
Enchant
Destruction
Alteration
Illusion
Conjuration
Mysticism
Restoration
Alchemy
Unarmored
Security
Sneak
Acrobatics
LightArmor
ShortBlade
Marksman
Mercantile
Speechcraft
```

```
HandToHand
Health
Magicka
Fatigue
Reputation
Disposition
AttackBonus
DefendBonus
ResistMagicka
ResistFire
ResistFrost
ResistShock
ResistDisease
ResistBlight
ResistCorprus
ResistPoison
ResistParalysis
Chameleon
ResistNormalWeapons
WaterBreathing
WaterWalking
SwimSpeed
SuperJump
Flying
ArmorBonus
CastPenalty
Silence
Blindness
Paralysis
Invisible
PCCrimeLevel (PC Only)
Fight (Changing this changes it for ALL references of the Actor)
Flee (Changing this changes it for ALL references of the Actor)
Alarm (Changing this changes it for ALL references of the Actor)
Hello (Changing this changes it for ALL references of the Actor)
Level (only works with Set and Get)
Scale (the 3D scale of the object)
```

Special use with ModStat only:

```
ModCurrentHealth, float
ModCurrentMagicka, float
ModCurrentFatigue, float
```

Special use with GetStat only:

```
GetHealthRatio                  returns float value
```

## ScriptRunning

ScriptRunning, ScriptName

If ( ScriptRunning, MessageTest == 1 )

This function returns 1 if the script you ask for is currently running. This is mainly used for global scripts that are written to run the more complex quests.

**See also:**
ScriptRunning, ScriptName
StartScript, ScriptName
StopScript, ScriptName

## StartScript

StartScript, ScriptName

StartScript, GrandCouncil

This function starts a script running. This is a Global script. It is not attached to any object, so functions like moving, rotating, checking distances and such have no bearing in a global script.

Global scripts are good for running complex quests, checking variables, or setting timers.

**See also:**
ScriptRunning, ScriptName
StartScript, ScriptName
StopScript, ScriptName

## StopScript

StopScript, ScriptName

StopScript, GrandCouncil

This stops a currently running global script.

**See also:**
ScriptRunning, ScriptName
StartScript, ScriptName
StopScript, ScriptName

## Position

Position x,y,z, zRot
Position 76345, 100.56, 215, 176

Warps the object to the exterior coordinate location specified by x, y, and z. The object will be facing the world angle specified by zRot. All parameters are floats.

## PositionCell

PositionCell x,y,z, zRot, “CellID”
PositionCell 76345, 100.56, 215, 176, “Gnisis Barracks”

Warps the object to the interior cell coordinate location specified by x, y, and z. The object will be facing the world angle specified by zRot. All parameters are floats. The interior cell name must be put in quotes.

## CellUpdate

CelUpdate

Updates the current objects cel position. This should be called when moving objects over large distances. The game keeps tracks of objects based on what cell they are in, and if an object moves a cell over from its starting position, it may not get processed correctly when running its script.

## GetPos

**GetPos, axis**
GetPos, Z

Returns the current world position of the object on the selected axis (X, Y, or Z) in float.

## GetAngle

**GetAngle, axis**
GetAngle, Z

Returns the current world rotation angle of the object on the selected axis (X, Y, or Z) in float.

## GetStartingPos

GetStartingPos, axis
GetStartingPos, Z

Returns the original world position of the object on the selected axis (X, Y, or Z) in float. The position is the position the object was placed in the editor, not it's current game state.

## GetStartingAngle

GetStartingAngle, axis
GetStartingAngle, Z

Returns the original world rotation angle of the object on the selected axis (X, Y, or Z) in float. The angle is the angle the object was placed in the editor, not it's current game state.

## GetSecondsPassed

GetSecondsPassed

Returns the number of seconds passed since the last game frame (in float). Extremely useful for running a timer within a script.

## PlayGroup

PlayGroup GroupName, [flags]
PlayGroup Walk
PlayGroup Walk, 1

Plays the animation group defined by GroupName. Optional flags can be used to start the group in different ways.

**Flags**
*0 = Normal*
The current animation will finish it's full cycle, and the new animation will start from its beginning.

*1 = Immediate Start*
The current animation will stop regardless of the frame it is on, and the new animation will start from its beginning.

*2 = Immediate Loop*
The current animation will stop regardless of the frame it is on, and the new animation will start at the beginning of its loop cycle.

**See Also**
LoopGroup GroupName, Number, [Flags]
SkipAnim

## LoopGroup

LoopGroup GroupName, 2, [flags]
LoopGroup Idle2
LoopGroup Idle2, 1

Plays the animation group defined by GroupName. The animation will be looped the number of times specified, and then return to the Idle animation. Optional flags can be used to start the group in different ways.

**Flags**
*0 = Normal*
The current animation will finish it's full cycle, and the new animation will start from its beginning.

*1 = Immediate Start*
The current animation will stop regardless of the frame it is on, and the new animation will start from its beginning.

*2 = Immediate Loop*
The current animation will stop regardless of the frame it is on, and the new animation will start at the beginning of its loop cycle.

**See Also**
PlayGroup  GroupName, [Flags]
SkipAnim

## SkipAnim

SkipAnim

Causes the current animation to not be played for this frame.

## Move

Move  axis, units/sec
Move  x, 100

Moves the object along the selected axis (x, y, or z) at the speed selected. This speed is in units per second (21.3 units per foot). This movement is based on the object's local rotation. Thus, a positive y movement will always move the object along its local forward vector.

**See Also**
MoveWorld

## Rotate

Rotate  axis, degrees/sec
Rotate  x, 100

Rotates the object along the selected axis (x, y, or z) at the speed selected. This speed is in degrees per second. This movement is based on the object's local rotation. Thus, a positive y movement will always rotate the object along its local forward vector.

**See Also**
RotateWorld

## RotateWorld

RotateWorld axis, degrees/sec
RotateWorld x, 100

Rotates the object along the selected axis (x, y, or z) at the speed selected. This speed is in degrees per second. This movement is based the world axis.

**See Also**
Rotate

## MoveWorld

MoveWorld axis, units/sec
MoveWorld z, 100

Moves the object along the selected world axis (x, y, or z) at the speed selected. This speed is in units per second (21.3 units per foot). This movement is based on the world axis, thus a positive z movement will always move the object up, regardless of it's local rotation.

**See Also**
Move

## GetPlayerDistance

GetPlayerDistance

This will return the distance in units (21.3 units per foot) from the player to this object reference.

## Script Commands

```
begin    ScriptName
end      ScriptName

; comment

short    varNameShort
long     varNameLong
float    varNameFloat

set  varName to 5
get  varName

while ( varName <= 3 )
endwhile

if ( varName > 100 )
elseif ( varName < 20 )
else
endif

if ( varName == x )
if ( varName != x )
if ( varName >= x )
if ( varName <= x )

set objectID.varName to 100
objectID ->rotate, z, 45
```

**See Also**

Using Variables in Functions

## Using Variables in Functions

Only certain functions can take variables as their arguments. These include:

HurtStandingActor, float
Journal "JournalID" value
SetStat, NewValue
ModStat, Mod

## Declaring Variables

You can declare three types of variables through scripts and as global variables.

| | |
|---|---|
| short | -32,768 to 32,767 |
| long | -2,147,483,648 to 2,147,483,647 |
| float | 3.4E +/- 38 (7 digits) |

**Reserved variables for companions**

*short stayOutside*

When this value is 1 the followers in the PC's group will wait outside the interiors for him to come back out.

*short companion*

When this value is 1 you will be given a companion inventory menu selected by dialogue. If the creature doesn't talk, he is alive, and you activate him you will be taken directly to the companion inventory menu.

*float minimumprofit*

This is how much of the PC's stuff this bounty hunter is carrying. The value gets modified every time an item is dropped on or removed from the companion inventory menu.

begin ScriptName

The begin command must be the first command in the script. It must be followed by an end command at the end of the script.

end ScriptName

The end command is placed at the end of a script.

## Object Referencing

Other object Ids can be used in scripting to set and get the values of local variables from other objects in the world. You can also run functions on these object references.

There are two forms of object references:

```
Set MyObject.variable to 100
```

This method changes a local variable in the object's script. The object must have a script on it for this to be valid.

```
MyObject->rotate, z, 45
```

This method tells the object to perform a function. The object does not need to have a script on it.

You can also use this method in an if statement:

```
If ( MyObject->GetHealth > 50 )
```

You can also reference scripts (very useful for changing global script variables).

NOTE: The scripting system looks at the *first* object in the database, thus you should only reference objects that are somewhat unique.

NOTE: You can only use ONE object reference per script line.

## System Requirements

Win95/98/2000/Me, 128 MB RAM, and hard drive space for game data files. It is recommended that it be run at 1024x768 or higher.

Direct3D is used for all 3D rendering. Since the Construction Set renders in a window, only those video cards that support accelerated window rendering are recommended. It also uses hardware acceleration for bilinear filtering, transforms, and lighting.

A 3D accelerated video card is not necessary to run the Construction Set, but it is highly recommended.

You can obtain better performance from the rendering a few ways:

1. Buy a better video card.
2. Decrease the size of the render window.
3. Turn Mip-Maps off in Properties.
4. Turn down the render distance in Properties.
5. Make sure the fogging option is OFF.
6. Buy a better video card.

If you have done a Maximum install of the Construction Set, you may notice it (and *Morrowind* the game) take longer to load. This is because both programs will use the raw version of the art, instead of compressed artwork. This takes longer to load.

**See Also**
Preferences

## Preferences

**Grid Snap / Use Grid**
When Use Grid is checked, objects shown in the Render Window move only by the number of units in the "Grid Snap" field. This can be helpful when lining up room pieces. Multiples of 8 are effective with the original objects (32, 64, 128).

**Angle Snap / Use Angle**
When "Use Angle" is checked, objects in the Render Window will rotate as many degrees as are specified in the "Angle Snap" field. Again, this is helpful when attempting to line up many objects.

**Cell Load Settings**
"Allow Render Window Cell Loads" will load the full cell every time the user selects an object in a cell adjacent to the exterior cell currently being worked on.
"Skip Initial Cell Load" will prevent the editor from loading the first cell in the list into the Render Window when an esm/ess file is loaded.

**Movement Speeds**
Numbers can be entered to change the speed at which the user rotates and moves both the camera and objects. The default values are 1.00.

**View Settings**
The slider bar in this section controls how much of the loaded cell will be visible in the Render Window. The farther towards "far" the slider is set, the more can be seen from any given camera position, which may slow down some systems. The "Renderer Settings" button brings up options for display device used by the Editor.

**Auto Save**
When checked, this option automatically backs up the user's work on a regular basis, as specified by the number of minutes entered. This backup file is called AutoSave.bak and is a regular plugin. Just rename it to restore your work.

## Cell Overview

Think of a cell as a mini-map. The entire game is made up of cells, for a huge continuous world. As the player moves around, cells are loaded and unloaded from the game.

There are two types of cells, exterior cells and interior cells.

**Exterior Cells**
The exterior cells are basically a grid that extends infinitely in all directions.

**Interior Cells**
Interior Cells are entered through the exterior. When in an interior, the exterior world is no longer visible. Each interior cell is essentially it's own universe.

**See Also**
Naming Cells
Interior Cell
Exterior Cell

## Naming Cells

All cells have Names. These names are used for:

1. The name of the player's location in the local map window.
1. The name of an NPC's location for dialogue sorting.
1. The name of the cell as it appears on the large world map.

Exterior Cells can, and should share names. The large city of Balmora may be made up of 4 cells. Each cell should be named “Balmora”, so that the name is not flashing around on the map screen. Also, if dialogue is attached to everyone in “Balmora”, this dialogue can be filtered easily for multiple cells without having to define them all as separate criteria.

Interior Cells that are supposed to be in an area (such as a tavern in Balmora) should be given names starting with the city, such as “Balmora, North Tavern”. With this name, everyone in the tavern will be a part of Balmora for dialogue, but the map will also read “Balmora, North Tavern” when the PC walks in it. Remember, **all interiors must have unique names.**

## Exterior Cell

The exterior cells are basically a grid that extends infinitely in all directions. Each exterior cell is 8192 units by 8192 units or 385 feet by 385 feet.

All exterior cells are given a name, but it **does not have to be unique**. This is the name that will appear in the player's map window as they walk around the world. Some cities and towns will exist over several cells, and we don't want the name of the cell changing as they walk across the city. All exterior cells are named based on the region they are part of by default.

All exterior cells are automatically given a grid number that represents their X,Y position from the center of the world. You will see this in the render window title as you select objects.

Settings are also available for Weather Type, sleeping, and designation on the World Map using a colored border.

## Interior Cell

Interior Cells are entered through the exterior. When in an interior, the exterior world is no longer visible. Each interior cell is essentially it's own universe.

Each interior cell gets a **unique name**. Interior cells also are *not* affected by weather or time of day, and all have their own settings for ambient, sun, fog colors, water level, and sleeping.

## Regions

The Regions window allows the user to set and edit different regions, which are areas of the exterior world with unique weather and ambient sounds.

**Weather Chances**
This section contains the percent chances that the different weather types will occur in the region. Any weather types set to 0 will never occur.

**Sleep Creature**
Whenever you rest in an exterior cell there is a percent chance that your rest will be interrupted by a creature. The Leveled Creature selected in this field is what will be used to calculate what creature awakes you.

**Region Map Color**
Different colors can be assigned for the Region Map, which can be accessed by clicking on the “Region Paint” button. This brings up a new window showing the exterior world. Highlighting a region in the list on the right brings up the region’s information below; clicking on exterior cells on the map to the left then includes those cells in that region. The cells’ color on the map will change to match the region color.

**Ambient Sounds**
The Sound files are listed in the table on the right side of the window, and can be filtered by three fields:

*Chance:* This is the percent chance the sound will play.

*Priority:* If two sounds are called at once, the sound with the higher priority will play.

*Sound ID:* the ID of the sound file.

## World Testing

Several selections are provided for quick testing of a plug-in without needing to actually load the game. The editor will perform a scan of all world content, and return a list of possible problem areas in the following categories:

**Model Test**: Runs through all objects in the file to verify that all have valid 3D.

**Dialogue Conflicts**: Returns a list of dialogue topics that share an ID with something else in the world, which could cause problems with the replaceID function in scripts.

**Non-teleporting Doors**: Returns a list of all exterior doors that are not linked to anything.

**Non-teleporting Interiors**: Returns a list of all interior cells that are not linked to either another interior or the exterior world.

**Test All Land**: Returns a list of exterior cells that have border seam problems.

**Test/Fix All Path Grids**: Runs through all path grids, removing points underwater and points without any connections. Also moves underground points to ground-level.

**Test All Cells**: Runs through all the cells in the world, moving the camera and re-drawing to check for animation errors.

**Test All Interiors Cells**: This performs the same function as Test All Cells, but only on interiors.

## Data Files

The Construction Set works off of a primary file called a TES file. This is basically a database of all of the data for the world, including object data, dialogue, gameplay settings, object placements, AI settings, landscape, script commands, cells, etc. Basically everything that goes into the editor is stored in TES files. One giant database.

There are two types of TES files, masters (esm) and plug-ins (esp). A master file is autonomous. It relies on no information other than itself. A plug-in (esp) file relies on information from a Master TES file. *Plug-ins CANNOT refer to information in another plug-in. They can only refer to data from a master.* Plug-ins can refer to multiple master files.

Most TES files are changes to the main Morrowind.esm file (a master file). The files get loaded in order of their last modified date. Masters get loaded first, by date. Then plug-ins get loaded by date. Each file is a "layer" that gets added to the world. The player can turn off the ones they don't like with the loader program, and these layers will be removed. This makes it impossible for a player created TES file to totally break someone's game

Users of The Construction Set will have all of their changes placed in an esp file. They can then allow someone else to load this file and play with their changes. They can add anything to the game, provided the artwork is there to support it.

The editor can have many esp files loaded at once, but only one of these is *active.* Active means that all of the user's changes are being saved into the *active* TES file.

Notes on Active files:

- You can only make a plug-in (esp) file active. You cannot save changes to a master file.
- The active file is always loaded last in the editor, regardless of its file date. This guarantees that all the info and changes in the active file can be viewed without being overwritten by another plug-in.
- If only an esm file is being loaded, an Active file does not have to be assigned. Any changes made to the master file will be saved as a new esp file, which the user will have to name. This new file will become the active file.

**See Also**

Data Files Example
Data Files Window

## Data Files Example

Here is an example of multiple data files and what happens, as they are loaded.

| File Name | Summary | Result |
|---|---|---|
| Morrowind.esm (Master) | The full game data. | Looks like shipped game. |
| Samurai.esm (Master) | Adds the samurai class to the game, which can be chosen by the player. | If player starts a new game, can now choose samurai. |
| Stormbringer.esp | Adds the magical sword Stormbringer to the game, locked behind an existing door in lower Vivec. | Adds the sword, changes the lock level on a door. |
| Lost Boys.esp | Small quest to find two NPCs in Vivec. Changes lock level on door, and adds key for the door. | Coincidentally changes the same door as “Stormbringer”, and adds a key to it. There are now 2 NPCs in the room with the sword. |
| Samurai Clan.esp | Adds several Samurai warriors to the hills in the north. | Adds the warriors provided the Class has already been added. This one relies on the master file Samurai.esm and Morrowind.esm |
| Min’elok Mines.esp | Adds a decent sized dungeon mine to the world, with monsters, items, and dialogue. This mine is built into one of the existing cave entrances, and the smaller cave pieces have been removed. | Existing mine pieces removed, new objects and now in the world there. |
| Mine of Monkeys.esp | Small mine built with modified race called ninja monkeys. | This (coincidentally) removes the same mine as Min’elok, and adds new pieces. But since it does not touch the objects placed from the previous one, this section of the world is very messed up, with both mines existing in the same space. |

Overall it is vitally important that external creators document their important changes in the summary of their TES file.

**See Also**
Data Files Window
Data Files

## Data Files Window

Lets the creator decide what TES' are loaded and which is the active one.

*TES Files.* This is a list of all of the TES files in the Morrowind\Data Files directory. The checked ones are the ones currently loaded. Double Clicking on one will change its state from loaded to unloaded. The Active esp file is shown as "Active File" in the Status column.

*Set as Active File:* Makes the currently selected file active.

*Merge to Masters:* Combines the active esp file to its masters. It also prompts you for the file that any new data should go into. After a file is merged, it is deleted.

*Details:* Opens a detail window for the currently selected TES file showing the actual chunk and form data that is contained. So if a new sword ID is added it will be listed, or if one is deleted, etc.

*Created By:* Creator's name

*Summary:* A summary of what the TES file does and perhaps how some of the more complicated sections work.

*Parent Masters:* displays the master files the currently selected file is based on.

**See Also**
Data Files Example
Data Files

## Landscape Overview

When viewing the exterior world of Morrowind, landscape is always present. It goes on to infinity, and is present even below the endless sea. Landscape works differently than other game objects, though the game treats them similarly.

The landscape is a height map, with vertices every 128 units. When you edit an exterior cel, which has a size of 8192x8192, you can also edit the landscape in that cel. Changes to the landscape are stored on a cel by cel basis. Thus, one plug-in may edit one part of the world, and other plug-ins, other areas.

The landscape may be edited on the Z axis only (up and down). It can also be textured using the Construction Set's library of landscape textures. Lastly the landscape can also be colored, using vertex colors to create shadows or extra paths. Thus the landscape has 3 modes of editing (height, texture, color).

Water is at height 0. All landscapes start at height of –2000.

Press **H** to enter Height editing mode. In this mode you can edit the heights and textures. Press **O** to use vertex colors on the landscape.

**See Also**
Landscape Texturing
Landscape Shaping

## Landscape Texturing

The landscape is also textured in the editor. Most landscape textures are 256x256 and cover 8 tiles by 8 tiles or 512 x 512 units (one landscape tile is 128 units).

To edit the landscape textures, select the Toolbar Icon for Landscape Editing, or press **H**. To disable Landscape Texturing press H again.

When texturing you select a texture to paint with and fill in the areas you want. Textures can only be placed following the grid above, so textures are automatically placed every 8 tiles.

To paint the currently selected texture on the landscape, **right click** on the landscape tile you wish to change. You can also **hold the right mouse button** to paint a larger area. Note: You only paint one tile at a time, regardless of the edit radius.

When painting a texture, the eight textures that are connected to the tile being painted will change automatically, so that the textures blend into one another. This allows you to paint with grass and the Construction Set will dynamically blend the textures around it so that grass blends into the sand smoothly.

You can add textures to the library of landscape textures the Construction Set uses. To do this select the texture window and hit the insert button on the keyboard. A texture may then be chosen and given an ID.

The Vertex Color menu enables you to paint color onto the landscape in addition to the textures. Press **O** to enable vertex coloring. By left clicking you will paint the landscape with the left selected color. The **right mouse button** will paint the right selected color. Note: the default right selected color is white and can be used to erase vertex coloring. The edit radius may be set to change the amount of vertices that are being colored. Custom color values may be stored by clicking on the select color button, selecting one of the 16 slots, and then hitting add to custom colors.

**See Also**
Landscape Shaping

## Landscape Shaping

To shape or edit the height of the landscape, either select the toolbar button or press **H**. **H** switches height editing on or off.

Once you are in height editing mode, the pointer in the render window will select vertices on the landscape as opposed to objects.

**Clicking and holding** on a vertex allows you to move it. **Drag the mouse up and down** to change the height.

You can edit the speed of vertex movement in the properties menu.

*Edit Radius:* This is the radius (in vertices) that will be changed as you pull the vertices up and down. A larger radius will create larger hills or valleys.

*Edit Falloff %:* This is the amount the vertices move based on how much you move the vertex you select, within the radius. A value of 0 will create a linear (straight) falloff, while 100 will create a curved falloff.

*Flatten Vertices:* When this option is selected, all vertices in the edit radius will be set to the height of the fist vertex you clicked on as you paint.

*Soften Vertices:* With this selected, all vertices in the edit radius will be set to the average height of the surrounding eight vertices and the center vertex. In effect this will smooth the shape of the landscape.

*Show Edit Radius:* This simply displays the red edit circle in the render window or hides it.

All Landscape starts at a height of –2000 units. Sea level is 0. Anything below 0 will be underwater. Thus when building a new area, you must pull the landscape up above the water.

**See Also**
Landscape Texturing

## Object Overview

The entire game is built using objects. They are the set of items and objects that make up the world. The object types are:

- Activator
- Alchemy
- Apparatus
- ArmorBody Part
- Book
- Clothing
- Container
- Creature
- Door
- Enchanting
- Ingredient
- Leveled Creature
- Leveled Item
- Light
- Lockpick
- Misc Item
- NPC
- Probe
- Repair Item
- Spellmaking
- Static
- Weapon

## Static

Statics are objects that only serve as general building objects, This includes buildings, tunnel walls, posts, etc.

*ID*: The object's unique ID.

*References Persist*: If checked, the object will remain persistent in world.

*Add Art File*: Art used for object.

*Blocked:* If checked, the specific object becomes read-only.

## Weapon

Weapons are any object that is used for attacking in the game. They are equipped as weapons and include long swords, bows, axes, and ammo (such as arrows and bolts).

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

Type: Type of weapon for Skill and hand position.

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Health:* How much damage the weapon can take before it breaks. This is the maximum health. References to this object in the world can have a lower health setting for placing used items in the world.

*Value*: The object's value in gold.

*Speed:* Attack speed of the weapon. This is the multiplier used for swinging the weapon. A value of 2.0 will swing the weapon twice as fast as normal, and 0.5 will swing it at half speed. In general small weapons like daggers are given 1.5, and big weapons, like hammers, are given 0.8.

*Reach:* Multiplier used to adjust how far away the weapon hits. Not used for ranged weapons.

*Enchantment*: The number of enchantment points the item has. Used for enchanting.

*Enchanting*: The enchantment attached to the object.

*Ignores Normal Weapon Resistance*: If checked, the weapon can hit creatures with the "Immune to normal weapons" ability.  Should be checked for everything silver and better.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Damage*: Min and Max damage for the three types of attacks. Ranged weapons only use the Chop damage.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Weapon Types

Weapon types define both what skill is used, how the weapon is held, or if it is ammo. The types are:

**SHORT_BLADE_ONE_HAND** (Daggers, shortswords, tantos)
**LONG_BLADE_ONE_HAND** (Broadswords, longswords, katanas)
**LONG_BLADE_TWO_CLOSE** (Claymores, Dia-katanas)
**BLUNT_ONE_HAND** (Maces, clubs)
**BLUNT_TWO_CLOSE** (Warhammer)
**BLUNT_TWO_WIDE** (Staffs)
**SPEAR_TWO_WIDE** (Spears, halberds)
**AXE_ONE_HAND** (war axes)
**AXE_TWO_CLOSE** (battle axes)
**MARKSMAN_BOW** (shortbows, longbows)
**MARKSMAN_CROSSBOW** (crossbows)
**MARKSMAN_THROWN** (knives, darts, stars)
**WEAPON_ARROW** (arrows, for bows)
**WEAPON_BOLT** (bolts, for crossbows)

## Armor

Defines a new Armor piece. Armor also acts as an object that replaces body parts for NPCs.

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Type*: The type of armor this piece covers. Includes: Shield, Boots, Cuirass, Helmet, Left/Right Bracer, Left/Right Gauntlet, Left/Right Pauldron

*Script*: The script assigned to object.

*Weight*: The weight of armor (in pounds).

*Health*: The object's health. When zero, the armor becomes unusable and needs repair.

*AR*: The object's armor rating. The higher the number, the better the protection.

*Value*: The object's value in gold.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Enchantment*: The number of enchantment points the item has. Used for enchanting.

*Enchanting*: The enchantment attached to the object.

*Biped Object*: The body part for clothing piece. See Body Part.

*Male Armor/Female Armor*: The corresponding body part ID created for armor piece.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

**See Also**
Character Body Layout

## Clothing

*ID*: The object's unique ID.  (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Type*: The type of clothing. Also used as the base name.

*Script*: The script assigned to object.

*Weight*: The weight of armor (in pounds).

*Value*: The object's value in gold.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Enchantment*: The number of enchantment points the item has. Used for enchanting.

*Enchanting*: The enchantment attached to the object.

*Biped Object*: The body part for clothing piece. See Body Part.

*Male Armor/Female Clothing*: The corresponding body part ID created for clothing piece.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

**See Also**
Character Body Layout

## Character and Creature Overview

Characters and Creatures provide the backbone for the entire game experience. They also contain most of the gameplay information. This includes AI, combat, and dialogue.

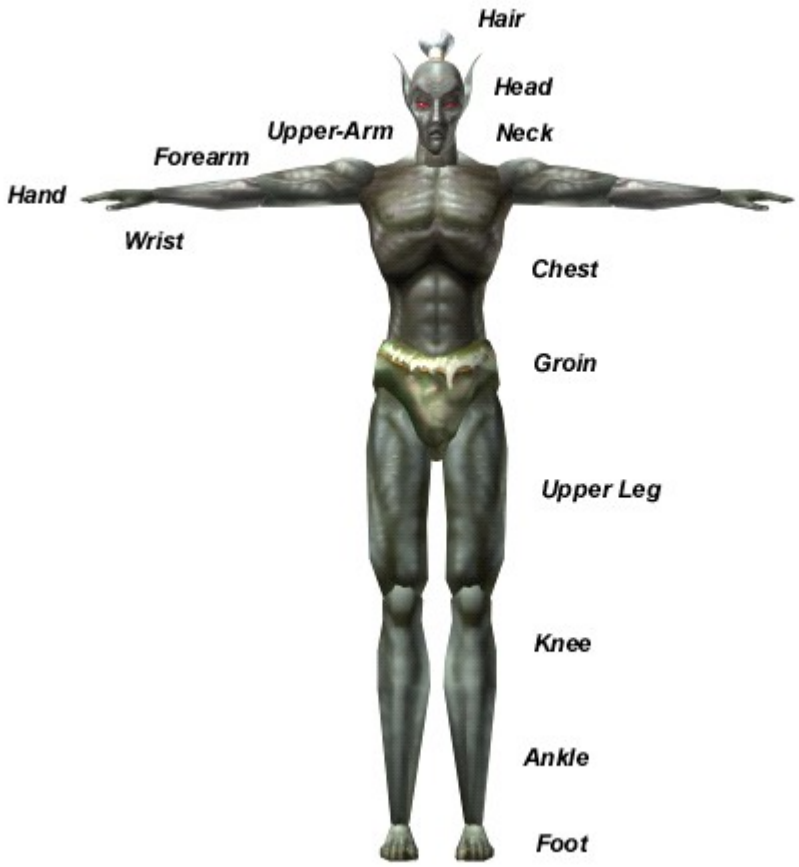
The combined set of Characters and Creatures are often referred to as Actors.

All Actors can perform the same basic actions like combat, moving, magic, etc.

Here are the major differences:

| Item | Character | Creature |
|---|---|---|
| **Animation** | All use the same file, called the base_anim. | Most use their own animation file specific for themselves and can only perform the actions animated for them. Creatures tagged as “biped” can use animations in the base_anim. |
| **Movement** | All characters can walk and swim | Can be tagged to walk, fly, swim, or not move. |
| **Dialogue** | All characters have dialogue by default | Creatures only have dialogue if you specifically tag it to them. |
| **Stats** | Characters have race, class, factions, etc. | Creatures do not have a race, class, or faction. |
| **Skills** | Characters get all 27 skills. | Creatures have only 3 skills, that encompass the whole set of 27. |
| **Items** | Characters can use all their items, and will select what weapons to use and clothes to wear. | Creatures cannot use their inventory. “Biped” creatures can use weapons and shields, but nothing else. |

## Character Body Layout




| Armor Type | Usually Covers |
|---|---|
| Helmet | Hair (and Head) |
| Cuirass | Chest |
| Pauldron | Pauldron and/or Upper Arm |
| Skirt | Groin |
| Greaves | Groin, Upper Leg |
| Boots | Foot, Ankle |
| Gauntlet | Hand, Wrist |
| Bracer | Wrist |
| Shield | |

| **Clothing Type** | **Usually Covers** |
|---|---|
| Pants | Groin, Upper Leg, Knee, Ankle |
| Shoes | Foot, Ankle |
| Skirt | Groin and Upper Legs |
| Shirt | Chest, Upper Arm, Forearm, Wrist |
| Belt | Not Rendered |
| Robe | All but hands, head, feet, and ankles. |
| Ring | Not rendered |
| Amulet | Not rendered |

## Container

Containers are unmovable objects in the world that hold or contain other objects. This includes chests, crates, dressers, bags, and even bushes. Objects can be placed in and removed from containers. Bushes and such are referred to as *organic containers*. These containers have contents such as berries and leaves. They cannot have objects put into them in the game, only removed.

*ID*: The object's unique ID. (character limit: 24)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: Opens script based window for specific flag setting either before or after the container is opened.

*Organic Container*: If checked, container is flagged as organic and cannot have objects placed into them, only removed.

*Respawns*: If checked, objects inside container re-appear after period of time. (4 months)

*Weight:* Containers cannot be moved, so this number represents the total weight the container can hold.

*References Persist*: If checked, the object will remain persistent in world.

*Animation*: Animation used for object.

*Add Art File*: Art used for object.

*Inventory List*: A list of any object the container holds. Items of a negative value are infinite.
- *Count*: Number of that item in container.
- *Object ID*: The object's unique ID
- *Type*: Type of object

*Blocked:* If checked, the specific object becomes read-only.

## Object Reference

Any object placed in the world is known as a *reference.* This is because it references an object in the object database (object window). So an *object* is an ID in the database, and a *reference* is a copy of that object. If the object is changed, all it's references are also changed. A reference can also have it's own unique data:

**Editing Reference Data**
Some objects include reference data, which appears in the bottom half of the object properties window. Editing this information will affect only the selected reference(s); other references of that object will not be changed. There are five types of reference data:

**Position**: the reference's position and orientation in the world.

**3D Scale**: The size of the object reference. Objects can be scaled from 50% (0.5) to 200% (2.0) of their normal size.

**Extra Data**
*Health / Uses:* Used for objects such as weapons and armor that have health and objects such as lockpicks and probes which have uses. Used for creating damaged or used references.

*Soul:* Only used for the misc Soul-Gems. Select a creature to place in the soul-gem reference.

*Apply to Selection:* Applies the Extra Data settings to **all** references currently selected. This is a quick way to assign ownership to all objects in a cell.

**Ownership**
*Owner:* Set which NPC or Faction owns the selected object.

*Global Variable/Rank:* Select the global variable or rank (if the owner is a faction) that allows the PC to use this object without committing a crime.

*Apply to All Selected*: Applies the ownership data and variable to all the selected objects. Very good for shops or guilds where many objects need ownership.

**Locked**
Locks a container or doors.

*Lock Level*: The Difficulty rating on the lock for trying to pick it with a lockpick. Should use values of 1 to 100. 1 is the easiest, while 100 is super-hard (the player would need a security skill of around 100).

*Key:* The item that can be used to open the container. Select from list of all Misc Items.

*Trap:* A trap that will go off when the container is opened. You can select from the list of available spells that have touch based effects.

**Teleport**
Used only on doors, this sets the door to teleport the user to a new map or area.

*Load Cell*: Select the cell that the door will teleport you to.

*Select Marker:* Selects the marker in the new cell so it can be placed.

Note that any options not available for a particular reference will be grayed out.

## Activator

An activator is any object that does something in the world, but cannot be picked up. Signs, flags, levers, pressure plates, etc. It may give off sound, flash lights, and any number of effects. Activators can also be given animation.

An activator will also show up in the world with a text label if it is given a “name”. It can still be activated (provided the script is written for this case).

A popular use of the name field for an activator is to label signs and such.

*ID*: The object’s unique ID. (character limit: 32)

*Count*: Number of times the object is placed in the game world

*Name:* The object’s name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Add Art File*: Art used for object.

*Animation:* Animation used for object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Door

*ID*: The object's unique ID. (character limit: 32)

*Script*: The script assigned to object.

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*References Persist*: If checked, the object will remain persistent in world.

*Animation:* View the data on the object's Animation. Greyed out if it has none.

*Add Art File*: Art used for object.

*Open Sound ID*: Sound file used for first activating door.

*Close Sound ID*: Sound file used for closing door.

## Book

Books and scrolls are objects that contain stories and information. They are opened when activated by the player.

*ID*: The object's unique ID. (character limit: 32)

*Name:* The name of the book. (character limit: 32)

*Script*: The script assigned to object.

*Weight:* The object's weight (in pounds).

*Value:* The value in gold of the item.

*Teaches*: A book can teach a single skill point to the PC. It acts just like training, but the benefits are only given once.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Scroll*: If checked, book appears as scroll. If unchecked, book appears as a book.

*Enchantment*: The number of enchantment points the item has. Used for enchanting.

*Enchanting*: The enchantment attached to the object. Can only attach "Cast Once" type enchantings and only to scrolls.

*Book Text*: In order for the text to appear as viewable in game, the book text uses a simplified HTML notation. To see an example of how the tags work, view the *Morrowind Books HTML.htm* file in your Morrowind\Data Files\BookArt directory. Also view the source code on that file through your browser. (character limit: 64,000)

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Ingredient

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Value*: The object's value in gold.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Effects:* Up to four effects can be assigned to an ingredient. The first effect is what is applied if the ingredient is eaten.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked*: If checked, the specific object becomes read-only.

## Apparatus

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Type*: An apparatus can be one of four types: Alembic, Calcinator, Mortar/Pestle, Retort

*Script*: The script assigned to object.

*Weight:* The object's weight (in pounds).

*Value:* The value in gold of the item.

*Quality:* The quality multiplier used when making a potion with this apparatus.

*Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Lockpick

Lockpicks are items used by thieves to unlock locked doors or containers. Like many other items, they also have a quality level multiplier.

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Uses*: The number of times a lockpick can be used before it disappears.

*Value*: The object's value in gold.

*Quality*: Multiplier that increases your chances of successfully picking lock.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Probe

Probes are items used by thieves to disarm traps on doors or containers. Like many other items, they also have a quality level multiplier.

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Uses*: The number of times a probe can be used before it disappears.

*Value*: The object's value in gold.

*Quality*: Multiplier that increases your chances of successfully disabling trap.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Misc Item

Miscellaneous Items are those mundane items that do not have specific use to the player, such as keys, forks, plates, cups, baskets, special rocks, etc. Their structure covers the most common set of object data.

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Value*: The object's value in gold.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Body Part

Body parts are the base pieces used to create NPCs. This includes all the objects used for a race's skin as well as armor and clothes.

*ID*: The object's unique ID. (character limit: 32)

*Part*: The body part represented. Includes: Ankle, Chest, Clavicle, Foot, Forearm, Groin, Hair, Hand, Head, Knee, Neck, Tail, Upper Arm, Upper Leg, Wrist

*Part Type*: Whether the piece is part of an NPC (skin), for clothing or for armor.

*Female*: If checked, the body part is for female. If unchecked, male.

*Playable*: Used for heads and hairs. If checked, the player can choose this head/hair during character generation.

*Add Art File*: Art used for object.

*Skin Info*: If Party Type is Skin, select the Race and the skin type, Normal or Vampire.

*Blocked:* If checked, the specific object becomes read-only.

Note: For all skin body parts but the Head and Hair, only one type can be defined per race and sex.

**See Also**
Character Body Layout

## Light

Lights are objects that give off light, and may be rendered as 3D objects, but affect the lighting on other objects.

There are three main types of light objects:

Light Object that has no 3d object attachment( ex, light coming in from a magical hole)
Light Object that has 3d object, but cannot be picked up (ex, large candelabra )
Light Object that has 3d object, and can be carried. (ex, candle or torch )

*ID*: The object's unique ID. (character limit: 32)

*Radius:* The radius in which the light should effect. Any object within the radius will get effected by the light.

*Script*: The script assigned to object.

*Can Carry*: Flag this if the light can be carried. Art for lights that can be carried must be centered where the character's left hand is.

*Off by default*: For lights that can be carried. This will make the light **not** affect the world when it is placed down. Useful for an unlit torch the player can find. When the player puts it down, it will be lit.

*Name*: The object's name. Should only be used for lights you can pick-up. This is what appears in the game to player. (character limit: 32)

*Inventory Image*: Select the Targa file for icon representation of object.

*Weight*: The weight of object (in pounds).

*Time:* The time (in seconds) that the light will burn when *equipped by the player.* The time does not run when on an NPC or placed in the world.

*Value*: The object's value in gold.

*Fire*: Use this flag if the 3D object used for the light has particles attached to it. This tells the game to stop the particles when the fire time is used up.

*Negative*: If checked, this makes the light a "dark". The color you selected for the light will be removed from the world, not added. So a white negative light will act like a black light.

Dynamic: If checked, This tells the light to light up any dynamic or moving objects (such as characters and creatures). Dynamic lights slow the game down a little more.

*Add Art File*: Art used for object.

*RGB, Select Color:* Choose the color the light adds to the world.

*Flicker Effect*: There are four flicker effects to choose from.

*Flicker: M*akes the light "flicker", or randomly change its brightness. Use for lights that should act like fire.

*Pulse*: Makes the light consistently change its brightness
*Flicker Slow*
*Pulse Slow*
*None*

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

## Water Layer

All exterior cells have a water layer that cuts through them at a height of 0. This is the ocean.

Interior cells can have a water layer as well, and can also have the height specified in the properties for that cell.

The water layer is the only type of liquid that characters can swim in.

Lava or other visual water is defined as activators.

## Repair Item

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Weight*: The weight of object (in pounds).

*Uses*: The number of times the item can be used to repair something before it is used up. Once the item is used up, it is removed from the world.

*Value*: The object's value in gold.

*Quality*: The item's quality multiplier. Used with the player's Armorer ability for the amount the item is repaired each time. A value of 0.5 halves how good the player can repair with it, while 2.0 doubles it.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*References Persist*: If checked, the object will remain persistent in world.

*Blocked*: If checked, the specific object becomes read-only.

## Spellmaking

*ID*: The object's unique ID. (character limit: 32)

*Full Name*: The name of the spell. This is what appears in the game to player. (character limit: 32)

*Effects*: The magic effects assigned to this spell.

*Cost*: Number of magicka points required to cast spell.

*Range*: How far away the spell is effective.

*Area*: The area the spell is in effect.

*Duration*: How long the spell remains in effect.

*Magnitude* (min/max): The strength of the spell.

*Type*: Spell types include: Abilities, Blight Disease, Disease, Curse, and Powers.

*Auto Calculate Cost*: Automatically calculates the cost and value of the spell. Any spell that is auto-calculated may also be used by auto-calculated NPCs.

*PC Start Spell*: If checked, this spell can be given to player at start of the game.

*Always Succeeds*: If checked, the spell will never fail when cast.

*Blocked:* If checked, the specific object becomes read-only.

## Alchemy

*ID*: The object's unique ID. (character limit: 32)

*Name*: The object's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to object.

*Add Art File*: Art used for object.

*Inventory Image*: Art used for icon representation of object.

*Weight*: The weight of object (in pounds).

*References Persist*: If checked, the object will remain persistent in world.

*Blocked:* If checked, the specific object becomes read-only.

*Effects*: The effects associated with the potion.

*Duration*: How long the potion effect lasts.

*Magnitude*: The strength of the effect.

*Auto Calculate Value:* Automatically calculates a value for the potion.

*Potion Value*: The potion's value in gold.

## Leveled Item

*ID*: The object's unique ID. (character limit: 32)

*Calculate from all levels <= PC's level*: Will return one of any of the items listed as equal to or below player's level (levels are set in left column). If this is not checked, only the items equal to the player's level can be returned.

*Calculate for each item in count*: If checked, will choose a new item for each time the list is called in a container. Otherwise, all items in a multiple count container will be the same. Essentially, if you have 5 items from a list in a chest, and this isn't checked, the container will have 5 of the same item. If it is checked, the items may (or may not) be different - it's random.

*Chance None*: Percent chance that no item will appear.

*PC Level*: The level a leveled item will appear at (defaults to 1; if multiple levels are listed, any new items added to list will have highest level)

*Item Nam*e: Item ID (or leveled list ID, since those can be called by other lists)

*Blocked:* If checked, the specific object becomes read-only.

## Enchanting

*ID*: The object's unique ID. (character limit: 32)

*Cast Type*: Cast When Used, Cast When Strikes, Cast Once, Constant Effect. Cast Once enchantments can only be placed on scroll book objects. Cast When Strikes can only be placed on weapons.

*Effects*: Select effects associated with enchanting.

*Cost*: Number of enchantment points required to cast enchanting.

*Range*: How far away the spell is effective.

*Area*: The area affected by the enchanting.

*Duration*: How long the effect lasts.

*Magnitude* (min/max): The strength of the effect.

*Auto Calculate:* Automatically calculates the Enchantment cost based on the effect costs.

*Blocked:* If checked, the specific object becomes read-only.

## Ownership

This data comes in one of two forms. Either *Faction and Rank*, or *NPC and Flag.* An object cannot have both.

Some Examples:
The door to the back room of a temple has an ownership value of Temple, 6. Thus if the player activates this object and is not a member of the Temple and at least Rank 6 (Disciple), he is committing a crime.

The door to the bedroom of Ken's house the has ownership of Ken, and the flag KenOK = true. This means that Ken owns it and unless KenOK = true, you are committing a crime if you activate that door. In Ken's house are several swords, each with ownership of Ken and no flags. This means there is no circumstance that will allow you to legally take the swords. Taking them is always a crime.

In addition, when bartering with an NPC in the game, the NPC will offer any objects he *owns*. So for a shopkeeper, it is important to mark all the objects you want him to sell.

He also owns any object placed in a container he owns.

## Character

*ID:* The object's unique ID. Most NPCs Name and ID are the same for ease of use. Remember, this is the ID by which the NPC is referred to in all code and a script and is never seen by the player. (character limit: 24)

*Name:* The NPC's name. This is what appears in the game to player. (character limit: 32)

*Script:* The script associated with the NPC.

*Race:* Pick from available races.

*Female*: If checked, NPC is female.

*Class:* The character's class chosen from the editor's list (Warrior, Spellsword, Mage, Bard, Agent, etc). Skills are predefined for the character based on Class and level. Custom classes that the creator wants to use must first be defined under the Character\Class option.

*Level:* The level of the NPC. Used to automatically adjust attributes, skills, and health.

*Faction:* The group the NPC belongs to.

*Rank:* Rank in the selected faction (cannot have Rank without Faction). List of Ranks taken from Faction's list.

*Essential:* This NPC has inventory or information essential to the game. If killed, this NPC's body will remain, and will not be cleared when the cell is loaded. The player will also get a message telling them an important NPC has died.

*Respawn:* NPC will reappear after certain time period.

*Corpses Persist:* After dead, their corpse remains. If the NPC's health is set to zero, they will be dead when they get loaded (used for placing corpses).

*Attributes*: Attribute values 1 to 100. Modified based on race, class, and level.

*Health:* Defaults to (Str + End/2) + ( Level * (End/10)). Can be manually edited by the user.

*Magicka:* Defaults to (Int x class and race modifier). Can be manually edited by user.

*Fatigue:* Defaults to (Str+End+Agi+Wil). Can be manually edited.

*Disposition:* Starting disposition for the NPC. This is the unmodified base value, and will change in the game based on several factors.

*Reputation:* Fame level of the NPC.  NPC's who are more famous are more difficult to persuade.

*Blood Texture:* The blood texture used for the splat when this NPC is hit.

*Auto Calculate Stats:* Creates stats and spells automatically for the NPC based on his Race, Class, and Level.

*Skills:* Value of each skill defined in its own field. Defaults defined by class and level.

*Dialogue:* Opens the dialogue window filtered for this NPC.

*Animation:* Opens animation Window for this NPC's animation file.

*AI:* Opens AI window for this NPC.

*Head & Hair:* Simply chose the available head and hair objects for this NPC. Only heads and hairs that have been defined under body parts for the NPC's race and sex will be available.

*Inventory / Spell List:* A list of any objects the NPC is carrying along with any spells he has available. Items of a negative value are infinite. Spells may include diseases, abilities, etc.

*Blocked:* If checked, the specific object becomes read-only.

## Auto Calculate Stats

Creates stats and spells automatically for the NPC based on his Race, Class, and Level.

Spells are placed in through the following rules:

1. Only checks Auto-Calced Spells.
2. NPC must have a 50% (fAutoSpellChance) chance of casting the spell.
3. NPC must have magicka to cast the spell at least 1 (iAutoSpellTimesCanCast) time(s).
4. To get any spell that affects an attribute or skill (such as absorb strength) the NPC must have at least a 70 (iAutoSpellAttMin) in that attribute/skill.
5. NPC will get a maximum of 3 (iAutoSpellSCHOOLMax) spells from each school (ie: Altereation, Conjuration, etc).

## Disposition

Disposition is how an NPC feels about you on a scale from 0 to 100.

Things that effect disposition are:

| PC Action | Default Value | Game Setting Formula |
|---|---|---|
| **Same Race** | 5 | iDispRaceMod |
| **Personality** | 0.5 * (AttPer - 50) | fDispPerMult * (AttPer - iDispPerBase) |
| **Faction Reaction** | (FactionReaction * 5) * (0.5 + (Rank * 0.5)) | ( Faction Reaction * fDispFactionMod ) * (fDispFactionRankBase + (Rank * fDispFactionRankMult)) |
| **Criminal Status** | | Crime Level * fCrimeDispMod |
| **Diseased** | -10 | fDispDiseasedMod |
| **Attack NPC** | -20 | iDispAttackMod |
| **Bargain** | 1 | iDispBargainSuccMod |
| **Failed Offer** | -1 | iDispBargainFailMod |
| **Stealing** | -1 * Item Value | fDispStealing * Item Value |
| **Pick Pocketing** | -25 | iDispPickPocketMod |
| **Trespassing** | -20 | iDispTresspass |
| **Taunting** | From Persuasion Formula | |
| **Intimidation** | From Persuasion Formula | |
| **Bribery** | From Persuasion Formula | |

## AI

Most AI in Morrowind is handled within the actor (character or creature) themselves. They will decide how to behave based on their current stats.

These stats include their attributes, skills, health, disposition, etc. It also includes 4 stats that are set through the AI window. These are

Fight: How much the actor wants to fight the PC.

Flee: How likely the actor is to flee when in combat.

Alarm: How likely the NPC is to react to the PC committing a crime. NPCs only.

Hello: The distance within which the actor will say “hello” to the PC (this is multiplied by the game setting iGreetDistanceMultiplier, which defaults to 7). NPCs only.

**AI Packages**

Each Actor may have a series of AI Packages. These are instructions they execute in order. When they are done with the last one in the list, they start the list over. There are 5 different packages you can assign to an actor:

Wander: Makes the actor wander around a set area.
Travel: Makes the actor travel to a specified location.
Follow: Makes the actor follow a specified target.
Escort: Makes the actor escort another object to a specific location.
Activate: Makes the actor “use” an object in the world, such as a door or lever.

When an actor finishes one package, they will start the next one in the list. If the actor enters combat (fighting and fleeing), the current package is suspended until the fight is over or the actor has sufficiently fled.

**Services**

The Services tab is where you select which services the actor can offer the PC. Simply check off the ones you wish him to have. The Barter Gold value is how much money the actor has to spend in barter. The amount the actor is left with after a transaction will reset to this value after a time. For Travel Services, locations he travels to are set up the same way Teleport locations are set up for Door references. If an NPC is marked as Auto-Calc, he can not select his own services but instead uses the Auto-Calc services selected for his class.

## Wander

Wander is the most popular form of AI, causing an actor to move around at random. Actors will intelligently choose points to walk to based in the path grid of the cell they are in.

**Idle Chances**
This is the percentage chance the actor will stand still and perform the particular idle. Each idle is checked, and the one that passes with the highest roll is played. If no Idle passes the random roll, the Actor will move (walk).

The Idles are:

- Idle2: Looking around
- Idle3: Looking behind
- Idle4: Scratching head
- Idle5: Shifting clothing or armor on shoulder
- Idle6: Rubbing hands together and showing wares
- Idle7: Looking at fingers and looking around furtively
- Idle8: Deep thought
- Idle9: Reaching for weapon

## Travel

Makes actor walk to a certain XYZ location. You can type in the location or “View the Location”.

*View Location:* A red X will appear in the render window that you can move around with the standard render window object controls. Place the X on the travel destination.

This package ends when the actor reaches that location.

## Follow

Makes the actor follow another actor to a location or for a specified period of time. During this time the actor will also protect the actor it is following.

*Target:* The ActorID to follow. Remember that since all ActorIDs share the same AI packages, putting this on an Actor with multiple references will cause ALL references of that actor to follow the same actor. Thus, this type of AI should only be placed on specific or unique sets of Actors.

*Duration:* The duration the actor should follow for. Trumped by providing a location.

*Follow to:* Check this to use location data for following.

*Cell:* The Cell to follow to.

*XYZ:* Like Travel, specify the XYZ location to follow to.

*View Location:* A red X will appear in the render window that you can move around with the standard render window object controls. Place the X on the follow destination.

## Escort

Makes the actor escort another actor to a location or for a specified period of time. During this time the actor will also protect the actor it is escorting.

If you are not doing this package with the player as the target, you'll want to also put a follow package on the target Actor, since escorting an actor makes the escorter wait for the other actor. If the Target does not know they are supposed to follow, the escorter will most likely just stand there.

*Target:* The ActorID to Escort. Remember that since all ActorIDs share the same AI packages, putting this on an Actor with multiple references will cause ALL references of that actor to attempt to escort the same actor. Thus, this type of AI should only be placed on specific or unique sets of Actors.

*Duration:* The duration the actor should escort for. Trumped by providing a location.

*Escort to:* Check this to use location data for the escort.

*Cell:* The Cell to escort to.

*XYZ:* Like Travel, specify the XYZ location to escort to.

*View Location:* A red X will appear in the render window that you can move around with the standard render window object controls. Place the X on the escort destination.

## Activate

This package tells the actor to activate the specified ObjectID. A powerful and admittedly underutilized and undertested package.

| Object Type | Activation |
|---|---|
| NPC | Dialogue |
| Container | Opens |
| Door | Opens |
| Weapon, armor, misc, etc | Picks Up |
| Book/Scroll | Reads |

## Fight

An actors fight setting determines how prone the actor is to attacking the PC. When an actor's fight setting hits 100, they will attack the PC.

Player actions will increase (or decrease) an actor's fight setting. These are:

| PC Action | Default Value | Game Setting Formula |
|---|---|---|
| **PC Distance** | 20 - (Char Distance * 0.005) | iFightDistanceBase - (Char Distance * fFightDistMult) |
| **Attack Actor** | 100 | iFightAttack |
| **Disposition** | ( 50 - Disposition )*1 | (50 - Disposition) * fFightDispMult |
| **Stealing** | 5 * Item Value | fAlarmStealing * Item Value |
| **Pick Pocketing** | 25 | iAlarmPickPocket |
| **Trespassing** | 25 | iAlarmTresspass |
| **Taunting** | From Persuasion Formula | |
| **Intimidation** | From Persuasion Formula | |
| **Bribery** | From Persuasion Formula | |

The following table gives you the following general behavior:

| | |
|---|---|
| **100** | Always Attacks |
| **95** | Will Attack as PC gets close (3000 units) |
| **90** | Will Attack as PC gets close (2000 units) |
| **80** | Will Attack as PC gets close or if he dislikes you (1000 units, 40 Disp) |
| **70** | Will Attack if close and strong dislike (1000 units, 35 disp) |
| **60** | Will Attack if he dislikes you and you get close (Disp below 30) |
| **50** | Will Attack if he hates you (Disp at 0) |
| **40** | Will attack if he dislikes you, and you get close. (500 Units, Disp 10) |
| **30** | Will Attack if hates you and you commit crime. |
| **20** | Will Attack if dislikes you and multiple crimes. |
| **10** | Will attack if he hates you and you do multiple crimes on him. |
| **0** | Will ONLY attack if attacked first. |

## Flee

This is used in determining how apt the actor is to flee. It is used in a much larger AI strategy that will not be documented in this help file.

Setting it to 100 will make the actor more likely to flee, but this may not always be the result, as the Actor will also use other factors like how much damage they can give out, or other strategies they may use such as magic and ranged combat.

## Alarm & Crimes

When a crime is committed, and it is detected by an NPC, they will shout something at the player, this also notifies other NPCs in the area.

When the NPCs hear this, they adjust their settings based on their alarm setting. The higher the alarm setting, the angrier they will get.

If an NPC has an alarm of 100,  he will put gold on the PC's head if they hear of a crime.

| PC Crime | Gold | Game Setting Formula |
|---|---|---|
| **Attack NPC** | 40 | IcrimeAttack |
| **Killing** | 1000 | IcrimeKilling |
| **Stealing** | 1 | fCrimeStealing * Item Value |
| **Pick Pocketing** | 25 | IcrimePickPocket |
| **Trespassing** | 5 | IcrimeTresspass |
| **Taunting** | 5 | IcrimeTaunt |
| **Intimidation** | 5 | IcrimeIntimidate |

If the NPC with alarm 100 is also of class "Guard", they will have extra behavior:

- Intercept the PC, by running up and arresting the PC.
- If the PC's CrimeLevel is over 10000, they will attack on site, instead of initiating dialogue.
- Guards will also attack any creatures they can see that are attacking people (including the PC).

## Hello

Hello equates to the distance at which the actor will stop, face the PC and say hello.

The setting (which defaults to 30) is multiplied by the game setting, iGreetDistanceMultiplier, which defaults to 7.

Thus, a setting of 30 yields a hello distance of 210 (just under 10 feet).

## Infinite Inventory Items

Any container, NPC, or creature can have items in their possession that are marked as infinite. These are marked by placing a negative amount in the container (such as −15). These items are always there, even if the NPC gives or sells them to the player.

The only time they truly go away is when the player interacts with their container outside of bartering. This includes picking the NPC's pocket, taking items off a corpse, or stealing from a chest.

Example: A priest has 3 healing potions, which are marked as infinite. Everytime you talk to him you can buy these three potions. He then has 0. The next time you talk to him, he has 3 again. If you buy two, then kill him. He has 3 potions on him. You can take these 3 potions off his body, but now they will not regenerate on his body.

NPCs that do not have infinite gold will only be able to spend what they have over the course of the game.

When you give or take money from the NPC through bartering, this amount is added (or subtracted) to his infinite amount.

Using an NPC's container through any means other than bartering (such as pick-pocketing, looting dead body), changes the infinite amount. So if I priest has 5 infinite potions, and you pick-pocket 2 of them, he now has 3 infinite potions.

## Creature

Creatures differ from characters (NPCs) in many respects. They are usually beasts, or things you fight, which do not adhere to the class structure and animation rules of NPCs (though they can be made to use the same object system for skeletons or goblins). They do not necessarily carry weapons and often have strange and variable attacks. Creatures do not perform the same actions as characters. They do not sneak, cannot have their pockets picked, etc. But they can be knocked out, and they do have attributes so that they work through the main systems of the game.

*ID:* Like objects, all creatures get a unique ID. The unique ID is used in game scripts and code. The player never sees this value. (character limit: 24)

*Name:* The creature's name. This is what appears in the game to player. (character limit: 32)

*Script*: The script assigned to creature.

*Type:* Creature, Daedra, Undead, or Humanoid. This categorizes the creature so certain spell effects will work on it. Certain actions *only* effect the Undead or Daedra for instance.

*Level:* The level of the Creature. This is not used to adjust attributes, but used to approximate the creature's power level to the player's. Level is also used in some of the game formulas.

*Essential:* This creature has inventory or information essential to the game. If killed, this creature's body will remain, and will not be cleared when the cell is loaded. The player will also get a message telling them an important creature has died.

*Corpses Persist:* The creatures corpse cannot be disposed of by the player and will remain forever. If the creature's health is set to zero, they will be dead when they get loaded (used for placing corpses).

*Respawn:* Creature will re-appear after certain time period.

*Attributes*: Attribute values 1 to 100.

*Health:* Defaults to (Str + End/2) + ( Level * (End/10)). Can be manually edited by the user.

*Spell Pts:* Defaults to (Int x advantage modifier). Can be manually edited by user.

*Fatigue:* Defaults to (Str+End+Agi+Wil). Can be manually edited.

*Soul:* The charge value of the soul if bound into a magical item.

*Combat, Magic, Stealth:* Creatures only get 3 skills; *Combat, Magic, Stealth*. These are used in place of all the skills used through the main resolution systems the game uses for NPCs. They can each be manually edited.

*Attack:* The min and max damage for each of the creatures attacks.

*Blood Texture:* The texture used for when the creature is hit.

*Dialogue:* Opens the dialogue window filtered for this Creature. Creatures, by default do *not* have dialogue, and only receive it when Dialogue is specifically selected for them. All NPCs have dialogue, but creatures only have it if their specific ID is attached to a piece of data.

*Animation:* Opens animation Window for this creature

*AI:* Opens AI window for this creature.

*Weapon & Shield:* Allows the creature to use any shields or weapons they have. To use these they must either be using the Biped flag (built to a character's bone structure), or have animations for weapon attacks.

*Movement:* Flies, walks, Swims, Biped or None

*Biped:* A check-box that sets the creature to use the same bone structure as an NPC. This allows these creatures to use the animations in the NPC's Base Animation file.

*Inventory / Spell List:* A list of objects the creature is carrying along with spells it has available. Items of a negative value are infinite. Spells may include diseases, abilities, etc.

*Sound Gen Creature:* The base creature from which this creature derives its sounds.

*Blocked:* If checked, the specific object becomes read-only.

*Scale:* This scale is applied to all references to that creature in addition to the reference's individual scale.

**See also**

Leveled Creature

## Leveled Creature

A leveled creature is a marker that chooses a creature to represent it in the world based on the level of the PC. When the cell is loaded, the creature will check the PC's level and load the creature in its list that is closest to the PC's level without going over. The user will fill in the level here for when that creature appears. If multiple creatures are listed for the same level, one is chosen at random. If no creatures meet the PC's level, no creature is displayed.

*ID*: The object's unique ID. (character limit: 32)

*Calculate from all levels <= PC's level*: Will return creatures listed as equal to or below player's level (level #s are set in left column). If this is unchecked, only creature listed as equal to the PC's level can be returned.

*Chance None*: Percent chance that no creature will appear.

*Blocked:* If checked, the specific object becomes read-only.

*PC Level*: level creature will appear at (defaults to 1; if multiple levels are listed, any new items added to list will have highest level)

*Creature Name*: Creature ID (or Leveled Creature ID, since those can be called by other lists)

## Factions

Opens the faction window, where factions and their properties can be defined. The main things that are defined are what the requirements are for each rank in terms of skills and traits, and the disposition of this faction towards other factions. Everything else for the faction must be checked in the individual NPCs. This includes the actual tasks to be given a rank increase and the actual services.

The NPCs can reference the data seen here. For example, Misa Drora, a member of The New Temple, can check to see if the PC's skills meet the requirements for rank 3, and offer a quest if they do. If they do not meet the requirements, she can say so without giving specifics. She can mention the skills and such that are important, if she is scripted to do so.

*Name:* The name of the faction as it appears to the player. (character limit: 32)

*New:* Creates a new faction.

*Favored Skills:* The entire list of skills with favored ones highlighted (these are also moved to the top, so they can be seen easily).

*Rank Name:* List of rank names. If edited here, they trickle down to duplicate locations of that rank.

*Attribute:* There are two of these. For each rank, the attribute level must be reached.

*Primary Skill:* The value that at least one of the favored skills must reach for rank advancement.

*Favored Skill:* The value that all of the favored skills must reach for rank advancement.

*Hidden from PC:* The faction will not be displayed to the player during the game.

*Reaction List:* List of selected factions and the base disposition of this faction towards them.

## Race

Definition of races, benefits and leveling advantages. Redguard, Nord, Imperial, Breton, High Elf, Dark Elf, and Wood Elf. The creator is free to define their own races as well. If a new race is added, the creator must also attach base body parts for that race. The creator can also make non-playable races for advanced creatures such as skeletons and goblins.

*Race:* Select the race to edit.

*New:* Start a new race.

*Name:* The name of the race as it appears to the player. (character limit: 32)

*Skill Bonus:* Pull down list of 6 skills the creator can modify for the race. You can add or subtract values to skills. The sum is listed below the skills. For balance, each race's skill bonus should equal 45.

*Base Attributes:* The starting level for race's attributes, male and female. For balance, should equal 320.

*Specials:* Spells can be dragged and dropped into here to add spells, abilities, and powers to all members of this race.

*Description:* The written description of the Race. Appears in character creation and in the Stats window of the game.

*Height:* Multiplier on the rendered height of the race in the game (y matrix multiplier).

*Weight:* Multiplier on the rendered width of the race in the game (x,z matrix multiplier).

*Playable:* If checked, this race can be used by the player for their character.

*Beast Race*: This race will use the beast animations (includes double jointed legs and tail). Such as Khajiits and Argonians.

## Class

Definition of classes, their advantages, major skills, etc.

*Name:* The name of the class as it appears to the player. (character limit: 32)

*New:* Create a new class.

*Primary Attributes:* Select 2 of the seven attributes to be primary.

*Specialization:* Category class fits into (Combat, Magic, Stealth). Class gets bonuses to all the skills of that specialization (20% easier to increase them, and starting +5 bonus).

*Major Skills:* Choose 5 major skills. Cannot repeat any here or from Minor.

*Minor Skills:* Choose 5 minor skills. Cannot repeat any here or from Major.

*Playable:* If checked, allows players to create PCs of this class, otherwise it is just used for NPCs. A farmer is an example of a class that the player would not choose from.

*AutoCalc Buys/Sells/Other:* If an NPC is marked as Auto-Calc, they do not select their own services but instead use the services selected here for their class.

## Skills

Costing data for all the skills. Use Values, governing attributes, etc. The skill list is static for the game. Skills cannot be added since their uses are coded into the game.

*Skill:* The current skill being edited.

*Governing Attribute:* The attribute that governs this skill. Governing attributes get raised when level increases and act as a ceiling for training that skill.

*Specialization:* The overall class for this skill. Combat, Magic, Stealth

*Actions:* These cannot be edited (use values can). But a skill can have up to four hard-coded actions that the game uses the skill for. An example would be that Athletics is used for the actions of Single Jump and Second of Run.

*Use Value:* The multiplier on the single action for how many uses it is worth. Need x number of uses to reach x level.

*Description:* Text description of the skill. Displayed in selection and stats screen of the game.

## Birthsigns

**ID**:  Allows you to enter, rename, or delete a Birthsign

**Full Name**:  The Birthsign name, as it will appear in the menus.  Underneath is a button for assigning an accompanying picture file. (character limit: 32)

**Spell List**:  Spells can be dragged from the Spellmaking tab in the Object Window to this list.

**Description**:  The full description of the Birthsign can be entered here, with as much (or as little) detail as the user chooses.

## Scripting Overview

You've probably noticed that all objects have a field in them for a script. Scripts are small pieces of code that can be placed inside of objects that allows them to perform special actions.

Examples of scripts usage include making an object move (such as a door opening), an object that gives off sounds (such as a buzzing light), or making an object hurt the player when it is equipped.

The script called **main** starts itself in the beginning of the game. This is used to start and stop other global scripts.

**See Also**
Script Commands
Script Functions
Sample Script
Global Scripts Start Scripts

## Global Scripts

Global scripts are scripts that run always, and are not connected with any particular object. They are used to run complex quests or time things that may happen globally, but not necessarily associated with a single object.

When the game starts, the script “Main” is started as a global script.

Like other scripts, global scripts can refer to other objects and so forth, but cannot do things like movements on itself (since the global script is not associated with an object reference)

The following functions are used on global scripts:

ScriptRunning, ScriptName
StartScript, ScriptName
StopScript, ScriptName

## Start Scripts

You can add or remove from this list under **Gameplay\Edit Start Scripts**

This is the list of global scripts, along with “Main” that will automatically be run when a game is started or loaded.

Convenient for launching new quests and such.

**See Also**
Script Commands
Script Functions
Sample Script
Global Scripts

## Sample Script

```
Begin TestScript

; comments are placed in using a semicolon

short var1          ;comments can also be placed at end of lines

if (MenuMode == 0)

    if (GetPlayerDistance > 200 )

        PlaySound ( TestSound )

    endif

endif

End TestScript
```

## Animation

Certain art files may contain animation for use with characters, creatures, or other objects. Within a creature or character you can define the animation it uses in different circumstances.

The game uses Animation Groups. The game plays these groups when the correct situation arises.

Creatures must all have their animation defined uniquely. That is, all creatures animate differently and use different art than other creatures. Characters, on the other hand, use a base set of animations that they all share. Characters can be given unique animations, which override the base animation groups.

## Animation Groups

**Idle** – **Idle9**
**WalkForward**
**WalkBack**
**WalkLeft**
**WalkRight**
**SneakForward**
**SneakBack**
**SneakLeft**
**SneakRight**
**Jump**
**TurnLeft**
**TurnRight**
**RunForward**
**RunBack**
**RunLeft**
**RunRight**
**SwimForward**
**SwimLeft**
**SwimRight**
**SwimBack**
**Knockout**
**Knockdown**
**SwimKnockout**
**Swimknockdown**
**Death**

**Hand to Hand** (Equip, Slash, Chop, Thrust, Block, Unequip)
**ThrowWeapon** (Equip, Attack, Block, Unequip)
**Crossbow** (Equip, Attack, Unequip)
**BowandArrow** (Equip, Attack, Unequip)
**WeaponOneHand** (Equip, Slash, Chop, Thrust, Block, Unequip)
**WeaponTwoHandClose** (Equip, Slash, Chop, Thrust, Block, Unequip)
**WeapnTwoHandWide** (Equip, Slash, Chop, Thrust, Block, Unequip)
**Shield** (Equip, Block, Unequip)

**Spell Idle**
**Spell** – **Self** (Start, Release, End)
**Spell** – **Touch** (Start, Release, End)
**Spell** – **Target** (Start, Release, End)

**Attack 1** – used for creatures only
**Attack 2** – used for creatures only
**Attack 3** – used for creatures only

**Hit1-Hit5** (Start, End)

When not mentioned, all of these groups have:

- Start
- Begin Loop
- End Loop
- End

Whenever an attack like Equip, Slash, Chop, Thrust, and Unequip are used, these also have the following properties:

Equip Start
Equip Stop
Slash Start
Slash Min Attack
Slash Max Attack
Slash Min Hit
Slash Hit
Slash Small Follow Start
Slash Small Follow Stop
Slash Medium Follow Start
Slash Medium Follow Stop
Slash Large Follow Start
Slash Large Follow Stop
Chop Start
Chop Min Attack
Chop Max Attack
Chop Min Hit
Chop Hit
Chop Small Follow Start
Chop Small Follow Stop
Chop Medium Follow Start
Chop Medium Follow Stop
Chop Large Follow Start
Chop Large Follow Stop
Thrust Start
Thrust Min Attack
Thrust Max Attack
Thrust Min Hit
Thrust Hit
Thrust Small Follow Start
Thrust Small Follow Stop
Thrust Medium Follow Start
Thrust Medium Follow Stop
Thrust Large Follow Start
Thrust Large Follow Stop
Unequip Start
Unequip Stop

## Base Character Animation

Here are the file names that are included with a brief explanation of what they do:

**Male Base animations** – This file contains the animations for every **humanoid** NPC in the game, as well as the Player when viewed in 3rd person mode. If you select this file in the editor, you can see the name of each animation and the frame at which the animation occurs.

**Male Base animations (1st person)** – This file contains all the base animations for the Player Character as viewed in First Person. All male players will use this file.

**Female Base animations** – These will override any male animations if they are labeled in this file for humanoids only.

**Female Base animations (1st person)** – Any animations include here will override the male version for **humanoid** Players.

**Khajiit/Argonian Base animations** – This file contains all the animations for the **beast races** - male and female, NPC and Player alike.

**Khajiit/Argonian Base animations (1st person)** – Any animations included in this file will overwrite the 1st person animations from the male **and** female 1st person files.

**Argonian Swim Base animation** – This file will overwrite the beast race base animation for

**Argonian** NPC's and Player Characters only.

## Animation Sound

Sound ID's are tagged individually in all animation files.

These get tagged in every creature in the Sound menu, and have generic lookups for Characters.

| | |
|---|---|
| SoundGen: Left | (left foot hitting ground) |
| SoundGen: Right | (right foot hitting ground) |
| SoundGen: Moan | (sound for monster idling, lurking, getting hit) |
| SoundGen: Roar | (sound for monster attacking or idling) |
| SoundGen: Scream | (sound for monster getting hit, dying, attacking) |
| SoundGen: Land | (sound when creature or character body hits the ground, either death, knockout, or landing a jump) |

Combat sounds are also labeled in the Sound menu of the editor and are played programmatically when the player or NPC gets hit, strikes with a weapon, or blocks. There are no sound notes placed in the animation files for these cases.

## Magic

Magic in Morrowind is made of up magic effects. Magic effects are the basic building blocks of all the magic in the game. This includes spellmaking, alchemy, and enchanting. The settings for each magic effect are:

*Scale Speed:* Speed multiplier for how fast the effect travels.

*Scale Size:* Set how much the visual effect should scale based on its magnitude.

*Size Cap:* The largest multiplier the effect can be scaled by.

*School:* The school skill used for casting this effect.

*Base Cost*: The effect cost used in all magic formulas.

*Particle Texture*: The particle texture that you see in the game.

*Effect Icon*: The magic menu icon for the magic effect.

*Description*: Describes how to use the magic effect and how it works.

*Creation Modes*: Flag Spellmaking and/or Enchanting to make the magic effect available for making spells or enchanting items.

*Casting Effects*: Select Sound and Visual files for effects when casting.

*Bolt Effects*: Select Sound and Visual files for bolt effects.

*Hit Effects*: Select Sound and Visual files for hit effects.

*Area Effects*: Select Sound and Visual files for area effects.

*Scale*:

*Speed* - Speed multiplier for how fast the effect travels.
*Size* - Set how much the visual effect should scale based on its magnitude.
*Size Cap* - The largest multiplier the effect can be scaled by.

*Lighting Effect*:

*RGB values*: Choose the color the magic effect adds to the world.

*Fire*: Use this flag if the 3D object used for the light has particles attached to it. This tells the game to stop the particles when the fire time is used up.

*Flicker*: Makes the magic effect's lighting "flicker", or randomly change its brightness.

*Negative*: If checked, this makes the lighting "dark". The color you selected for the light will be removed from the world, not added. So a white negative light will act like a black light.

*Dynamic*: If checked, this tells the lighting to light up any dynamic or moving objects (such as characters and creatures). Dynamic lights slow the game down a little more.

## Dialogue Overview

Dialogue in Morrowind is one large database that every NPC and creature pulls from. The basic principle being that a character will say the highest priority dialogue they pass all conditions for. If they pass no conditions, they have nothing to say about that topic.

**Adding, Deleting, and Editing Dialogue IDs**
You can add a new ID by right-clicking in the list on the left side of the Dialogue window and selecting "New." You can delete an ID by right-clicking in this list and selecting "Delete" or by selecting an ID and pressing the Delete key. You can edit an ID by selecting it and then clicking on it again with the left mouse button. Note that when you edit an ID which already exists in a master file, the editor marks the old ID as deleted and creates a new ID which exists only in your plugin.

**Topic**
This tab on the left-side of the Dialogue window shows all the topics that people in the game will respond to. Every time anyone in the game speaks in dialogue, that text is searched for this list of topics. If the text matches and the speaker has something to say about the topic, it will show up as a hyperlink in the game that you can select. People only have something to say if they meet the conditions for one of the possible responses in a topic.

Hyperlinks in the editor are shown as @hyperlink#. You can see the hyperlinks in the editor by clicking on the “Update Hyperlinks” button.

Topic matching is not case-sensitive and will always match longer phrases before shorter ones and earlier phrases before later ones. If you had the topics "My cat is smart," "Smart as a cat," and "Cat," the phrase "My cat is smart as a cat" would match "@My cat is smart# as a @cat#." The phrase "My cat is as smart as a cat" would match "My @cat# is as @smart as a cat#."

Topic matching for Creatures is slightly different than for NPCs. Creatures only get the dialogue that is assigned to their ID. If you assign dialogue to everyone in the Cell "Balmora" and you put a Scamp in Balmora, he will not have any of this dialogue.

**Voice**
Voices have audio files attached to them which are played whenever a "voice" is called for. You can select which file is attached to each voice ID with the Sound Filename button.

Alarm voices are not used, you should not see any.

Attack voices are played while the NPC or Creature is attacking. Creatures generally have these tagged in their animations, so this is mainly for NPCs.

Flee voices are played when an NPC is fleeing.

Hello voices are played when an NPC or Creature greets the player. At what distance this occurs is set by the Hello AI Setting.

Hit voices are played when the NPC or Creature takes damage. Again, most Creatures have these embedded in their animations.

Idles play randomly. These are like "clutter" for voices. You control how often and under what conditions idles are played with the conditions here.

Intruder voices are played when you are caught activating an owned item such as a door or container.

Thief voices play when an NPC detects a crime such as stealing or assault. Note that this occurs when a crime is detected. Even an NPC with an Alarm AI Setting of 0 will say this voice if they detect the crime.

**Greeting**
Greetings are what people say when they are first activated. In general, they work exactly like topics. You cannot enter dialogue with someone who has no Greetings. It is best to make sure there is a "default" greeting with no conditions that everyone will say if they fail everything above it.

The system looks for an ID that meets all the conditions starting with Greeting 0 through Greeting 9. The Greetings are divided into these 10 categories only out of convenience. In general, Morrowind uses Greetings this way:

Greeting 0: Alarmed
Greeting 1: Quests
Greeting 2: Vampires, Nudity
Greeting 4: Crime and Disease
Greeting 5: Quests
Greeting 6: Factions
Greeting 7: Classes, Endgame, Slaves
Greeting 8: Clothing
Greeting 9: Locations

This is just a guide to finding a particular Greetings in Morrowind. You can use them however you want.

**Persuasion**
In the Persuasion category are responses to different types of persuasion as well as Info Refusals and Service Refusals.

**Persuasion Fail/Persuasion Success**
Admire Fail shows all the things people can say when you fail to admire them. Admire Success shows all the things people can say when you succeed in admiring them. The same is true for Bribe Fail, Bribe Success, Intimidate Fail, Intimidate Success, Taunt Fail, and Taunt Success.

**Info Refusal**
Info Refusals are what people will say when the speaker does not meet certain conditions. This occurs most often with disposition. If someone talks about "Morrowind" only with a disposition of 70, the topic may appear in their topic list, but if their disposition is only 40, they will respond with an Info Refusal instead.

**Service Refusal**
Service Refusals prevent certain people from offering services. When the player selects Training, Barter, or Travel, a search is made through this list. If any of these pass, that person will not offer services and will say a Service Refusal instead. Most service refusals are based on faction rank or disposition.

Note that Disposition is "backwards" in Service Refusals. Normally Disposition will pass if the person you're speaking to has a disposition higher than the number here in dialogue. However, Service Refusals will pass (and the person will refuse to give services) if the person's disposition is lower than the number here.

You cannot use Persuasion on Creatures, only NPCs.

**Journal**
Journals show the text that is added to the player's journal with the Journal script function. Each journal has an index, which is used with the Journal command. This tab just shows all the journals and their text and index. For more information about how to use journals, see the Journal command.

Starting with Tribunal, you can also name journal entries for tracking quests and flag them as completed. See Quest Title .

**Filter for**
Filter on the bottom left side of the dialogue window will show all the dialogue in a particular person. It will show all dialogue for that person's ID, race, class, gender, cell, etc... Some things such as local variable states and journal conditions are not checked in the editor, so the list you see with Filter is often larger than the list of topics someone will have in the game.

**Info / Response**
The top right side of the window shows the text and some of the conditions for each line of dialogue. These are usually called "infos" or "responses." Each Topic (or Voice or Greeting, etc) can have more than one response.

When any topic (or Greeting or Voice, etc) is selected, the system looks through all the responses and returns the first one that matches all conditions. The search is done top to bottom. If you have more than one response with the same conditions, only the top one will ever be said. If you want to move an info in the list, select it and use the **Left and Right Arrow keys.** This will change its priority, but will also modify any info it goes past.

The Info/Response window shows a few of the conditions which can be put on responses, such as Disposition, ID, Faction, Cell, and the six Function/Variables. This makes it easier to find certain bugs at a glance.

You can edit the text of each response in the large text window or, if the response is only a few characters, you can double-click in the Info/Response window and edit the text there. If you cut & paste text into the editor, you should use the large text window, since the info/response window has a limit to the number of characters it can display. The large text window will let you enter up to 512 characters per response.

Some game data can be displayed in dialogue. For a list of these, see Text Defines.

**Speaker Condition**
These fields set up conditions for each response. All the conditions must be "True" in order for anyone to say this response.

**ID**
This is used to put dialogue in a specific person or persons. While most NPCs are unique in Morrowind, this function will put the dialogue in every instance of an NPC. If you have three of "Bob Smith" in your world, all three will have dialogue assigned to the ID "Bob Smith."

Creatures only have dialogue assigned to their ID. Otherwise, this works the same as for NPCs. Dialogue assigned to "scamp" will appear in every instance of "scamp."

**Race**
This assigns dialogue to a particular race. If you add a new race in the editor, the new race will appear in the pull-down list. Only people of this race will have this dialogue. If you delete a race, any dialogue assigned to that race will be blank, which effectively assigns it to everyone, regardless of their race.

**Class**
This assigns dialogue to a particular class. Just like Race above, if you make a new class, it will appear here and you can select it. Just like races, if you delete a class, this will be blank.

**Faction**
This assigns dialogue to a particular faction. Otherwise, it works just like Race and Class.

**Rank**
This assigns dialogue to everyone of a certain rank or higher. If the Faction is blank, it will assign it to everyone of that rank or higher, regardless of their faction. For instance, Faction == Redoran and Rank == 3 will assign dialogue to everyone in House Redoran of Rank 3 and above. Faction == Blank and Rank == 7 will assign dialogue to everyone in the game with rank 7 or higher who belongs to any faction.

**Cell**
This assigns dialogue to everyone in a particular area. Note that we only match the first characters. Dialogue in Cell Balmora will show up in the cell "Balmora" as well as "Balmora, Lucky Lockup" and "Balmora, Dorisa Darvel: Bookseller." Note also that the test is made on which cell the player is currently in. If an NPC is following you and you go to a new cell, the NPC's dialogue can change as a result.

**PCFaction**
This will only give dialogue if the player is in a particular faction.

**PCRank**
This will only give dialogue if the player is of a certain rank. If Faction is left empty, the player can be of this rank in any faction.

**Sex**
This lets you assign dialogue to one gender or the other.

**Disp**
This is short for "Disposition." Any dialogue with a disposition value will only be said by people whose disposition is higher than this number (for the exception, see Service Refusal above).

**Function/Variable**
These six conditions are where you have the most control over who says what when. See Dialogue Functions and Variables.

**Results**
In the results window, you can put in one-line script commands. You cannot use any commands that require more than one line, such as "if" statements. The results field is compiled and processed after the response is displayed, not before. See Scripting Overview.

**Update Shared By**
This button will show you everyone in the world who will have this dialogue with your current conditions. If you make changes or switch to another response, you will have to press this button again to get a new list. Changes you've made to a response may not be reflected in this window until you select another response which forces a "save" of the response data.

**Journal Preview**
This window lets you quickly see a journal without having to switch to the Journal tab and lose your place in the dialogue. Note that you can also have more than one dialogue window open at the same time.

**Update Hyperlinks**
This button will update all the hyperlinks in all the text in the editor. With a large topic list, this can take several minutes. Note that when you save your plugin, the hyperlinks are removed.

**Error Check Results**
This button will compile everything in the results field for all responses. It will display any warnings it finds while doing this (these are also saved to a file named warnings.txt in the Morrowind directory). It is a good idea to run this at least once before you release a plugin.

Warning: It is best to load your plugin, run the error check, and then close the editor and load your plugin again to fix any errors. Error Check Results can sometimes mark the responses it is checking and global variables as "changed." These then get saved into your plugin, which makes it larger than necessary and can interfere with other people's work.

**Sound Filename**
This button opens a directory view window where you can select which audio files are played in Voice. To add or change the sound file, just click on this button and open whatever file you want to use.

**OK**
This button just closes the dialogue window. Note that any changes to dialogue are "saved" when the changes are made. There is no way to "cancel" changes made in the dialogue window.

**%Defines**
You can use various variables and defines in text, such as %Name, which is the name of the speaker. See full list.

## Quest Title

With the Tribunal expansion, the journal can filter to display either all quests you have begun, or only those quests you haven't completed.

In the editor, quest journals are entered in the Dialogue interface on the tab marked 'Journal'.

*Quest Titles:* Each quest can be given a title. Quest titles must have an index number of '0'. The box for 'Quest Title' must be checked.

*Quest Finished:* Any journal entry with the 'Finished' box checked appears in the Journal as completed, and is hidden when the journal is filtered to show active quests. For any given quest, there may be more than one journal that ends the quest.

*Quest Restarting:* In some cases, you may design a quest that finishes, then restarts. For example, you may want the player to THINK the quest is done, then surprise him later by adding a journal that tells him he is not done after all.

EXAMPLE
Here's an example of journals for a simple quest.

We'll call the Quest 'Bad Zombie'.

First the quest needs an ID. Right-click in the column under the tab 'Journal' and select 'New'.

Enter the ID MS_BadZombie in the empty field. [We preface our miscellaneous quests with the prefix MS_ for easy reference; you can do anything you want.]

Now right-click in the Info/Response area and select 'New' to create a new Journal info.

This journal info will have the index '0' [see the entry location for 'Index' in the 'Speaker Condition' area], and will be the quest title. Enter the Quest Title 'Bad Zombie' in the empty field.

Our quest has only three significant states:
1. The questgiver told us to go kill the bad zombie.
2. We killed the bad zombie.
3. We reported back to the questgiver and he gave us some government cheese.

We will have a journal entry for each of these states.

The first journal is:
The False Todd said there was a bad zombie under his bed, and that I should go kill it for him, because he is scared.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the first journal, and give that journal an Index of '1'.

The second journal is:
I found a bad zombie under the False Todd's bed, and killed it. Now I have to go back and tell the False Todd. I bet he'll give me something really swell.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the second journal, and give that journal an Index of '50'.

The third journal is:

I told the False Todd I killed the bad zombie under his bed. And what reward does he give me? Government cheese! What a cheapskate.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the third journal, and give that journal an Index of '100'. AND, because this is the last journal, put a check in the 'Finished' box.

Now, when you play the 'Bad Zombie' quest, as soon as you receive the quest from the False Todd, an entry will appear in your journal for the 'Bad Zombie' quest. This title will appear in your journal whether you are displaying All Quest, or just Uncompleted Quests, because it isn't finished yet. Duh.

However, when you finally get your government cheese, and the quest is over, when you display only uncompleted quests in the journal, the title 'Bad Zombie' will no longer appear. Because it's finished. Natch.

And, supposing you only THOUGHT the bad zombie was dead...

Now we need three more journal entries.

The fourth journal is:
I talked to False Todd, and what do you know! He said there was STILL a bad zombie under his bed, and that I should go kill it for him, because he is REALLY scared now.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the fourth journal, and give that journal an Index of '101'. AND, because we are restarting the quest, put a check in the 'Restart' box.

The fifth journal is:
Sure enough, I found there was still a bad zombie under the False Todd's bed. This time I killed it and waited around for five days to make sure it was really dead. Now I have to go back and tell the False Todd. This time he better give me something really swell.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the fifth journal, and give that journal an Index of '150'.

The sixth journal is:
I told the False Todd I REALLY killed the bad zombie under his bed. And this time he gave me a swell Pocket Flaming Warhammer, so I am very happy.

Right-click in the Info/Response area and select 'New' to create a new Journal info. Type in the sixth journal, and give that journal an Index of '200'. AND, because, REALLY, HONEST, this time it's the last journal, put a check in the 'Finished' box.

## Text Defines

The following can be used in text strings, such as dialogue and message boxes:

| | |
|---|---|
| %Name | The speaker's name. |
| %PCName | The player's name. |
| %Race | The speaker's race. |
| %PCRace | The player's race. |
| %Class | The speaker's class. |
| %PCClass | The player's class. |
| %Faction | The speaker's faction. If they have no faction, it will be blank. |
| %Rank | The speaker's rank. |
| %PCRank | The player's rank in the speaker's faction. |
| %NextPCRank | The player's next rank in the speaker's faction. |
| %Cell | The cell the player is currently in. |
| %Global | Any global variable value. Floats display as 1.1, such as %Gamehour. |

**Defines for controls**

%ActionForward
%ActionBack
%ActionSlideLeft
%ActionSlideRight
%ActionMenuMode
%ActionActivate
%ActionUse
%ActionReadyItem
%ActionReadyMagic
%ActionCrouch
%ActionRun
%ActionToggleRun
%ActionJump
%ActionJournal
%ActionRestMenu
%ActionNextWeapon
%ActionPrevWeapon
%ActionNextSpell
%ActionPrevSpell

## Dialogue Variables

**Function**
Lists all the special dialogue functions you can use. Very similar to script commands, but there are many unique ones. **Full List**.

**Global**
This lists all the global variables. You can test against the value of any global variable. Note that the results may not be what you expect if this global can be changed while you are in dialogue. To avoid this for scripted variables, do not change the value when MenuMode == 1.

**Local**
This lists all the local variables. You can test for any variable, but if the speaker does not have a script or their script does not contain the variable you specify, this will return 0.

**Journal**
This returns the highest index that has ever been set for a particular journal. You can make Journals and set the index numbers in the Journal tab of dialogue. You set the index (and add it to the player's in-game journal) with the Journal script command. See Script Functions.

**Item**
This returns the number of any item that the player has in his or her inventory. You can test for any item in the game that can be carried.

**Dead**
This returns the number of any NPC or Creature ID that is dead. For instance, you can test if the unique individual "Bob Smith" is dead or you can check to see if the player has killed more than 30 Scamps. Two things should be kept in mind. First, it is safest (with most functions) to test > 0 instead of = 1 since someone can make a plugin with two of "Bob Smith." Second, if you kill 10 generic "scamps" and also "Creeper" (who is a scamp, but with a different ID), testing "Dead Scamp" will return 10, not 11.

**Not ID**
This is true if the speaker is not this particular ID. For all of these except Not Local, it does not matter what you set it equal to. "Not ID Scamp = 0" is the same as "Not ID Scamp = 1."

**Not Faction**
This is true if the speaker is not in this faction.

**Not Class**
This is true if the speaker is not of this class.

**Not Race**
This is true if the speaker is not of this race.

**Not Cell**
This is true if the player is not in this cell. Note that this function takes the most time to calculate and can slow the dialogue responses down if you use several of them in the same topic.

**Not Local**
This is true if the speaker does not have this local variable. Unlike most "Not" functions, this one does care what you set the variable to. Both the dialogue and the variable itself should be set to 0. This can be confusing. Here is a table of how this works:

| Not Local (in dialogue) | Variable Exists (y/n) | Value (in the script) | Pass? (speaker will say this) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| = 0 | No | NA | Yes |
| = 0 | Yes | 0 | No |
| = 0 | Yes | 5 | Yes |
| = 1 | No | NA | Yes |
| = -3 | Yes | -3 | No |

## Dialogue Functions

**Alarm**
This returns the base value of the speaker's Alarm AI Setting.

**Alarmed**
This is 1 if the speaker is currently Alarmed (has detected a crime) and 0 otherwise.

**Attacked**
This is 1 if the speaker has ever been attacked, and 0 otherwise.

**Choice**
This works with a command in the results field called "Choice." When the function "choice" appears in the results field, the game searches the current topic again, this time setting the value of "choice" to whatever number the player selected. Just like a regular search, it stops at the first response for which all conditions are true (so it may not necessarily hit the response you expect).

Warning: Make sure any choices you make are "reachable" or you can send the game into an infinite loop. For instance, if you had "Choice One 1 Two 2" and no other responses in that topic, the game would default to displaying the choice again. And again. And again.

**Creature Target**
Returns true (1) if the speaker is targeting a creature.

**Detected**
This is 1 if the speaker detects the player and 0 otherwise.

**Function Faction Rank Difference**
This is the player's rank in the speaker's faction minus the speaker's rank. Note that the first rank in a faction is 0 and your "rank" is −1 if you do not belong to that faction. A return value of 0 is the same rank, 1 is PC is one rank higher, -2 is PC is two ranks lower.

**Fight**
This is the AI Fight Setting of the speaker. This is the base value, not the value after disposition, distance, crime level, and so on have been added.

**Flee**
This returns the base value of the speaker's Flee AI Setting.

**Friend Hit**
Detailed explanation.

**Health Percent**
This returns the percent health of the speaker.

**Hello**
This returns the base value of the speaker's Hello AI Setting.

**Level**
This is the current level of the speaker.

**PC Acrobatics**
This is the player's current skill level in Acrobatics. There are other functions for the player's skills and attributes. They all work the same way. All of them return the current value of the attribute or skill, which includes disease, magical enhancements, etc.

**Blight Disease**
This is 1 if the player has a blight disease and 0 otherwise.

**PC Clothing Modifier**
This is the total value of all the clothing and armor the player is wearing. The value of your equipment changes the disposition of people in the game.

**PC Common Disease**
This is 1 if the player has a common disease and 0 otherwise.

**PC Corprus**
This is 1 if the player has Corprus and 0 otherwise.

**PC Crime Level**
This is the amount of gold the player has on their head, the same as the "bounty" on the character sheet.

**PC Expelled**
This returns 1 if the player is expelled from the speaker's faction and 0 otherwise.

**PC Level**
This is the level of the player.

**PC Reputation**
This is the value of the player's total reputation. This is separate from individual faction reputations and affects how everyone in the world reacts to you.

**PC Sex**
This is 0 if the player is male and 1 if the player is female.

**PC Vampire**
This is 1 if the player is a vampire and 0 otherwise.

**Rank Requirement**
This checks to see if you "qualify" for the next rank in the speaker's faction.
This returns 0 if you do not have enough Faction Reputation and do not meet the skill requirements.
This returns 1 if you meet the skill requirements, but do not have the Faction Reputation.
This returns 2 if you have the Faction Reputation, but do not meet the skill requirements.
This returns 3 if you qualify.

**Reaction High**
This returns the highest faction reaction between the speaker's faction and all of the player's factions.

**Reaction Low**
This returns the lowest faction reaction between the speaker's faction and all of the player's factions.

**Reputation**
This returns the speaker's reputation.

**Same Faction**
This is 1 if the speaker and the player are in the same faction and 0 otherwise.

**Same Race**
This is 1 if the speaker and the player are of the same race and 0 otherwise.

**Same Sex**
This is 1 if the speaker and the player are of the same gender and 0 otherwise.

**Should Attack**
This is 1 if the speaker wants to start combat with you.

**Talked to PC**
This is 1 if the speaker has ever talked to the player and 0 otherwise. You can use this to have someone say something the first time you speak with them.

**Weather**
This returns the current weather so you can have people talk about the rain or the ash storms, etc. See ChangeWeather script command for enums on weather types.

## Friend Hit

Used in dialogue for when you attack a member of your group (like a follower)

The return values are:

```
0 = never been hit
1 = hit by pc 1st time
2 = hit by pc 2nd time
3 = hit by pc 3rd time
4 = hit by pc 4th time and the npc/creature is no in combat with the pc
```

## Game Settings

Game settings let you change several defined variables for use in the game. These range from movement speeds, numbers used in combat formulas, to the text displayed on menus.

There are several sections for settings:

| | |
|---|---|
| Gameplay | Numbers used in gameplay formulas. |
| Items | Text used for standard item names. |
| Magic | Text used for magic effect names. |
| Stats | Text used for stat names. |
| Menus | Text used for menu displays. |

## Gameplay Settings

| | |
|---|---|
| AthleticsRunBonus | The amount that the athletics skill effects running speed. |
| BaseRunMult | The base multiplier for speed when running. |
| EconomyScale | Value that all item values and service costs are multiplied by. |
| EncumbranceMoveEffect | The amount encumbrance effects the reduction on character speed. |
| HandFollowMult | Percentage rate that the 1st person hand moves with the camera on the x axis. |
| MinimumWalkSpeed | The base minimum units per second that a character or creature walks. |
| MaximumWalkSpeed | The base maximum units per second that a character or creature walks. |
| ScaleHandSeconds | The number of seconds the 1st person hand takes to return to normal position after an attack. |
| TimeScale | Value that every real time second is multiplied by in game time. |

## MessageBox

MessageBox, “Message”, [var1], [var2], [“button1”], [“button2”]

MessageBox, “This is a Message”
MessageBox, "Shall I start to spin? GameHour = %.2f", GameHour, "OK", "No Way"

Displays a message on the screen. There are two basic types of MessageBoxes.

1. One that displays just text. This is displayed at the bottom of the screen and will go away in a small amount of time.
2. One that displays buttons the player can choose. This box stops time and displays itself in the center of the screen until the player chooses an option.

Variable names can also be passed into the MessageBox command. These are displayed in the order they are used as parameters. The message must say *how* the variable is to be displayed.

| **Notation** | **Variable type** |
|---|---|
| f | Float |
| D | Short or Long |
| S | string |

Float variables must also specify how many decimal places they should show.

Either type of MessageBox can display variables.

See the sample script “MessageTest”.

## Random

Random, Value
Random, 100

Random returns a value from 0 to the Value – 1. So the line:

Set myVar to Random, 100

makes myVar set to a random number from 0 to 99.

## MenuMode

MenuMode

If (MenuMode == 0)

This function should be used in just about every script. It tells you if the player is in menu mode or not. It is a good idea to separate your script logic into processing of commands in and out of menumode. Certain actions like using items are done when in menu mode.

## UsedOnMe

UsedOnMe, ObjectID
if ( UsedOnMe, Misc_pot_redware_01 )

Returns true if the ObjectId has been used on the calling object. This is used for scripts that make objects do certain things of the player uses an object on it.

See “UseTest” script

## OnActivate

OnActivate
If ( OnActivate == 1)

This function returns TRUE if the calling object has been activated. If you use this function in a script, it *overrides* the objects default activation. The system assumes you will be filling in the code to make the object do what you want it to do.

To use the default object's activation you must call Activate on it.

Note that the activator type of object has NO default activation.

## Activate

Activate

```
If ( OnActivate == 1)
        Activate
endif
```

This function tells the object to do its default activation.

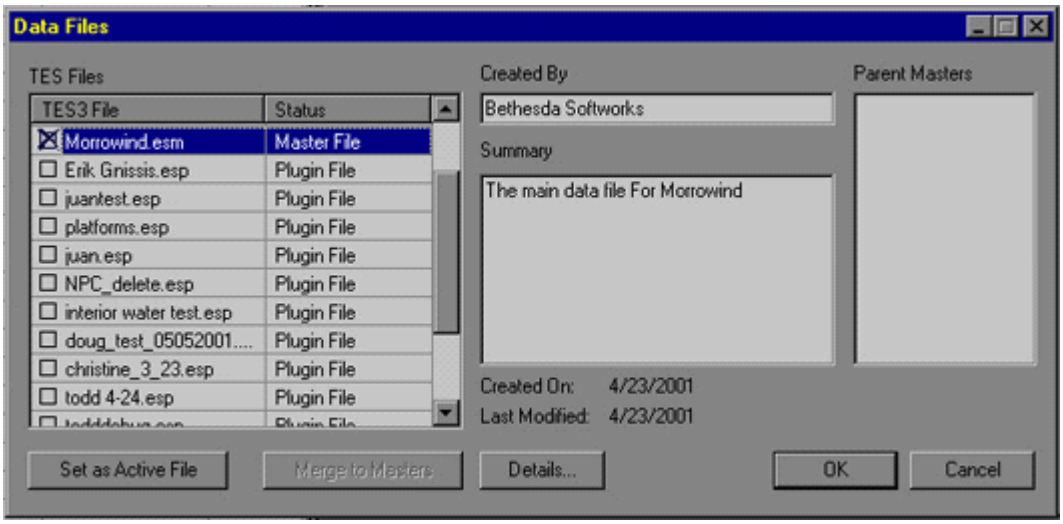
| Object Type | Activation |
|---|---|
| NPC | Dialogue |
| Container | Opens |
| Door | Opens |
| Weapon, armor, misc, etc | Picks Up |
| Book/Scroll | Reads |

## Dungeon Tutorial

Here is a brief look at how to create your own dungeon using the Construction Set.

### 1. Load up Morrowind Master file.

When starting any new plug-in, you should always load the “Morrowind.esm” first, since that is the file your plug-in will rely on for objects, creatures, landscape and any other data. All changes you do while working will be tracked as well as its relationship to the data in “Morrowind.esm”




### 2. Create a new interior cell.

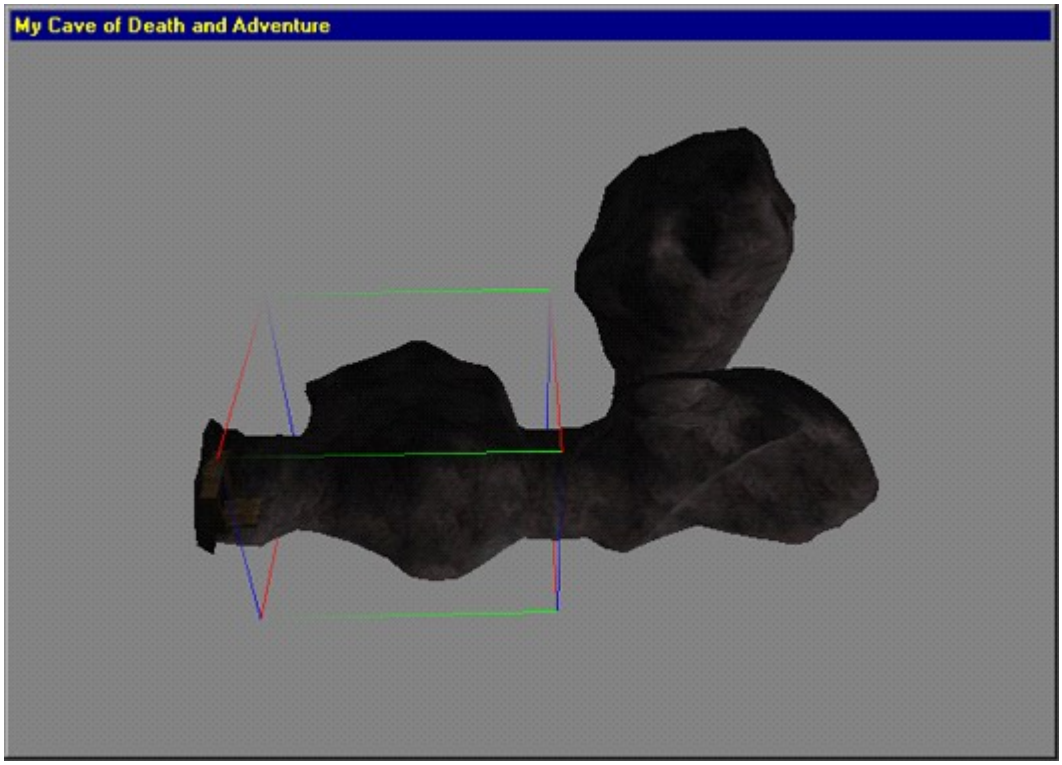
Once Morrowind.esm has loaded (this is a big file and may take up to a minute), create a new interior cell for your dungeon. All areas of the world are referred to as cells. Select **World\Interior Cell** and press **New**. Give your map a name (“My Cave of Death and Adventure”). Keep in mind that the player will see this name when entering the cell. You can also select other options here such as the ambient, fog, and sunlight color in your map, if it has water, and if the player can sleep in his area legally. Last, select your new cell from the cell view window by double clicking it. The render window will now switch to your cell, viewed from the top.


### 3. Layout the basic geometry.

The first thing to do in building your dungeon is to layout the pieces. Select the “Static” tab in the Object Window and select the objects you want to use. To place these, simply drag and drop pieces into the render window. Static pieces are used for basic building blocks such as walls, houses, furniture and other objects whose only purpose is to collide with.

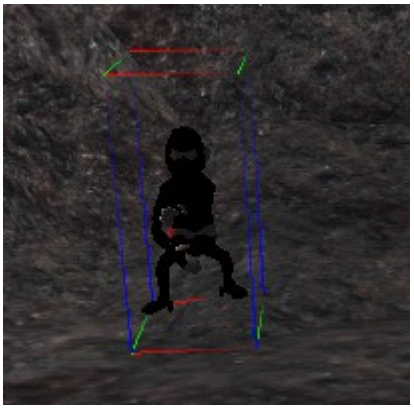
After you have the pieces in the cell, move them around and “snap” them together by dragging them around the render window. In this example we’ve used 4 of the pyrite cave pieces to snap together.

This is a small sized cave, but should provide a nice quick diversion for the player.




**4. Add creatures.**
Next select the “Creature” tab in the object window and drag in some creatures. We’re going to fill this dungeon with undead creatures, so we’ll use some skeletons and bonewalkers.

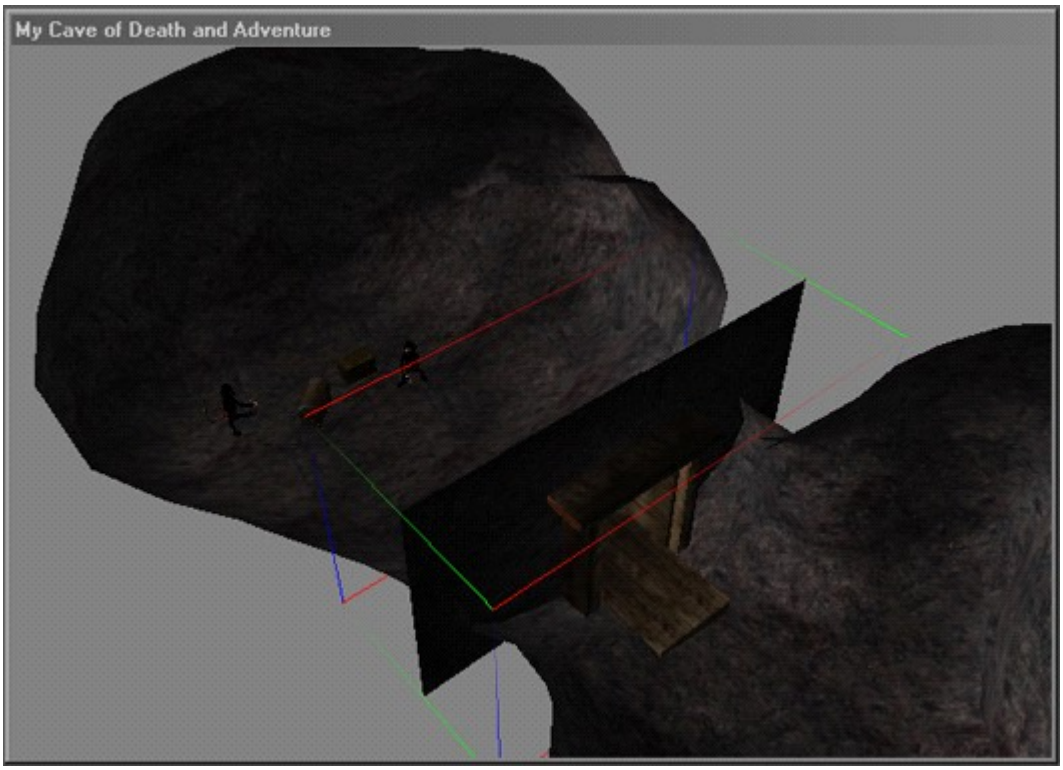

In the back room, we'll put some "Leveled Creatures". These are creatures that will be generated from a list that, like everything else in the editor, can be modified. In this example we'll fill the "leveled creature" list with undead creatures to keep the theme of the dungeon consistent, but any creature can be placed in the "Leveled Creature" list. Leveled Creatures are selected from the list based on the player's level, so our cave will be filled with harder creatures for higher level characters, and easier creatures for low level characters. These creatures will also respawn if the player kills them and comes back later on, so there will always be a few creatures in our dungeon for fun later in the game. Notice that the Leveled Creatures look like "Ninja Monkeys" in the Construction Set, but these will become real creatures when the player enters the area in the game. We've placed in two leveled undead creatures we created to keep things consistent.



**5. Add treasure, clutter, and doors.**

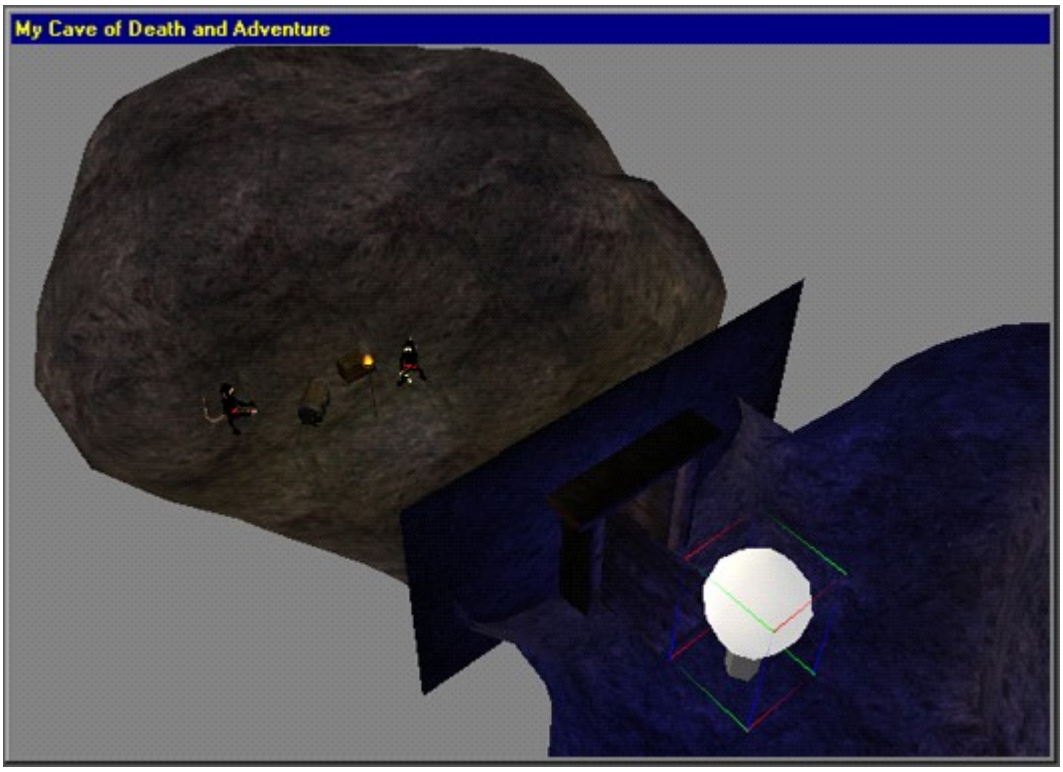
All dungeons need treasure and other stuff to make it interesting. So we'll drop in some chests from

the “Containers” tab that have some gold and other Leveled items. The Leveled Items work much the same way as leveled creatures. The designer can fill the Leveled Items list with objects that will be interesting and leveled to the player. You may want to drop in some furniture and such from the “Static” section to keep it interesting. We’ll also drag in a doorway (each architecture style has a doorway or door-jam piece) and a door into the hall to keep this back room separated. The doorway can be found under “Static” and the door under the “Door” tab.



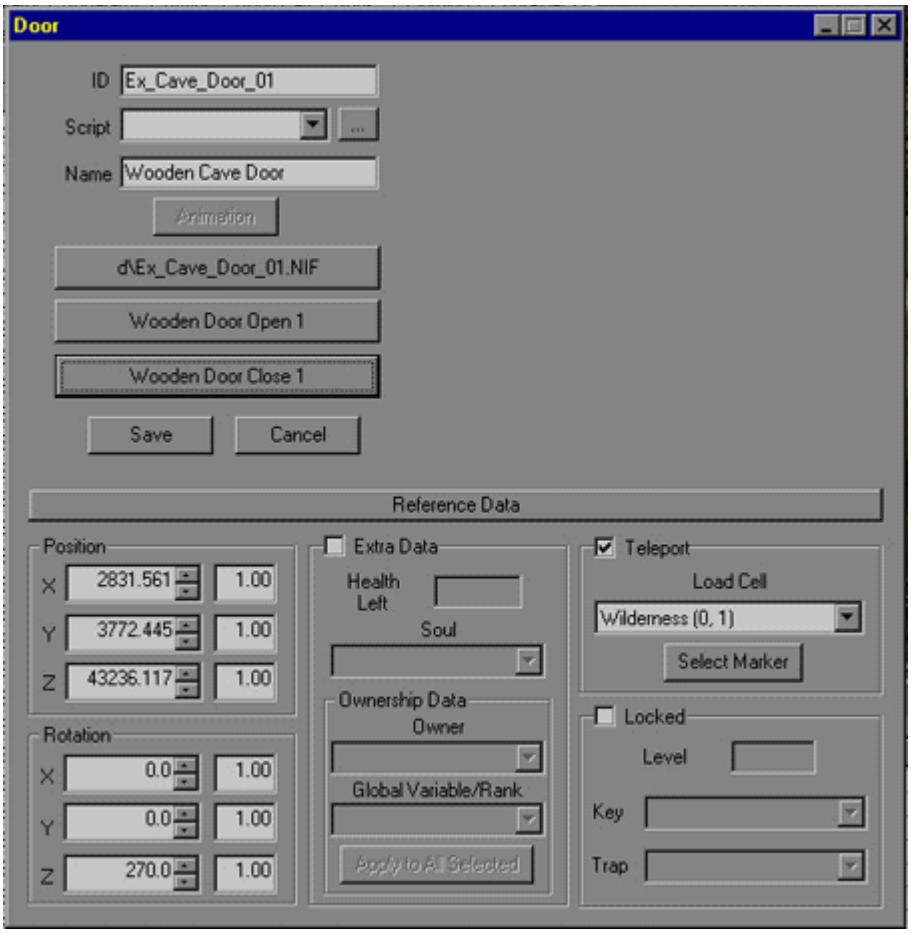

**6. Add Lights.**

Next you’ll want to add some lights, under the “Light” tab. There are two basic kinds of light, one is just raw light and has no associated art, such as the deep blue glow we’ve placed in the following shot. These lights will be represented as big light bulbs for placing purposes. The other kind is lights that have art, such as the torch we placed by the Leveled Creatures (ninja monkeys) in the back room.




**7. Link to the outside.**

The final step here is to link this dungeon to the outside world, or tell the dungeon exit where to go when the player uses it. Place a door at the entrance, double click it and set it to teleport the player to another location (choose an exterior cell, but you can also have that door lead to another cave or dungeon). You'll need to add a dungeon entrance to the outside world as well, so see the landscape sample for how to do that.




**8. Save and test.**

Last, press save and give your plug-in a name (MyPlug-in.esp). Everything you've changed has been tracked by the editor and will be saved into this file. Run Morrowind with your plug-in and test out your dungeon. Make sure all the doors work and it's fun. You'll probably want to tweak the creature placement and lighting after you play through it. They make a huge difference on how fun something is and how nice it looks. You may even want to add a few more rooms after you get the hang of things.

## The Elder Scrolls Construction Set

*The Elder Scrolls Construction Set* allows you to edit and create any data for use with *The Elder Scrolls 3: Morrowind*. Any data that goes into the game goes in through the Construction set and is stored in data files (labeled either ESM or ESP files) that are read by the game.

The Construction Set allows you to create your own areas (towns, dungeons, islands, etc) and populate them with characters, creatures, and stories.

You can add new races, classes, spells, potions, magic items, and anything else your mind can dream up.

You can also edit the data that comes with *Morrowind*. Don't like how fast everyone moves? Change the movement settings. Do you want Nords to start with better abilities? Want to rebalance spell costs? It's all here.

The most powerful feature of the Construction Set is the ability to create new data and stories, which are stored as plug-in files. You can place your plug-in on the Internet and have others play the quests and areas you have created. And since they are plug-ins, they add directly into someone's existing game without damaging the data already there.

Visit www.elderscrolls.com for info on how to download the best plug-ins out there.

**Some early sections you should read:**
Data Files
The Object Window
The Render Window
The Cell View Window

## The Object Window

The Object Window is the window from which the database of objects can be viewed and edited. It is also the window from which objects are dropped into the world (such as adding a hut to a village).

**Opening the Object Window**
The object window is opened by selecting View \ Object Window. It remains on screen until it is closed. It is a tabbed menu that allows you to view all object types, Characters, and creatures. Object types are tabs at the top. Clicking one opens the list of those objects in the database.

**Placing and Item in the World**
To place an object in the world, select it, and then drag and drop it into an open Render Window. You must drag from actual text, as trying to grab empty space will result in the object not being grabbed. The object will appear in the world at the point you let it go, and the Render Window will now be selected so you can place the object correctly.

**The Object Count**
Every object has a Count field. This is the number of references in the currently loaded game world to this object. Loading extra plug-ins will change this number if the object is also used in them. Keep in mind that including an object in a Leveled/Random list will only count once per list, not each time that list is referenced.

**Object Use Information**
To get information about how and where an object is used in the world, select it and press F1 to bring up the Use Report dialog. The bottom pane displays a list of the references to this object in the world. Double-click on an item in the list to go to that reference in the Render Window. The top pane displays a list of other objects that use this object in some way. This includes leveled lists which contain the item, Containers and actors who have the item in their inventory or spellbook, and even actors who have an AI Package related to this item. The Use Report dialog can also be opened by right-clicking on an item and selecting Info from the popup menu.

**Editing Object Data**
The object ID can be edited directly in here by clicking on the selected item (like renaming a Windows file). You can double-click the object to open a window showing all of its data. This is the most useful way of editing items such as scripts, art, etc. This window can also be opened by right-clicking on an item and selecting Edit from the popup menu.

**Sorting Object Data**
Each object type has certain data such as ID, type, name, health, etc. Clicking any of these field names will sort the list by that field. Clicking the field name again will sort by that field in the opposite direction (ascending or descending).

**Adding an Object**
To add an item to the database, simply select the category you want with the tabs, and press the INSERT key. This will open a window containing the object data. This window can also be opened by right-clicking in the Object Window and selecting New from the popup menu.

**Deleting Objects from the Database**
Pressing delete will remove the selected item. If this object has been used in the world the deletion is confirmed. This can also be accomplished by right-clicking and selecting Delete from the popup menu.

## The Render Window

The Render window is the area where the world can be viewed and manipulated. Objects can be moved, copied, deleted, edited. Much of the editing process involves dragging and dropping objects from the Object Window into the Render Window.

**Moving your view (camera)**
The camera can be moved in several ways:

- To pan the camera, hold the **SPACEBAR** while moving the mouse, or hold down the mouse-wheel.
- To Zoom the camera, hold down **V** while moving the mouse or spin the mouse-wheel.
- The **arrow keys** allow you to move quickly through the world by moving a half cell distance at a time.
- To rotate the camera, hold down **SHIFT** while moving the mouse. If an object is selected, the camera will rotate around the center of the screen, at the distance of the selected object.
- You can center a selected object in the window by pressing **C**. This is also a fast way of zooming in on an object.
- You can also center on a selected object in the window by pressing **T**. This switches the camera to a close-up, top-down view of the object.

**Selecting Objects**
Selecting objects can be done multiple ways:

- **Left click** to select an object. A rotated bounding box highlights it. Other selected objects are deselected.
- **Left clicking** empty space (or landscape) deselects all objects. Pressing **D** also deselects.
- Hold **Ctrl** to select/deselect multiple objects by clicking on them.
- Drag a selection box to select multiple objects.

**Moving Objects**
Only selected objects can be moved. You'll know if you are able to move an object if the + symbol appears along with your cursor. Hold down the **left mouse button** and drag the object to move it.

Objects move, by default on the horizontal (xy) plane. If you want to move an object vertically (z plane), hold down the **Z** key while moving the mouse. You can also lock the object's movement to the world's XY axis by holding down **X** while moving.

To rotate the object, hold down the **right mouse button** while moving the mouse. The default axis is Z, since most object rotations are done around this axis. If you want to rotate around the X axis, hold **X**, and if you want to rotate around Y, hold **Z** (yes, holding Z is bizarre for Y rotations, but it was better than having to hold it all the time since most rotations are Z rotations, so they were switched).

**Dropping Objects**
Press **F** to make objects fall. They will hit any object. You may have to press it multiple times if you have multiple objects selected that can fall onto each other. This is a quick way to get objects to "snap" to the floor, or a shelf, etc.

**Editing Object Data**
Double clicking an object opens its properties window. Changes made to an object here that are not explicit reference data, such as ownership, are inherited by the other objects of this ID in the world.

If you edit weap_longsword_07, and make it do more damage, you have changed every weap_longsword_07 in the world. See also Object Reference.

**Deleting Objects**
Pressing delete will delete the selected objects from the world.

**Copying, Cutting, Pasting, and Duplicating Objects**
You can use **Ctrl-C** to copy objects, **Ctrl-X** to cut them, and **Ctrl-V** to paste them. Pasted objects are placed in front of the camera. Multiple objects can be copied and pasted at once; they simply all need to be selected at the same time, and will retain their relative orientation when pasted.

**Ctrl-Shift-V,** will place the objects down, “in-place” or at the coordinates they previously existed, but in the current cell you are viewing.

You can also press **Ctrl-D** to duplicate selected objects. This places duplicates of the objects right on top of the selected ones. You can then move the new object(s) into place. This is great for duplicating hallway pieces for quick building.

## The Cell View Window

The Cell View Window allows you to view all of the cells in the world, both exterior and interior. It is the only way to switch your Render Window to interiors, and also serves as a quick way of jumping around the world.

The Cell View has two sections. The first lists all of the cells, their name and exterior grid number (or interior). The second section lists all of the objects within the selected cell.

Both sections can be sorted by their respective fields:  the cell list by Cell Name, Grid, Reference Count, and Path Grid, and the object list by Object ID, Type, and Ownership.

**Loading/Moving to a Cel**
Double click the cell name and the render window will switch to an overhead view of that cell. If the cell is not currently loaded into memory you will get a pause while the art loads.

**Switch View to an Object**
By double clicking an object from the list of objects in the cell, your render view will switch to this object, placing it in the center of the screen (like pressing **C** from the Render Window). The object will also be selected. This is one of the fastest ways of finding and editing objects.

**Renaming a Cell**
To rename a cell, select it, and then click its name. Remember that all interior cells must have unique names.

**See Also**
Cell Overview
Naming Cells

## The Preview Window

This window allows you to see a preview of the art for your selected object. You can also play the object's animation. When selected, the Render Window will actually change into the Preview Window. The object currently selected in the Object Window will then be displayed. Closing the Preview Window will restore the Render Window.

**Rotating the Object**
Use the arrow keys on your keyboard to rotate the object in the window.

**Viewing the Animation**
If the object has any animation attached to it, buttons will appear at the top of the window that allow you to play, rewind, and pause the animation. The current frame and number of frames in the animation will also be shown.

**Viewing a new Model**
You can right click the preview window to load/view a new model. This DOES NOT make the currently selected object use this model; it is only for viewing purposes.

**See Also**
Animation

## GetEffect

GetEffect, Effect
If ( GetEffect, sEffectShield == 1)

This function returns TRUE if the calling Actor is being affected by the effect.

**See Also**
Magic Effect List

## RemoveEffects

RemoveEffects, Effect
RemoveEffect sEffectShield

Removes all spells on the actor that include the Effect.

**See Also**
Magic Effect List

## Magic Effect List

When using script commands with magic effects, the following values should be used.

sEffectWaterBreathing
sEffectSwiftSwim
sEffectWaterWalking
sEffectShield
sEffectFireShield
sEffectLightningShield
sEffectFrostShield
sEffectBurden
sEffectFeather
sEffectJump
sEffectLevitate
sEffectSlowFall
sEffectLock
sEffectOpen
sEffectFireDamage
sEffectShockDamage
sEffectFrostDamage
sEffectDrainAttribute
sEffectDrainHealth
sEffectDrainSpellpoints
sEffectDrainFatigue
sEffectDrainSkill
sEffectDamageAttribute
sEffectDamageHealth
sEffectDamageMagicka
sEffectDamageFatigue
sEffectDamageSkill
sEffectPoison
sEffectWeaknessToFire
sEffectWeaknessToFrost
sEffectWeaknessToShock
sEffectWeaknessToMagicka
sEffectWeaknessToCommonDisease
sEffectWeaknessToBlightDisease
sEffectWeaknessToCorprusDisease
sEffectWeaknessToPoison
sEffectWeaknessToNormalWeapons
sEffectDisintegrateWeapon
sEffectDisintegrateArmor
sEffectInvisibility
sEffectChameleon
sEffectLight
sEffectSanctuary
sEffectNightEye
sEffectCharm
sEffectParalyze
sEffectSilence
sEffectBlind
sEffectSound
sEffectCalmHumanoid
sEffectCalmCreature
sEffectFrenzyHumanoid

sEffectFrenzyCreature
sEffectDemoralizeHumanoid
sEffectDemoralizeCreature
sEffectRallyHumanoid
sEffectRallyCreature
sEffectDispel
sEffectSoultrap
sEffectTelekinesis
sEffectMark
sEffectRecall
sEffectDivineIntervention
sEffectAlmsiviIntervention
sEffectDetectAnimal
sEffectDetectEnchantment
sEffectDetectKey
sEffectSpellAbsorption
sEffectReflect
sEffectCureCommonDisease
sEffectCureBlightDisease
sEffectCureCorprusDisease
sEffectCurePoison
sEffectCureParalyzation
sEffectRestoreAttribute
sEffectRestoreHealth
sEffectRestoreSpellPoints
sEffectRestoreFatigue
sEffectRestoreSkill
sEffectFortifyAttribute
sEffectFortifyHealth
sEffectFortifySpellpoints
sEffectFortifyFatigue
sEffectFortifySkill
sEffectFortifyMagickaMultiplier
sEffectAbsorbAttribute
sEffectAbsorbHealth
sEffectAbsorbSpellPoints
sEffectAbsorbFatigue
sEffectAbsorbSkill
sEffectResistFire
sEffectResistFrost
sEffectResistShock
sEffectResistMagicka
sEffectResistCommonDisease
sEffectResistBlightDisease
sEffectResistCorprusDisease
sEffectResistPoison
sEffectResistNormalWeapons
sEffectResistParalysis
sEffectRemoveCurse
sEffectTurnUndead
sEffectSummonScamp
sEffectSummonClannfear
sEffectSummonDaedroth
sEffectSummonDremora
sEffectSummonAncestralGhost
sEffectSummonSkeletalMinion

sEffectSummonLeastBonewalker
sEffectSummonGreaterBonewalker
sEffectSummonBonelord
sEffectSummonWingedTwilight
sEffectSummonHunger
sEffectSummonGoldensaint
sEffectSummonFlameAtronach
sEffectSummonFrostAtronach
sEffectSummonStormAtronach
sEffectFortifyAttackBonus
sEffectCommandCreatures
sEffectCommandHumanoids
sEffectBoundDagger
sEffectBoundLongsword
sEffectBoundMace
sEffectBoundBattleAxe
sEffectBoundSpear
sEffectBoundLongbow
sEffectExtraSpell
sEffectBoundCuirass
sEffectBoundHelm
sEffectBoundBoots
sEffectBoundShield
sEffectBoundGloves
sEffectCorpus
sEffectVampirism
sEffectSummonCenturionSphere
sEffectSunDamage
sEffectStuntedMagicka

## PCRace

If you want to know the race of the PC check the global flag PCRace.

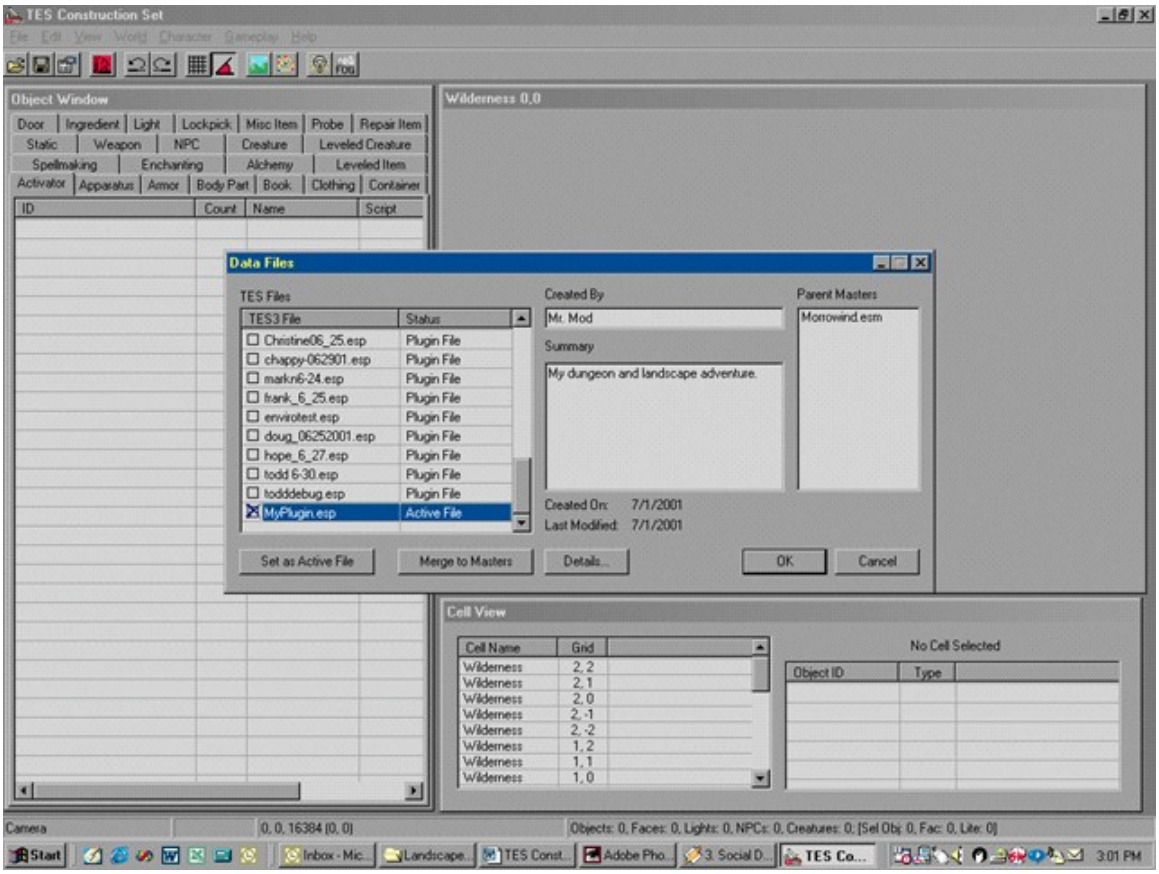
```
if ( Player->GetRace "Argonian" == 1)
      set PCRace to 1

elseif ( Player->GetRace "Breton" == 1)
      set PCRace to 2

elseif ( Player->GetRace "Dark Elf" == 1)
      set PCRace to 3

elseif ( Player->GetRace "High Elf" == 1)
      set PCRace to 4

elseif ( Player->GetRace "Imperial" == 1)
      set PCRace to 5

elseif ( Player->GetRace "Khajiit" == 1)
      set PCRace to 6

elseif ( Player->GetRace "Nord" == 1)
      set PCRace to 7

elseif ( Player->GetRace "Orc" == 1)
      set PCRace to 8

elseif ( Player->GetRace "Redguard" == 1)
      set PCRace to 9

elseif ( Player->GetRace "Wood Elf" == 1)
      set PCRace to 10
```

## Landscape Tutorial

You'll probably want to read the tutorial on World Building first. Landscaping is one of the key ways you can use in the *Construction Set* to give an area a unique look.

### 1. Load your plugin file.

You'll want to load your plugin first, "MyPlugin.esp" (see the World Building tutorial). When you load it, select it to be the "active" plugin. You can run the *Construction Set* with many plugins loaded, so you must set one to active. The active plugin is the one that all of your changes are being saved to.




### 2. Find an area to build on.

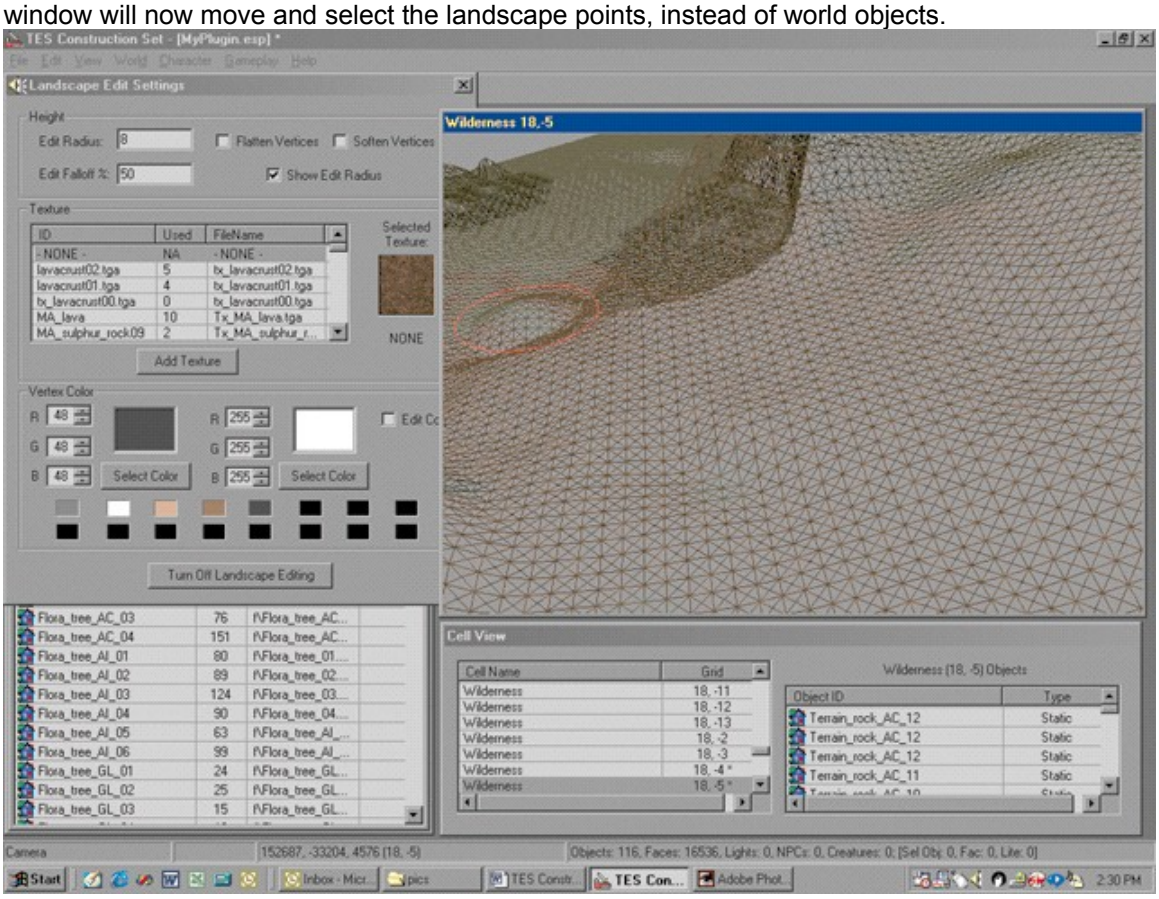
Landscape is infinite in *Morrowind.* It goes on forever. Most of it is underwater though. An infinite ocean surrounds the game area. You can add your own islands and continents there, or modify the landscape on the existing map. If you wish to edit the landscape that is already there, you'll probably want to find a place that isn't too close to any existing towns. For our demo, we'll start with an area off the existing continent that is just water.

### 3. Shape the landscape.

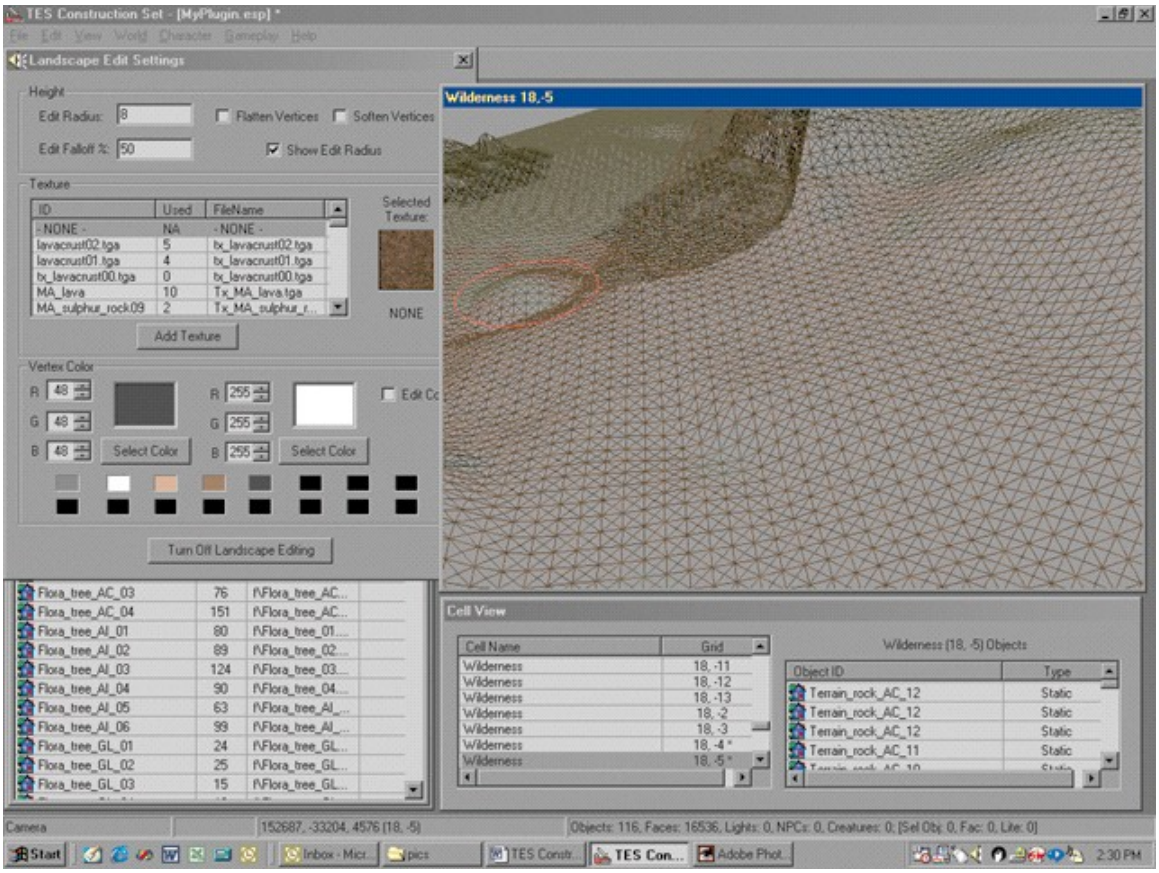
The landscape is a "height map" made up of vertices. You can pull these vertices up and down to shape the landscape. You can only move them up and down because this keeps the massive amount of landscape data to a reasonable level for storage.

Click the landscape button on the toolbar to enter landscape editing mode. Your clicks in the render

window will now move and select the landscape points, instead of world objects.

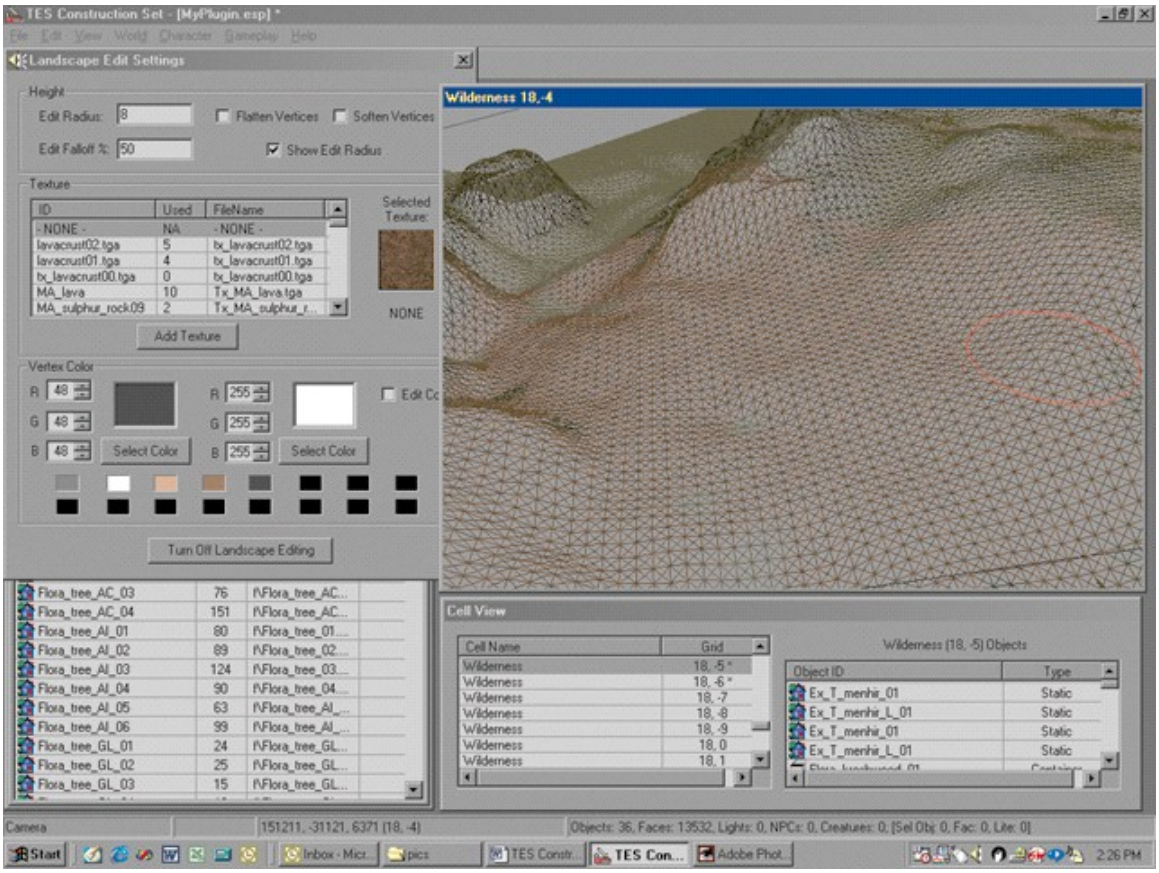


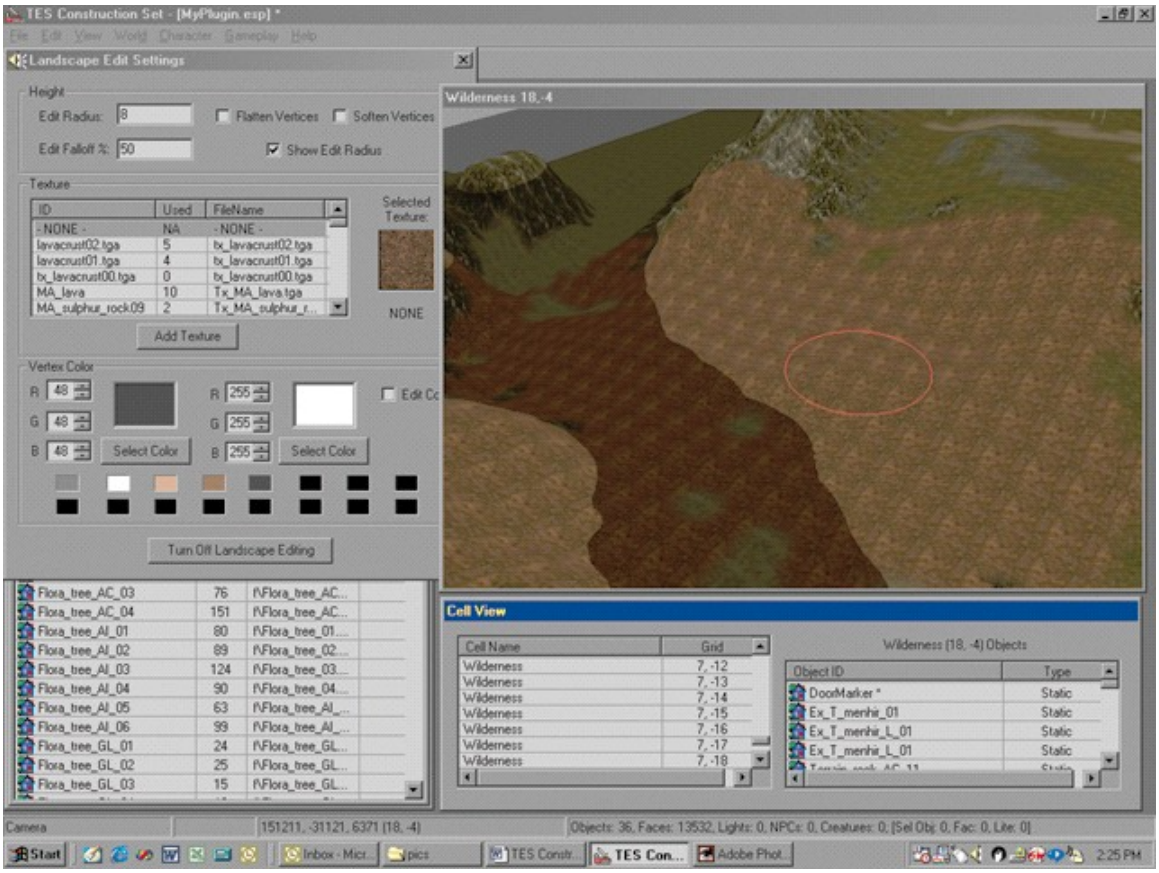

A red circle shows how big of an area you will be pulling up and down. By changing the Radius and Falloff %, you choose how the vertices move as you select and move the mouse up and down. Start pulling the landscape up so it's over the water.




By moving large sections of landscape up and then others down, you can sculpt the mesh like clay. Also try using the “smoothing” checkbox. This will allow you to smooth out areas to get rid of unwanted jaggedness that can occur from rapid height changes.

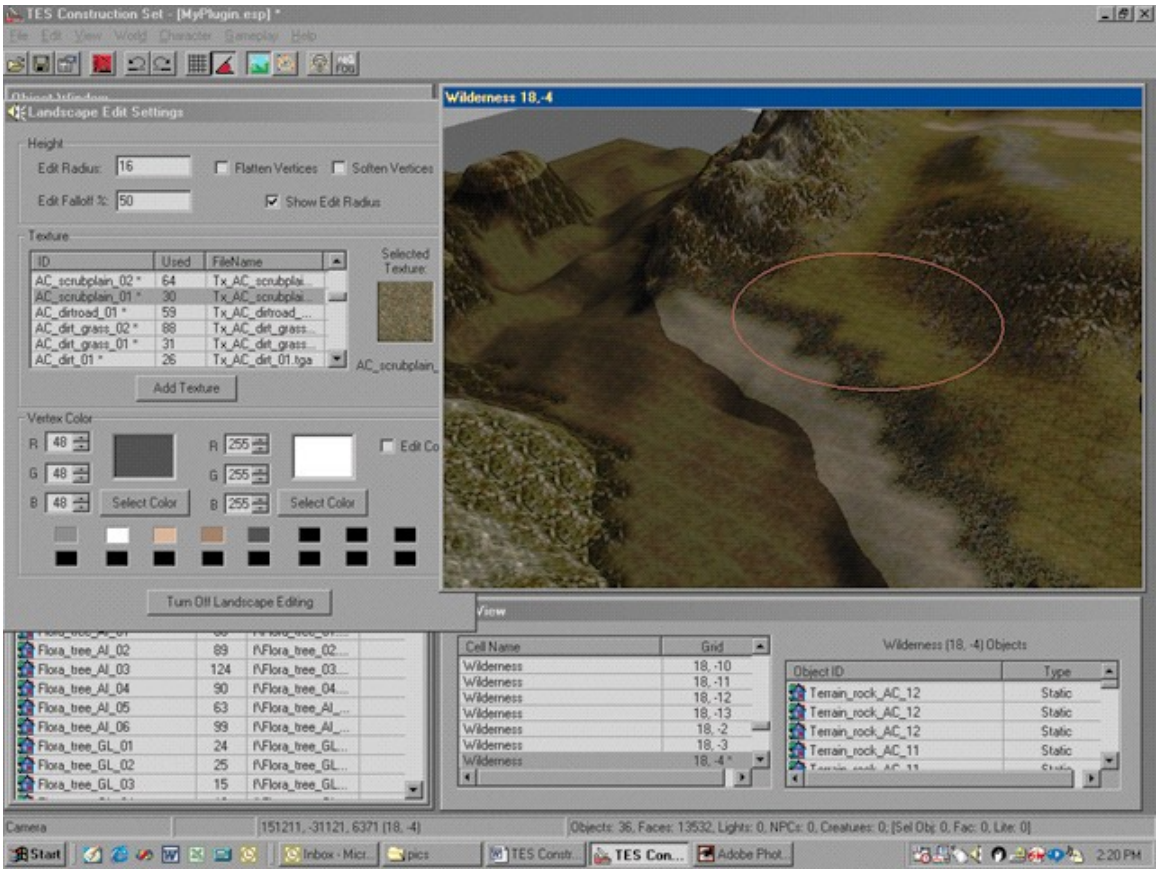
It is best to edit the heights in wireframe mode, but switch often to textured mode so you can see where the ocean height is hitting your area.


TES Construction Set - [MyPlugin.esp] *
File Edit View World Character Gameplay Help
Landscape Edit Settings
Height
Edit Radius: 8
Flatten Vertices
Soften Vertices
Edit Falloff %: 50
Show Edit Radius
Texture
ID
Used
FileName
- NONE -
NA
lavacrust02.tga
5
lavacrust01.tga
4
tx_lavacrust00.tga
0
MA_lava
10
MA_sulphur_rock09
2
Selected Texture:
NONE
Add Texture
Vertex Color
R 48
G 48
B 48
Select Color
R 255
G 255
B 255
Select Color
Turn Off Landscape Editing
Wilderness 18,-4
Cell View
Cell Name
Grid
Wilderness 18, -5
Wilderness 18, -6
Wilderness 18, -7
Wilderness 18, -8
Wilderness 18, -9
Wilderness 18, 0
Wilderness 18, 1
Wilderness [18, -5] Objects
Object ID
Type
Ex_T_menhir_01
Static
Ex_T_menhir_L_01
Static
Ex_T_menhir_01
Static
Ex_T_menhir_L_01
Static
Camera
151211, -31121, 6371 [18, -4]
Objects: 36, Faces: 13532, Lights: 0, NPCs: 0, Creatures: 0, [Sel Obj: 0, Fac: 0, Lite: 0]
Start
2:26 PM




**4. Texture the landscape.**

Once you have a shape you like, you'll want to texture it. Right clicking on the landscape paints it with your currently selected texture. Try to follow the shapes you have built. The landscape textures will automatically smooth into one another. If you get more than 2 textures hitting the same area, you'll want to move some things around and try to get the texture match on the corner, as only 2 textures look perfect hitting the same spot. A landscape texture covers 4 landscape vertices, so you may also want to adjust the shape of the landscape when you see where the textures fall.




5. **Place landscape foliage.**