WARLORDOCRACY

BUILDER'S MANUAL

Laughing Coyote Software
Last Updated: 24 December 2025

TABLE OF CONTENTS

Scripting Intro (home)

The Warlord Scripting Language (WSL) commands can be used in editor mode, as well as in the WSL script files using Notepad or another simple text editor.

While in Editor Mode, left click the mouse to select a tile and/or object, right click to open the command prompt. You can also hit delete to quickly delete the selected object (same as the "objDelete" WSL command). Hit F1 or ESC to exit Editor Mode.

Media Folders (home)

All of the game's assets are found in the media folder in the game's directory. The sub-folders are listed below. Images should all be PNG format with set transparency. Sound files are all WAV, and music files are all OGG.

If there are too many files in a single folder, you can make your own sub-folders to better organize your sprites, etc. The main campaign is divided into separate chapters, so I have separate dialogue and map script sub-folders for each chapter ("dialogue", "dialogue2", etc.).

If you want to make your own story called "Sword of Destiny" or some cheesey crap, you can make your own scripts folder called "scripts\swordOfDestiny" for example. Just be careful not to overwrite the default assets or you will break the main campaign story.

"Abilities" folder	:: PNG icon images for game abilities (32x32).
"Achieves" folder	:: PNG icon images for achievements (96x96).
"Docs" folder	:: All documents (like this one) that can be opened from game menu.
"Equip" folder	:: PNG animation images for equipped objects (drawn over mobiles).
"Extra" folder	:: Reference material such as variable lists (among other files).
"Fonts" folder	:: The TFF fonts the player can choose from in settings.
"Hud" folder	:: All PNG system images required for the game to run (windows, etc.).
"Music" folder	:: OGGs for music (logo, intro, victory, and gameover all required).
"Objects" folder	:: Initial WSL scripts for objects (run when object is loaded).
"Parties" folder	:: Exported parties, loaded with “loadParty” command (PAR format).
"Saves" folder	:: Player save files (GAM format).
"Screens" folder	:: PNG full-screen images like title, gameover, etc. (1024x768).
"Scripts" folder	:: WSL scripts (including globals.wsl and timePass.wsl).
"Sounds" folder	:: WAV sound files (everything in “hud” sub-folder are required).
"Sprites" folder	:: PNG sprites for all objects (one frame or sprite sheet).
"Stories" folder	:: New stories (GAM format, same as saved games).
"Tiles" folder	:: PNG map tiles (32x32).
"Weather" folder	:: PNG weather and map overlays (1024x768).

Default Sounds (home)

fail.wav	:: fail sound.
select.wav	:: menu select sound #1.
select2.wav	:: menu select sound #2.
bonus.wav	:: sound for earning points, learning abilities, etc.
inv.wav	:: sound for picking up or dropping items.
wealth.wav	:: sound for earning wealth.
objective.wav	:: sound for getting or completing an objective.

Default Skill Effects (home)

There are 5 skills in the game, each with a default effect. Mobile objects (characters) also have base stats like Max Health and Melee Damage, which you can see under the section “Object Commands (Mobile Type)” below, but upgrading skills will upgrade the base stats further.

The names of each skill can be changed with script commands, but their default effects cannot be changed. If the final skill's name is set to nothing, it will not appear in the HUD.

Upgrading a skill at a trainer requires 5 skill points, the same as unlocking an ability. This is only the default system. You can script custom systems for your own story using the scripts.

Vitality (skill 1) :: Each point is +5 points to Max Health.
Willpower (skill 2) :: Each point is +5 points to Max Meta.
Physique (skill 3) :: Each point is +5 to Max Load and +2 Melee Damage, +3% Parry.
Cunning (skill 4) :: Each point is +5 to Ranged Aim and +3% Melee Evasion.
Education (skill 5) :: No default effect, used only in scripts and to unlock abilities.

Default Scripts (home)

There are only two system scripts in the root script folder: globals and timePass. Everything else is called from map or object scripts. The game's opening should be called from the map script, which is set using the "mapScript" command. Checks specific to party members should be in the mobile's check script using the "objScrChk" command, etc.

scripts\globals.wsl :: Runs when starting new game, sets default global variables.
scripts\timePass.wsl :: Runs every 4 seconds on every map.

Script Variables (home)

Script variables range from 1 to 999, and they are kept track of in separate TXT files in the media\extras folder. They can be set to positive or negative integers, or 0 (default). The variable commands can be found below under the General Commands section.

They can be set with the "var" command (General Commands section), and increased or decreased with the “varUp” command (negative value to decrease). For example, to decrease variable #3 by 1, use the command "varUp=3,-1".

Then, to test variables in scripts, use the "if_var" boolean (listed in the Boolean Commands section). You can use the equals sign or the inequality signs to test variables. For example, to check if variable #20 is greater than 10, use the boolean "if_var>20,10".

In addition to setting or adjusting a variable by a given integer, you can also set/adjust variables to specific in-game values, such as player wealth or faction respect. For example, if you want to set variable #20 to the player's wealth, use the command "var=20,wealth". You can set variables (#1-999) to the following in-game values:

wealth :: set given variable to player's wealth.
points :: set given variable to player's skill points.
time :: set given variable to current time.
react1 (to 10) :: set given variable to faction's respect (ID 1 to 10).
skill1 (to 5) :: set given variable to current object's skill (ID 1 to 5).
resist1 (to 10) :: set given variable to current object's resistance (ID 1 to 10).

Other Notes (home)

-The story begins on whatever map you saved the story on with the "saveStory" command. The player object may or may not be on that map (or even exist), but if the player object dies, it's game over.

-The introduction of your story should be called in the map script of your first map. Map scripts are run immediately after the map is loaded (and again every 4 seconds). So you should set a variable in the map script so the intro is only called one time.

-The official stories begin on map #1, called "Limbo". The map script ("map\mapLimbo.wsl") immediately calls the introduction script ("misc\introCh1.wsl"), which handles character creation. A variable is then set so the introduction is only called one time.

-The "globals.wsl" script calls the "misc\encyclopedia.wsl" script, which sets all the encyclopedia entries. These are not saved in game files, but they are set in this script every time the game launches and the global script is called.

-Objects named "glimmer" are system objects, used for bringing attention to obtainable items. Do not name any custom objects "glimmer".

General Commands (home)

General commands don't require a selected object. They can be used any time in either game scripts or the world editor. Many commands can be set to negative to decrease values.

quit :: Quit the game immediately (without saving).
mainMenu=*int* :: Return to main menu (without saving, show gameover if parameter is 1).
saveStory=*str$* :: Save the story under given filename (remember to name all maps).
saveGame=*str$* :: Save the game under given filename (remember to name all maps).
loadStory=*str$,int* :: Load the story of given filename (if 2nd parameter=1, transfer party).
loadGame=*str$* :: Load the game of given filename.
lockStory=*str$* :: Lock story of given name (add story to lock.ini file).
unlockStory=*str$* :: Unlock story of given name (remove story from lock.ini file).
savePartyy=*str$* :: Save the current party under given filename;
set to "response" to set party name to last input window response.
loadParty=*str$* :: Load party of given filename into current game;
set to "response" to set party name to last input window response.
loadPartyList :: Bring up a list of parties player can choose to load.
gameoverScr=*str$* :: Call given script on gameover instead of returning to menu;
reset every time map changes, set to "none" to reset to default gameover.
storyNew :: Completely erase the current story and all maps, and rerun the globals script.
storyDemo=*int* :: Delete all maps after given map ID (to easily make demo versions).
storyFail=*int* :: Set to 1 to fail current story (and disable all saving), 0 to revert to normal.
storyName=*str$* :: Change the story name to given string.
changeMap=*int,int* :: Change the current map (1 to 30);
if optional second parameter is set to 1, don't automatically call map script.
disableAI=*int* :: Set to 1 to disable AI and check scripts for all objects.
disableSave=*int* :: Set to 1 to disable manual saving (for custom save systems).
msg=*str$* :: Write a message to the message window.
msgParty=*str$* :: Like msg, but only shows if current object is in party.
msgNew=*str$* :: Write a message only if not a repeat of last 6 messages.
msgErase :: Clear the message window.

var=*ID,int* :: Set script variable of ID 1 to 999 to given integer;
    you can set the second parameter to various in-game values (see section above).
varUp=*ID,int* :: Increase script variable of ID by given integer (negative to decrease).
varRand=*ID,int* :: Set variable of ID from 1 to given integer.
varResponse=*ID* :: Set variable of ID to previous dialogue response.
resetVars :: Reset all variables to 0.
resetTarget :: Reset target coordinants (for ability and item use scripts).
resetRep :: Reset all faction respect, reputations, and objectives.
time=*int* :: Set time to given integer (or “nightfall”, “sunrise”, “noon”, or “midnight”).
timePass=*int* :: Make time pass by given number of hours (1 hour=100 seconds).
daysPass=*int* :: Make days pass by given number (no status effects, etc.).
timeMax=*int* :: Set max time before it resets to 0 (default=600).
timeNight=*int* :: Set time that night falls (default=300).
timeStop=*int* :: Set to 1 to stop time from progressing until you change maps.
playSound=*filename$,int* :: play sound of given filename at given volume.
stopSounds=*int* :: Stop all sounds (if parameter is 1, stop ambient sound as well).
stopCombat :: Stop all combat targets on map and make all mobiles stop.
reputation=*str$,int* :: Add party reputation of given string;
    if optional integer paramter is set to 1, don't show any messages.
reputationErase=*str$* :: Erase party reputation of given string.
reputationEraseAll :: Erase all party reputations.
objective=*str$* :: Add objective of given string.
objectiveErase=*str$* :: Erase objective of given string.
objectiveEraseAll :: Erase all objectives.
factionReact=*str$,int* :: Set faction's reaction to percent ("objFaction" for object's faction);
    set first parameter to "all" to set reactions for all factions.
factionReactUp=*str$,int* :: Increase faction's reaction by percent (negative to decrease).
weather=*str$* :: Set current weather to image in the weather folder (or "clear" for no weather).
weatherMoveX=*int* :: Set weather movement X.
weatherMoveY=*int* :: Set weather movement Y.
weatherOpacity=*int* :: Set weather opacity (0 to 100, default=100).
weatherGlue=*int* :: Set to 1 for weather to be glued to screen (like fog).
weatherAim=*int* :: Set aiming modifier for ranged combat (-500 to 500, default=0).
screen=*str$* :: Set current background picture screen (or set to "none").
pause :: Pause the game with a screen open to make slideshows, chapter screens, etc.
goto=*str$* :: Goto label in same script that ends with a colon (“:”, don't include colon here).
camPos=*int,int* :: Position camera at given coordinants.
camShake=*int* :: Shake camera for given frames (300 frames max = 5 seconds)**.**
camFlash=int :: Flash camera for given number of frames (max 50).
camFlashRed=*int* :: Camera flash red tint (0 to 255).
camFlashGreen=*int* :: Camera flash green tint (0 to 255).
camFlashBlue=*int* :: Camera flash blue tint (0 to 255).
choice=*ID,str$* :: Set dialog window choices to given string (1 to 6);
    use these commands right before the "dialog" command below;
    if ESC key pressed, automatically choose choices called "(Cancel)" or "(Continue)";
    if choice begins with double quote and speaker has <2 Education, fuck up grammar.
choiceParty :: Set dialog window choices to the 5 main party members and “(Cancel)”.
choiceSprite=*ID,str$* :: Sprite that shows when each of the above options is highlighted.
dialog=*str$* :: Open dialog window with given string;
    player choices are set using the "choice" command above;
    the player's response can be checked with the "if_response=" boolean;
    "/r" starts a new line, "/n" is player's name, "/w" is player's wealth;
    "/p" is player's points, "/v???" is given variable (/v001, etc),
    "/d" is default object's name, "/t" is target's name (party member talking).

inputText=*str$* :: Open a dialog window with given string and text input;
the player's response can be checked with the "if_response=" boolean.
callScript=*filename$* :: Call script from the "scripts" folder (don't include WSL file extension).
callScriptIni=*filename$* :: Call ini script from the "objects" folder.
wealth=*int* :: Set party's wealth to given integer (or “v” and 3 digits for variable).
wealthUp=*int* :: Increase party's wealth by given integer.
points=*int* :: Set party's skill points by given integer (or “v” and 3 digits for variable).
pointsUp=*int* :: Increase party's skill points by given integer.
victoryMusic :: Play the "music\victory.ogg" music temporarily.
restingMusic :: Play the "music\resting.ogg" music temporarily.
partyPos=*int,int* :: Position the 5 main party members around given X, Y coordinants.
partyHeal=*int* :: Heal and recharge entire party (if parameter is 1, remove status effects).
partyDelete :: Delete all mobiles in the party.
partyInvDelete=*str$* :: Delete all items in party's inventory of given name;
do not include parameter to delete all inventory items of party members.
partyPhased=*int* :: Temporarily phase out entire party from existence;
cancels on map change, if selected object is in party, ignore this object.
phaseAll=*int* :: Phase in/out all objects on map (0=phase out, 1=phase in).
phaseAllName=*str$,int* :: Phase in/out all objects of given name.
deleteAll=*str$* :: Delete all objects on map of given name.
deleteAllCorpse :: Delete all dead mobiles from map.
destroyAll=*str$* :: Destroy all objects on map of given name.
factionAll=*str$,str$* :: Change faction of all objects on map of given name to given faction.
make=*str$* :: Make object of given INI script (from "objects" folder).
achieve=*str$* :: Unlock achievement of given Steam API.
encyclopedia=*str$* :: Display encyclopedia entry of given title.
majorBattle=*int* :: Set to 1 to start a major battle, 0 to end it (no waiting).
cancelDrop :: Use in drop scripts to cancel dropping the item.
dialogTimer :: Set time limit for dialogue window and minigames (in seconds).
minigameString=*str$* :: Set minigame string (see minigame booleans);
set to "seed" and number to make random lockpicking sequence of given length;
the sequence is based on object and map ID, so it never changes for same lock;
example: "minigameString=seed8" (make lock sequence with 8 tumblers).
testMaps :: Load first 30 maps back-to-back to easily test story integrity.
findObj=*str$* :: Find every instance of object in current story of given name;
set to "response" to search for player's dialogue response.
setTarget=*int,int* :: Set target coordinants for scripts like artillery;
can also be set to "obj" to set coordinants to currently-selected object.
refresh :: Refresh all objects on the screen (used after opening chests, etc.).
sync :: Sync the screen, useful after changing weather effects during dialogue.

Global Commands (home)

These commands set global names for skills, etc that shouldn't be changed during the game, but it is possible. They should be set in the "globals.wsl" script in the "scripts" folder.

pacesName=*str$* :: Set name of paces (tile) units.
weightName=*str$* :: Set name of weight units.
wealthName=*str$* :: Set name of wealth units.
pointsName=*str$* :: Set name of skill points.
damageName=*ID,str$* :: Set names of damage types (ID 1 to 10);
"body" damage does not wake mobiles from unconsciousness.

skillName=*ID,str$* :: Set names of skill types (ID 1 to 5).
factionName=*ID,str$* :: Set names of factions (ID 1 to 10).
statusName=*ID,str$* :: Set names of status effects (ID 1 to 20).
statusBonus=*ID,str$* :: Set first stat bonus, equal to status level;
set to either "maxHealth", "maxMeta", "maxLoad", "damage", "evasion", "aiming", "parry", or any one of the damage type names for resistance.
statusBonus2=*ID,str$* :: Second stat bonus (same options as first bonus).
statusPen=*ID,str$* :: Set first stat penalty (same options as first bonus).
statusPen2=*ID,str$* :: Second stat penalty (same options as first bonus).
statusRecover=*ID,float* :: Float recovery every second (negative to get worse);
could be 0.1 to recover 1 point every 10 seconds, for example.
statusDam=*ID,int* :: Amount of damage dealt each second by status.
statusDType=*ID,str$* :: Damage type dealt each second by status.
statusDrain=*ID,int* :: Meta drain every second by status.
statusRed=*ID,int* :: Sprite's red color correction (-255 to 255).
statusGreen=*ID,int* :: Sprite's green color correction (-255 to 255).
statusBlue=*ID,int* :: Sprite's blue color correction (-255 to 255).
statusSpeed=*ID,int* :: Status increases speed by this level (speed is 0 to 5);
set to negative to decrease speed.
statusConfused=*ID,int* :: Status causes occasional random movement if 1.
statusConcealed=*ID,int* :: Status conceals mobile from sight if 1;
set to 2 to toggle total invisibility (can only be spotted with infravision),
set to 3 for submerged beneath water, 4 for burrowed under slow tiles.
statusSpotting=*ID,int* :: Status allows mobile to see concealed objects if 1;
set it to 2 to toggle infravision (more effective), 3 is blindness.
statusBerserk=*ID,int* :: Status causes mobile to attack randomly if 1;
chance to change target to anything 5 spaces away (hostile or not);
set it to 2 to have mobile actively seek out and attack allies only.
statusNoTip=*ID,int* :: Status doesn't appear in tooltips until it reaches this amount;
set it to -1 for status to never appear in tooltips.
statusSleeping=*ID,int* :: Status means target is unconscious (upside-down sprite).
abilityName=*ID,str$* :: Name of ability (ID 1 to 100).
abilityImg=*ID,filename$* :: PNG filename of ability icon ("abilities" folder).
abilityDesc=*ID,str$* :: Encyclopedia topic that is the decription of object when examined.
abilityScript=*ID,filename$* :: WSL filename of ability script.
abilityTarget=*ID,str$* :: Set to either "none", "self", or "other".
abilityRange=*ID,int* :: Range of ability (if target is set to "other").
abilityCost=*ID,int* :: Meta cost to use ability.
abilityReq=*ID,str$* :: Required skill to learn ability, set to "none" to forbid learning.
abilityReqLvl=*ID,int* :: Required skill level to learn ability.
encTitle=*ID,str$* :: Encyclopedia entry title (1 to 9999).
encTopic=*ID,str$* :: Encyclopedia entry topic (1 to 9999).
encSprite=*ID,str$* :: Encyclopedia entry sprite (1 to 9999).
encText=*ID,str$* :: Encyclopedia entry text (1 to 9999).
achieveTag=*ID,str$* :: Achievement tag, same as Steam API (1 to 99).
achieveName=*ID,str$* :: Achievement name, same as Steam API (1 to 99).
achieveDesc=*ID,str$* :: Achievement description, same as Steam API (1 to 99).

Boolean Conditionals (home)

Booleans are for conditional branches and end with an "endif" command. Many "=" signs can be replaced with "<" or ">". If you want to check for not-equal, use "=" and then "else" on the next line. Booleans are only used in scripts, not the world editor. Nothing is case-sensitive.

exit :: Exit the current script.
reset :: Restart the current script.
else :: Switch to opposite conditional.
endif :: End current conditional branch.
if_storyName=*str$* :: Check for story name.
if_mainStory :: Check if story is one of the main chapters.
if_failed :: Check to see if the main story has been failed.
if_demo :: Check to see if demo version is enabled.
if_editor :: Check to see if world editor is enabled.
if_touchscreen :: Check to see if touchscreen setting is enabled.
if_response=*str$* :: Check for player's response from dialog window;
    if normal window, it's from "1" to "6", if text input then it can be any string;
    use "none" to test for empty text input.
if_random=*int* :: Check for random percent chance (1 to 99).
if_var=*ID,int* :: Check for script variable of given ID (replace with "<" or ">").
if_numObj=*int* :: Check number of objects on current map (replace with "<" or ">").
if_numCombat=*int* :: Check number of combatants on current map (replace with "<" or ">").
if_time=*int* :: Check for current game time (600 max by default, 25 ticks=100 seconds);
    can also be set to "night", "day", "sunrise", "noon", "midnight", "daybreak";
    "morning", "afternoon", "evening", or "late".
if_daysPassed=*int* :: Check number of days passed since story began.
if_weather=*str$* :: Check for current weather overlay ("clear" for no weather).
if_wealth=*int* :: Check for party's wealth.
if_points=*int* :: Check for party's points.
if_partyLvl=*int* :: Check for party level.
if_partySize=*int,int* :: Check number of characters in party (including henchmen);
    if optional 2nd parameter is 1, only count main characters, if 2, only count henchmen.
if_partyAI=*int* :: Check for current party AI setting (0 to 5);
    can also check for strings "off", "basic", "ability", "healing", "advanced", or "maximum";
    checking for "ability" and "healing" also includes "advanced" and "maximum" in check.
if_partyWeapons :: Check if any of the 25 party members is carrying a weapon;
    ignores items with weapon type set to "hidden".
if_partyHave=*str$,int* :: Check if party has given item (and optionally given amount).
if_partyResponsive :: Check if the 5 main party members are responsive (not unconscious).
if_majorBattle :: Check if major battle is currently happening ("majorBattle" command).
if_mapName=*str$* :: Check for name of map.
if_mapOverlay :: Check map overlay.
if_mapDark :: Check if all 3 map colors are at or below 120.
if_combat :: Check if any mobiles are engaged in combat on the current map.
if_waiting :: Check if "timePass" command is currently happening (to skip certain checks).
if_factionReact=*str$,int* :: Check given faction name's reaction.
if_reputation=*str$* :: Check if party has given reputation.
if_damage=*int* :: Check for damage dealt (used only in object damage, die, and hit scripts).
if_objCount=*str$,int* :: Check for number of objects on map of given name;
    remember that the equals sign can be replaced with "<" or ">".
if_objCountForces=*str$,int* :: Check for number of mobiles of given faction name.

if_minigame=*str$,int* :: Hangman minigame with given word and number of attempts;
you can set a time limit for most minigames with “dialogTimer” command.
if_minigame2=*str$,str$* :: Word Scramble minigame with given word and topic.
if_minigame3=*int* :: Lockpick minigame with given tensile strength;
L/R sequence is set with "minigameString" command.
if_minigame4=*str$,int* :: Music minigame with given instrument and tempo (1-5);
music string is set with "minigameString" command (0-9, 0 meaning skip a beat).
if_minigame5=*int,int* :: Fishing minigame with given difficulty and agitation (both 1-6).
if_minigame6=*int,int* :: Gambling minigame with given honesty level (0-10);
if second parameter is set to 1, reveal opponent's honesty level.

Booleans (Object-Specific) (home)

The following booleans must have a specific object selected, either the default script object, one chosen with a “pick” command, or one selected in the world editor.

if_objName=*str$* :: Check for name of currently-selected object;
    set to "default" to test if object has same name as the default script object.
    set to "none" to test if object does not exist (doesn't have a name).
if_objNameAdj=*str$* :: Check for first word of object's name (often an adjective).
if_objNameSuff=*str$* :: Check for last word of object's name (often main noun).
if_objSprite=*str$* = Check for object sprite's filename.
if_objFlip :: Check if object's sprite is flipped horizontally.
if_objID=*int* :: Check object ID (1 to 10000).
if_objMaterial=*str$* :: Check for material of currently-selected object;
    type “stone” and "metal" will always damage weapons (instead of 20% chance).
if_objType=*str$* :: Check for object's type ("normal", "item", "mobile", "usable" or "effect).
if_objPosX=*int* :: Check for object's X position;
    set parameter to "default" to test position against default object.
    set parameter to "max" to test position against map size.
if_objPosY=*int* :: Check for object's Y position (same special parameters as above).
if_objHE=*int* :: Check for health of current object.
if_objME=*int* :: Check for meta of current object.
if_objHEMax=*int* :: Check for max health of current object.
if_objMEMax=*int* :: Check for max meta of current object.
if_objHEPer=*int* :: Check for percent health of current object (based on max).
if_objMEPer=*int* :: Check for percent meta of current object (based on max).
if_objLoad=*int* :: Check for current weight load of current object.
if_objLoadFree=*int* :: Check for current available (unused) load of current object.
if_objLoadPer=*int* :: Check for percent current weight load (based on max).
if_objLoadMax=*int* :: Check for object's max load (<0 means inventory disabled).
if_objLoadTile=*int* :: Check for max load on same tile as object (for puzzles mostly);
    item objects add up normally, mobile objects all have a load of 100.
if_objDelay=*int* :: Check object's delay (negative is timer).
if_objDelay2=*int* :: Check for number of ticks since object’s normal attack delay;
    used in AI scripts to make sure abilities aren’t used immediately after attacking.
if_objDelayAtt=*int* :: Check for object’s delay after a normal attack.
if_objFaction=*str$* :: Check for object's faction;
    set to "none" for faction #0, "default" for default object's faction.
if_objInWater :: Check if object is currently in the water.
if_objInSand :: Check if object is currently on a sand tile (mapSlow=1).
if_objInGrass :: Check if object is currently on a tall grass tile (mapSlow=2).
if_objIndoors :: Check if object is currently indoors.
if_objHighGround=*int* :: Check if object is on high ground (0 to 5).
if_objOnTile :: Check if object is on given tile image;
    checks for first 4 letters only, (ex. "dirt", "sand", “gras”).
if_objPhased :: Check if object is phased out.
if_objParent :: Check if object is being carried by another.
if_objTargetFacing :: Check if object's target is currently facing current object.
if_objFacingDefault :: Check if current object is facing default object.
if_objHaveDest :: Check if object has a current destination.

if_objItmType=*str$* :: Check for item type ("normal", "usable", "weapon", "wealth", or "storage").
if_objItmDam=*int* :: Check for item's damage as a weapon.
if_objItmAim=*int* :: Check for item's aim bonus as a weapon.
if_objItmRange=*int* :: Check for item's range.
if_objItmDelay=*int* :: Check for item's delay (60 ticks = 1 second).
if_objItmWeight=*int* :: Check for item's weight.
if_objItmValue=*int* :: Check for item's value.
if_objItmWpnType=*str$* :: Check item weapon type string.
if_objItmUseType=*str$* :: Check item usable type string.
if_objMobResponsive :: Check if object is in party and able to be controlled.
if_objMobSleeping :: Check if object is sleeping.
if_objMobTarget :: Check if mobile currently has a target.
if_objMobDir=*str$* :: Check for mobile direction ("N", "S", "E", or "W").
if_objMobPlayer :: Check if mobile is the main player.
if_objMobParty :: Check if mobile is in party.
if_objMobPartyMain :: Check if mobile is in main party (not henchman).
if_objMobSpecies=*str$* :: Check mobile's species.
if_objMobFlying :: Check if mobile is flying.
if_objMobAnimal :: Check if mobile is an animal (no equipment or talking).
if_objMobAquatic=*int* :: Check for mobile aquatic value (see *objMobAquatic* command).
if_objMobHave=*str$,int* :: Check if mobile has item of given name;
the second integer is the count of the item in the inventory (at least).
if_objMobHaveType=*str$,int* :: Check if mobile has item of given weapon/usable type.
if_objMobUnarmed :: Check if mobile is not carrying a weapon.
if_objMobSkill=*str$,int* :: Check the level of mobile's skill of given name.
if_objMobAbility=*str$* :: Check if mob knows given ability.
if_objMobAbilityFull :: Check if mob knows 10 abilities (max amount).
if_objMobRange=*int* :: Check mob's attack range.
if_objMobDamage=*int* :: Check mob's attack damage.
if_objMobAim=*int* :: Check mob's ranged attack aiming.
if_objMobEvade=*int* :: Check mob's evasion.
if_objNear=*str$,val* :: Check if object of given name is within given paces of current object;
set to "default" to test for objects of the same name as current object.
if_objNearSight=*str$,val* :: Same as "if_objNear" but also check for line of sight;
set to "default" to test for objects of the same name as current object.
if_objNearFast=*int* :: Same as "if_objNear" but always pick lowest ID for faster processing.
if_objNearSightFast=*int* :: Same as "if_objNearSight" but always pick lowest ID.
if_objNearMob=*val,val* :: Check if mobile of a different name is in given range and LOS;
if optional 2nd parameter is set to 1, ignore line of sight check.
if_objNearHostile=*str$,val* :: Same as "if_objNearMob" but only for hostile mobiles.
if_objNearFriend=str$*,val* :: Same as "if_objNearMob" but only for mobiles of same faction.
if_objNearTarget=*val* :: Check if current object's target is within range and line of sight.
if_objNearCorpse=*val* :: Check if corpse is within given paces of current object.
if_objNearWpn=*val* :: Check if weapon is within given paces of current object.
if_objNearParty=*val* :: Check if any party member is within given paces of current object.
if_objNearPartyAll=*val* :: Check if all party members are within given paces of current object.
if_objNearPlayer=*val* :: Check if main player is within given paces of current object.

Map Commands (home)

These commands alter the properties of the current map. These are often set/changed in the map script, such as setting the color tints during different times of day.

mapName=*str$* :: Set name of current map to given string.
mapSize=*int,int* :: Set size of current map to given X and Y tiles respectively.
mapScript=*filename$* :: Set filename of map's script (run every 3 seconds).
mapMusic=*filename$* :: Set map's music (set to "none" for no music);
tracks beginning with "combat" will loop, otherwise it only plays once.
mapMusicNight=*filename$* :: Set map's music at night (time is 301 to 600).
mapAmb=*filename$* :: Set map's ambient sound (set to "none" for no sound).
mapAmbNight=*filename$* :: Set map's ambient at night (time is 301 to 600).
mapOverlay=*string* :: Set map overlay (from weather folder), or “none” to disable;
if set, map is considered indoors, so day/night cycle and weather disabled.
mapRed=*int* :: Set map red tint (0 to 255).
mapGreen=*int* :: Set map green tint (0 to 255).
mapBlue=*int* :: Set map blue tint (0 to 255).
mapTile=*filename$* :: Set tile at selected position in world editor to filename;
if the image is larger than 32x32 pixels, the tile will automatically be animated.
mapTile3x3=*filename$* :: Set 3x3 tiles at selected tile in world editor to filename.
mapTile5x5=*filename$* :: Set 5x5 tiles at selected tile in world editor to filename.
mapTile9x9=*filename$* :: Set 9x9 tiles at selected tile in world editor to filename.
mapTileAll=*filename$* :: Set all tiles on map to filename.
mapBlock=*int* :: If 1, tile at selected tile in word editor blocks movement;
this will also automatically set mapBlockVis to 1 (you can turn it off after).
mapBlock3x3=*int* :: Same as "mapBlock" but does so in a 3x3 area.
mapBlock5x5=*int* :: Same as "mapBlock" but does so in a 5x5 area.
mapBlock9x9=*int* :: Same as "mapBlock" but does so in a 9x9 area.
mapBlockAll=*int*:: Same as "mapBlock" but does so for entire map.
mapBlockVis=*int* :: If 1, tile at selected tile in word editor blocks visibility.
mapBlockVis3x3=*int* :: Same as "mapBlockVis" but does so in a 3x3 area.
mapBlockVis5x5=*int* :: Same as "mapBlockVis" but does so in a 5x5 area.
mapBlockVis9x9=*int* :: Same as "mapBlockVis" but does so in a 9x9 area.
mapBlockVisAll=*int*:: Same as "mapBlockVis" but does so for entire map.
mapBlockFly=*int* :: If 1, tile at selected tile in word editor blocks flight.
mapBlockFly3x3=*int* :: Same as "mapBlockFly" but does so in a 3x3 area.
mapBlockFly5x5=*int* :: Same as "mapBlockFly" but does so in a 5x5 area.
mapBlockFly9x9=*int* :: Same as "mapBlockFly" but does so in a 9x9 area.
mapBlockFlyAll=*int*:: Same as "mapBlockFly" but does so for entire map.
mapHigh=*int* :: If 1 or more, tile is flagged as high ground (for ranged attacks);
elevation can go up to 5, giving range and damage bonus to lower elevation.
mapHigh3x3=*int* :: Same as "mapHigh" but does so in a 3x3 area.
mapHigh5x5=*int* :: Same as "mapHigh" but does so in a 5x5 area.
mapHigh9x9=*int* :: Same as "mapHigh" but does so in a 9x9 area.
mapHighAll=*int*:: Same as "mapHigh" but does so for entire map.
mapWater=*int* :: If 1, tile is water.
mapWater3x3=*int* :: Same as "mapWater" but does so in a 3x3 area.
mapWater5x5=*int* :: Same as "mapWater" but does so in a 5x5 area.
mapWater9x9=*int* :: Same as "mapWater" but does so in a 9x9 area.
mapWaterAll=*int*:: Same as "mapWater" but does so for entire map.

mapSlow=*int* :: If 1, tile is rough and slows movement (for sand enemies);
if 2, tile obscures lower half of body like water, but doesn't slow movement (tall grass).
mapSlow3x3=*int* :: Same as "mapSlow" but does so in a 3x3 area.
mapSlow5x5=*int* :: Same as "mapSlow" but does so in a 5x5 area.
mapSlow9x9=*int* :: Same as "mapSlow" but does so in a 9x9 area.
mapSlowAll=*int*:: Same as "mapSlow" but does so for entire map.
mapIndoors=*int* :: If 1, tile is indoors (only used with "if_indoors".boolean).
mapIndoors3x3=*int* :: Same as "mapIndoors" but does so in a 3x3 area.
mapIndoors5x5=*int* :: Same as "mapIndoors" but does so in a 5x5 area.
mapIndoors9x9=*int* :: Same as "mapIndoors" but does so in a 9x9 area.
mapIndoorsAll=*int*:: Same as "mapIndoors" but does so for entire map.
mapStepSound=*int* :: Play given sound when tile is stepped on (or “none” to cancel).
mapStepSound3x3=*int* :: Same as "mapStepSound" but does so in a 3x3 area.
mapStepSound5x5=*int* :: Same as "mapStepSound" but does so in a 5x5 area.
mapStepSound9x9=*int* :: Same as "mapStepSound" but does so in a 9x9 area.
mapStepSoundAll=*int*:: Same as "mapStepSound" but does so for entire map.
mapDelete :: Completely delete the current map and reset all variables.
mapRandom :: Erase and randomize map based on list of options.
mapCopy=*int* :: Copy map of given ID over current map.
mapCopyFrom=*string$,int* :: Copy map from given story of given ID over current map.
mapShift=*int,int* :: Shift all tiles and objects on map by given X and Y tiles.

Pick Commands (home)

These commands pick the current object so it can be altered with the object commands below. Objects can also be picked in the world editor, but these can be used in scripts as well as the world editor.

pick=*int* :: Pick object of given ID (1 to 10000), or 0 to unselect an object.
pickName=*str$* :: Pick random object on the map of given name.
pickParty=*int* :: Pick party member (1 to 25), leave blank for random main party (1 to 5).
pickDefault :: Pick the object that started the current script.
pickPlayer :: Pick the main player object.
pickTarget=*int* :: Pick the current object's target;
if optional parameter is set to 1, only pick target if currently in combat.
pickEquip :: Pick the current object's equipped item.
pickParent :: Pick the current object's parent (reverse of "pickEquip").
pickLast :: Pick the previously-selected object.
pickTile=*int,int* :: Pick tile of given X+Y coordinants to edit with map commands.

Object Commands (home)

These commands alter the properties of a specific object. The selected object can be changed with the pick commands above, or selected in the world editor. All objects must have a name, but they cannot be interacted with if they are of type "normal".

objName=*str$* :: Set current object's name;
"objName=response" will set name to previous window response.
objNameAdj=*str$* :: Add adjective word before object's name.
objNameAdjDelete=*str$* :: Delete adjective word from object's name.
objSprite=*filename$* :: Set current object's sprite.
objPhased=*int* :: Set to 1 to temporarily phase object out of existence.
objPos=*int,int* :: Immediately position object at given X, Y coordinants;
alternatively, set single parameter to "default" to position on to default object;
set single parameter to "target" to position on to target object.
objPosChange=*int,int* :: Immediately change position by given amounts.
objDest=*int,int* :: Set destination to given X, Y coordinants.
objDestLast :: Set destination to the coordinants of previously-selected object.
objMove=*str$*,*int* :: Move given number of tiles in given direction;
set to either "N", "S", "E", "W", "NE", "NW", "SE", "SW", "random", or "forward".
objStop :: stop object in its tracks (cancel destinations).
objConcealed=*int* :: set to 1 for object to be concealed (need spotting status to see);
2 for total invisibility, 3 for submerged in water, 4 for burrowed in slow terrain.
objDrawPri=*str$* :: Set draw priority relative to other objects.
set to "normal", "over", "under" (items), or "bottom" (corpses, bloodstains);
can also set to "top" to draw over everything else (unexplored fog of war).
objFlip=*str$* :: Set to 1 to flip object's sprite horizontally.
objType=*str$* :: Set to either "normal", "item", "mobile", "usable", or "effect";
by default, all mobile objects have all resistances set to 0%.
by default, all non-mobile objects have all resistances set to 100%.
by default, all item objects objects have draw priority set to "under".
objDesc=*str$* :: Set object's encyclopedia entry when examined.
objMaterial=*str$* :: Set object's material to given string.
objHEMax=*int* :: Set maximum health of object (without Vitality ability).
objMEMax=*int* :: Set maximum meta of object (without Willpower ability).
objHEMaxUp=*int* :: Increase max health of object (negative to decrease).
objMEMaxUp=*int* :: Increase max meta of object (negative to decrease).
objTemp=*int* :: Set to 1 to have object deleted upon leaving the map.
objEditMode=*int* :: Set to 1 to make object only visible in Editor Mode (waypoints, etc.);
set to 2 to make object invisible in Editor Mode (unexplored fog of war).
objTravel=*int* :: Set to 1 to make object a travel point (transition between maps);
object will appear as “Travel” in tooltips to hide real name that is used in scripts.
objFaction=*str$* :: Set object's faction to given faction name.
objResist=*str$,int* :: Set object's resistance percent to given amount (-500 to 500);
if 2nd parameter is "v" followed by 3 digits, set to variable (ex. v001);
set first parameter to "all" to set resistance to all damage types.
objResistUp=*str$,int* :: Increase object's resistance percent by given amount.
objFrameSize=*int* :: Set frame size for each frame on sprite sheet;
Set to 0 for no animations.
objFrameRand :: Set object to random frame.
objSpeed=*int* :: Set object's movement speed (0 to 6);
also determines speed of animations, 0 speed means the object cannot move.
objSpeedUp=*int* :: Increase object's movement speed.

objLoadMax=*int* :: Set mobile's base max weight load;
-1 to disable inventory and exclude from crime checks (mostly for livestock).
objLoadMaxUp=*int* :: Increase mobile's base max weight load.
objBlock=*int* :: Set to 1 to have object block movement, 2 to block only non-party members;
3 to block complex AI pathfinding, 4 to ignore pathfinding (like unlocked doors).
objBlockVis=*int* :: Set to 1 to have object block visibility.
objBlockFly=*int* :: Set to 1 to have object block flight.
objColSizeX=*int* :: Set X collision size in tiles (default=0);
X collision size counts from both east and west, so 1 would actually be 3.
objColSizeY=*int* :: Set Y collision size in tiles (default=0);
Y collision size counts only behind the object, so 1 would actually be 2.
objSelSizeX=*int* :: Set X selection size in pixels when selecting object with pointer;
If default 0, selection size is set to the size of the sprite.
objSelSizeY=*int* :: Set Y selection size in pixels when selecting object with pointer;
If default 0, selection size is set to the size of the sprite.
objTable=*int* :: Set to >1 for object to be a table;
items can be set on it even if block is set to 1;
items on table will appear given number of pixels higher.
objWaypoint=*int* :: Set to 1 to have object act as a waypoint for AI (doorways, etc.);
waypoints extend 1 tile south and 2 tiles north to accomodate for doorways.
objShop :: Shop with given object.
objTrain :: Train with given object as the trainer;
if optional parameter is >0, set skill max if there is no trainer mobile (20 max).
objCreateChar=*int* :: Bring up character creation screen (parameter is max skill).
objScrUse=*filename$* :: Set name of script run when object is used;
set to "none" to cancel any script (like the rest of these commands).
objScrAtt=*filename$* :: Set name of script run when object attacks.
objScrDam=*filename$* :: Set name of script run when object is damaged.
objScrCol=*filename$* :: Set name of script run when object collides.
objScrGet=*filename$* :: Set name of script run when object is taken.
objScrDrp=*filename$* :: Set name of script run when object is dropped.
objScrDie=*filename$* :: Set name of script run when object dies.
objScrSel=*filename$* :: Set name of script run when object is selected.
objScrAI=*filename$* :: Set name of script run every second if mobile.
objDamage=*int,str$* :: Damage current object by given amount, of given damage type.
objUseAbility=*str$* :: Use ability of given name at target.
objUse :: Use the object.
objUseSelf :: Run the object's own use script.
objExamine :: Examine the object.
objAttack=*int* :: Make object attack it's current target (2nd parameter optional);
2nd parameter: 1=ignore evade, 2=ignore parry, 3=ignore both.
objThrow=*int* :: Throw currently-equipped object to targeting coordinant;
if the optional parameter is set to 1, don’t check if there is room to drop.
objStopAttackers :: Stop all other mobiles from attacking current object.
objDestroy :: Destroy/kill current object automatically.
objDelete :: Delete current object.
objCopy :: Copy current object (to same location).
objDrop :: Drop object if it’s in a mobile’s inventory (can be used on item or mobile).
objTeleport=*int* :: Teleport object to different map (1 to 100, same location).
objDelay=*int* :: Set current object attack delay (or set to "attack" for normal attack delay).
objCooldown=*int* :: Set object cooldown for item and ability use, as well as delay.
objCrimeCheck=*val* :: Check for witnesses, decrease faction respect by given amount;
mobiles with max load set to -1 do not count as witnesses (mostly livestock).
objArtillery=*str$,int* :: Player selects target for object's attack script (for artillery, etc.);
string is direction ("N","S","E","W", or "C" for center), integer is range.

objDetectName=*str$,int* :: Detect objects of given name within given range;
show glimmers and display how many in game messages.
objDetectMaterial=*str$,int* :: Detect objects of given material within given range.
objDetectSpecies=*str$,int* :: Detect objects of given species within given range.
objMissile=*str$* :: Draw missile of given sprite to current object from default object.
objKnockback=*int* :: Force object backwards from target (-1 to force forwards).
objGive=*filename$,int* :: Give item of given INI script to current object;
the optional second parameter is the object count.
objInvDelete :: Delete all items in current object's inventory.
objSound :: Play sound attached to current object (prevents voice overlap).

Object Commands (Item Type) (home)

These commands deal with objects of the "item" type only (obtainable). Therefore, an object of item type must be selected.

objItmSprite=*filename$* :: Set item's overlapping sprite while equipped;
overlapping equip sprites around found in the "equip" folder.
objItmAnim=*int* :: Set item's animation when used (mostly for weapons).
0=idle, 1=use, 2=thrust, 3=walk, 4=slash, 5=shoot, 6=die.
objItmType=*str$* :: Set item's type (as an inventory item);
set to either "normal", "usable", "weapon", "wealth", or "storage";
when obtained, "wealth" items are deleted and the value added to player's wealth.
objItmTarg=*str$* :: Set item's target type if usable to "self" or "other".
objItmWpnType=*str$* :: Set item's weapon type string (for scripts only).
objItmUseType=*str$* :: Set item's usable type string (for scripts only).
objItmDam=*int* :: Set item's damage as a weapon to given amount.
objItmDamUp=*int* :: Increase item's damage by given amount.
objItmDType=*str$* :: Set item's damage type to given damage name.
objItmParry=*int* :: Set item's parry resistance as a weapon to given amount.
objItmParryUp=*int* :: Increase item's parry by given amount.
objItmRange=*int* :: Set item's range (if weapon or usable).
objItmRangeUp=*int* :: Increase item's range by given amount.
objItmAim=*int* :: Set item's aim bonus to given amount.
objItmAimUp=*int* :: Increase item's aim bonus by given amount.
objItmDelay=*int* :: Set item's delay to given amount (60 game ticks = 1 second);
actual delay is now double this value so combat lasts longer.
objItmDelayUp=*int* :: Increase items dleay by given amount.
objItmAmmo=*int* :: Set item's ammo type to given ID.
objItmAmmoReq=*int* :: Set item's required ammo type to given ID;
set to negative to only show the ammo missile sprite.
objItmWeight=*int* :: Set item's weight to given amount.
objItmWeightUp=*int* :: Increase item's weight by given amount.
objItmValue=*int* :: Set item's value to given amount.
objItmValueUp=*int* :: Increase item's value by given amount.

Object Commands (Mobile Type) (home)

These commands deal with objects of the mobile type only (characters). Therefore, an object of "mobile" type must be selected.

objMobCharScreen :: Show character screen of given mobile object.
objMobSpecies=*str$* :: Set mobile's species to given string;
set to "hidden" to make mobile untargetable (off-screen arrows, etc.).
objMobMelee=*int* :: Set mobile's base melee damage bonus.
objMobMeleeUp=*int* :: Increase mobile's base melee damage bonus.
objMobEvade=*int* :: Set mobile's base melee evasion.
objMobEvadeUp=*int* :: Increase mobile's base melee evasion.
objMobParry=*int* :: Set mobile's base parry resistance (frontal melee damage).
objMobParryUp=*int* :: Increase mobile's base parry resistance.
objMobAim=*int* :: Set mobile's base ranged aiming (setting to 100 cancels evasion).
objMobAimUp=*int* :: Increase mobile's base ranged aiming.
objMobSkill=*str$,int* :: Set mobile's skill of given name to given amount ("all" for all skills);
if 2nd parameter is "v" followed by 3 digits, set to variable (ex. v001).
objMobSkillUp=*str$,int* :: Increase mobile's skill.
objMobAbility=*str$,int* :: Learn ability of given name for mobile if set to 1;
forget the ability if set to 0.
objMobStatus=*str$,int* :: Set mobile's status of given name to given amount;
if 0, automatically cure status, otherwise only change to higher amount.
objMobStatusUp=*str$,int* :: Increase mobile's status.
objMobStatusNone :: Remove all status effects.
objMobDType=*str$* :: Set mobile's unarmed damage type to given string.
objMobRange=*int* :: Set mobile's unarmed range (1 or 2 is considered melee).
objMobRangeSpr=*int* :: Set ammo type for missile sprite for unarmed range attack.
objMobAnim=*str$* :: Play given animation (idle, use, thrust, slash, shoot, walk, death).
objMobAquatic=*int* :: Set to 1 to allow swimming, 2 for no swimming speed penalty,
3 for water only (no speed penalty); 4 for slow tiles only (sand, shallows, etc.);
5 for no slow tile penalty, 6 for swimming and no penalities at all.
objMobFlying=*int* :: Set to 1 if mobile is flying, 2 if out of melee attack reach.
objMobAnimal=*int* :: Set to 1 if mobile is animal (no equipment or talking allowed).
objMobFlee=*int* :: Set health percent at which mobile flees from target.
objMobLowAI=*int* :: Set to 1 to set AI to low (better framerate, farm animals, etc.);
objMobDir=*int* :: Set mobile's direction to 0, 1, 2, or 3;
respectively, this can be "N", "S", "E", or "W".
objMobParty=*int* :: Set to 0 to leave party, 1 to join main party, 2 to be a henchman;
henchmen can be controlled, but they do not follow the party from the map.
objMobPlayer=*int* :: Set to 1 to one to set object to be the main player.
objMobUse :: Make mobile object use its current target (regardless of distance);
this is often used for auto-dialogue when walking into a room, etc.
objMobGet :: Make mobile object get the target (don't check range).
objMobDrop :: Make mobile object drop currently-equipped item at feet.
objMobDropAll :: Make mobile object drop all items at feet.
objMobAttack :: Make mobile object attack its current target (must be in range).
objMobTarget=*str$* :: Make mobile object target object of given name.
objMobTargetPlayer :: Make mobile object target current main player.
objMobTargetParty=*int* :: Make mobile object target random main party member.
objMobTargetDefault :: Make mobile object target default object (script starter).
objMobTargetLast :: Make mobile object target the previously-selected object.
objMobTargetNone :: Cancel mobile's current target.

objMobFindWater=*int* :: Set destination to nearby water tile;
set parameter to 2 to look for slow tiles, and 3 to look for tall grass.
objMobEquipMelee :: Mobile switches to most powerful melee weapon.
objMobEquipRanged :: Mobile switches to most powerful ranged weapon.
objMobLunge :: Mobile lunge animation (for monsters that have no attack frames).
objMobAnimDefault :: Set all sprite frames to default mobile sprite sheets.
objMobAnimIdleN=*int,int* :: Set start and end sprite frames for idle animation (north).
objMobAnimIdleS=*int,int* :: Set start and end sprite frames for idle animation (south).
objMobAnimIdleE=*int,int* :: Set start and end sprite frames for idle animation (east).
objMobAnimIdleW=*int,int* :: Set start and end sprite frames for idle animation (west).
objMobAnimUseN=*int,int* :: Set start and end sprite frames for use animation (north).
objMobAnimUseS=*int,int* :: Set start and end sprite frames for use animation (south).
objMobAnimUseE=*int,int* :: Set start and end sprite frames for use animation (east).
objMobAnimUseW=*int,int* :: Set start and end sprite frames for use animation (west).
objMobAnimThrustN=*int,int* :: Set start and end sprite frames for thrust animation (north).
objMobAnimThrustS=*int,int* :: Set start and end sprite frames for thrust animation (south).
objMobAnimThrustE=*int,int* :: Set start and end sprite frames for thrust animation (east).
objMobAnimThrustW=*int,int* :: Set start and end sprite frames for thrust animation (west).
objMobAnimSlashN=*int,int* :: Set start and end sprite frames for slash animation (north).
objMobAnimSlashS=*int,int* :: Set start and end sprite frames for slash animation (south).
objMobAnimSlashE=*int,int* :: Set start and end sprite frames for slash animation (east).
objMobAnimSlashW=*int,int* :: Set start and end sprite frames for slash animation (west).
objMobAnimShootN=*int,int* :: Set start and end sprite frames for shoot animation (north).
objMobAnimShootS=*int,int* :: Set start and end sprite frames for shoot animation (south).
objMobAnimShootE=*int,int* :: Set start and end sprite frames for shoot animation (east).
objMobAnimShootW=*int,int* :: Set start and end sprite frames for shoot animation (west).
objMobAnimWalkN=*int,int* :: Set start and end sprite frames for walk animation (north).
objMobAnimWalkS=*int,int* :: Set start and end sprite frames for walk animation (south).
objMobAnimWalkE=*int,int* :: Set start and end sprite frames for walk animation (east).
objMobAnimWalkW=*int,int* :: Set start and end sprite frames for walk animation (west).
objMobAnimDeath=*int,int* :: Set start and end sprite frames for death animation.

Object Commands (Effect Type) (home)

These commands deal with objects of the effect type only (explosions, fire, poison gas, etc.). Use the collision script (explained above) to affect other objects, like fire damage, etc.

objEffLife=*int* :: Set lifespan of effect object in frames (60 FPS);
a life of 0 is infinite.
objEffRange=*int* :: Set range of effect object in tiles, collision script will affect nearby tiles;
objEffSpeed=*int* :: Set the speed of effect object's animation in frames.
objEffFadeOut=*int* :: If 1, effect object fades out when life runs out.
objEffNoParent=*int* :: If 1, effect object doesn't affect parent object.
objEffParentDef=*int* :: Set effect's parent object to the same parent as the default.

Modding Permissions (home)

You can sell your own mods for Warlordocracy on any site you want and pocket all the money, as long as you include a link to the Warlordocracy store page and DO NOT include the EXE files in your download.

Itch.io is a good place to sell mods. Let me know if you make one (you don't have to).