BRIGAND SCRIPTING LANGUAGE (BSL)

Laughing Coyote Software
Last Update: 5 June 2025

Table of Contents (Basics)

1) Game Scale
2) Editor Modes
3) New Story Files
4) Saves Folder
5) Game Assets
6) Objects Folder
7) Images Folder
8) Music Folder
9) Sounds Folder
10) Scripts Folder
11) Types of Scripts
12) System Scripts
13) Object Scripts
14) Default Skill Effects
15) Script Variables

Table of Contents (Commands)

16) Basic Commands
17) Constant Commands
18) Ability Commands
19) Status Effect Commands
20) Window Commands
21) Anim Commands
22) Map Commands
23) Temp Commands
24) Boolean Conditionals
25) Pick Commands
26) Object Commands
27) Object Commands (Sentient)
28) Object Commands (Sentient AI)
29) Object Commands (Obtainable)
30) System Sound Effects
31) Additional Notes
32) Modding Permissions

Game Scale (home)

This game engine uses the following scale: 33 units are about 1 meter, and 33 ticks are about 1 second (meaning the frame rate is about 33 frames per second). The game speed can be increased in the Settings menu.

The maximum jump height is 60 units high with maximum Agility skill, so remember this when designing maps.

Editor Modes (home)

There are two world editor modes: GUI and Command Prompt. The GUI editor can be accessed by pressing F1. It will bring you to an editor menu where you can edit the game properties, terrain, objects, scripts, etc. from an overhead view.

The Command Prompt is accessed by pressing F2. It will allow you to move around freely and select objects more carefully. Just click the mouse for the Command Prompt to appear. Target an object before clicking to edit that object's properties. You can use any of the commands listed below (except the Boolean "if" commands, which can only be used in scripts).

New Story Files (home)

You can save your new story file with the "saveStory" command, but be sure you set the story's name with the "storyName" command first (or just change it in the GUI editor). A new assets folder will be created, along with the story file. You can put assets unique to your new story in these new files, otherwise the game will use the default assets folder.

Saves Folder (home)

The saves folder contains all the saved games made by the player, which should be in GAM format. There can be up to 6 saved game files that will appear in the loading menu.

Game Assets (home)

The Assets folder should contain a "Default" sub-directory for all default resources. Never alter the contents of the default Assets. There should be a separate folder for each story file with the same name. You can make a copy of an asset to put in the story's Assets folder if you want to change something, such as dialogue. The story's Assets folder takes priority, but if the resources aren't found there, the game will look in the "Default" Assets folder.

The sub-directories under the Assets folders are:

Images: images, including textures, HUD, etc. divided into different folders.
Objects: models, divided into different folders along with their scripts.
Scripts: system scripts (not the object-specific scripts).
Sounds: sounds in WAV format (includes a Voices sub-directory, must be mono).
Music: soundtrack in MP3 format.

Assets: Objects Folder (home)

Each imported model used in the game will have a sub-directory inside the objects folder. The name of the sub-directory is the object's TAG. The sub-directory should contain the model file(s), which can be in the following formats: .X, .3DS, or .DBO. The latter is specialized for the engine and takes up less system memory and disk space.

Most objects will only have one model file, unless it is a sentient object. Sentient objects should contain 5 separate model files for each animation. The file names should start with the object's TAG (the name of the folder), followed by the type of animation. For example, the walking animation for a thug would be thugWalk.x or thugWalk.dbo.

TAG.dbo: This is the basic model file. For sentient objects, this should contain idle animation such as breathing, but it doesn't have to.

TAGWalk.dbo: For sentient objects, this is the animation played when the object moves.

TAGHurt.dbo: For sentient objects, this is the animation played when the object is damaged.

TAGAttack.dbo: For sentient objects, this is the animation played when the object attacks another object. This can be either a melee attack or a ranged attack animation, depending on the variable set in the "unarmed" command.

TAGFall.dbo: For sentient objects, this is the animation played when the object dies. The object animation will freeze at the final frame to appear dead.

Assets: Images Folder (home)

The images folder contains all the game images, divided into 8 sub-directories, as well as a blank image that is used when the specified image does not exist. The engine recognizes 3 image formats: BMP, PNG, and DDS.

If an object uses a BMP image as a texture, all black pixels will appear transparent in the game, useful for plants and tree canopies.

The following are the sub-directories that should be in the image folder.

Anims: Contains all the 2D sprite animations such as explosions. Each sprite animation image can be any size you want, but it will be divided into 16 frames, 4x4. To learn more about how sprite animations work, see the Anim Commands section below. These should all be BMP format so that black pixels will appear transparent in the game. There are a few required anim images: blood1, blood2, blood3, blood4, hit, dirt, bubbles, splashSmall, splashBig.

Faces: Contains all face images for sentient objects. These faces are displayed when showing the statistics screen, and can also be displayed when using a window command. All faces should be 60x78 pixels.

HUD: Contains all HUD images, as well as the cursor image. There should be 116 images total in this folder, which you can browse by looking through it. If you want to replace one of the images, make sure it has the same dimensions as the original. These should all be BMP format so that black pixels will appear transparent in the game. All images in this folder are required for the game to work.

Items: Contains all the item icons. These are displayed in the player's inventory, and also when examining an object. All object icons should be 64x64 pixels. These should all be BMP format so that black pixels will appear transparent in the game.

Menus: Contains all background images for menus or special scenes. All menu backgrounds should take up the entire screen, meaning they should be 1024x768 pixels. There are 4 required menu images: Title.bmp, Gameover.bmp, Info.bmp, and Credits.bmp (which should actually be taller than 768 pixels so that it can scroll).

Skies: Contains all the skyboxes, divided into 6 panels. The file names should start with the name of the sky, followed by a letter from A to F. Skies can be any dimensions you want.

Terrain: Contains all the terrain images used for the ground matrix and the cliff matrix. Though they can be any dimensions you want, the ground terrain images are divided into 16 tiles, 4x4, and the cliff terrain images are divided into 4 tiles, 2x2. The planes at the corner of the map will be textured with single textures of the same name from the textures folder (below).

Textures: Contains all textures used by the basic game objects and the complex objects found in the objects folder. By default, an object will use the texture with the same name as the object TAG, but it can be changed with the "skin" command. Remember, if an object uses a BMP texture, all black pixels will appear transparent.

Assets: Music Folder (home)

The music folder contains all the game music, which should be in MP3 format. You can have as many music tracks as you want, played using the **mapMusic** command. However, there are also 4 special music tracks that should have the following names: Intro.mp3 (played when the game launches into the main menu), Gameover.mp3 (played on the game over screen), Credits.mp3 (played when the credits are scrolling), and Victory.mp3 (played once all the way through and cannot be stopped by an additional **mapMusic** command until it ends, so it should be short).

Assets: Sounds Folder (home)

The sounds folder contains all sound effects for the game, which should be in WAV format. You an also keep sub-directories in the sounds folder for character voices, etc.

There are many required sound files of the following names: select1, select2, fail, money, obtain, drop, bonus, slip, hit, thump1, thump2, splash1, splash2, swim, step1 – step4, dam3 – dam7 (damage types 1 and 2 have no sound effects).

You can include footstep sounds for different object materials as well. Just name the file step + material + 1 and 2 (two footstep sounds). For example, if you want footsteps on metal objects, the file names would be "stepMetal1" and "stepMetal2". The same is true for bullet impact sounds. Just name the file hit + material, for example "hitMetal".

Assets: Scripts Folder (home)

The scripts folder contains all the system scripts, which are in the BSL format. They can all be opened and edited with notepad, but be sure to turn off the "Word Wrap" option. For more information on scripts, see below.

Types of Scripts (home)

The commands in this document are for use in two places: the in-game Advanced Editor (F2) and also object scripts (*.bsl files). The script files can be found in the scripts folder in the game directory (system scripts), and also in the individual object folders (object scripts). You can read more about the specific object scripts below.

System Scripts (home)

System Scripts are found in the scripts folder, and they can be called from any other scripts at any time. There are two system scripts that are required to be in the scripts folder:

Globals.bsl is called when the game launches, and should include game constant declarations such as skills, abilities, effects, objective descriptions, etc.

PlayerChecks.bsl is called once every 3 seconds to run checks for the player object such as hunger and passive ability checks. Keep this script as lean as possible, because if it is too dense, it will lag the game every 3 seconds.

Travel.bsl is called whenever the player travels to another map by way of the map edge. It can be used to pass the time, etc.

Map Scripts are also important. Each map also has a map script, which is also called every 3 seconds. Set the map script with the "mapScript" command. Map scripts can contain things such as day/night cycles, special AI for patrols, and constant 2D animations for campfires, etc. It is best to have a separate script for day/night cycles that is simply called from the map script, so it is consistent on every map.

Object Scripts (home)

Object scripts are found in each folder inside the objects folder, the same place as the model files. Each object can have 16 special scripts attached to it, but none of them are required. The name of the script should start with the object's TAG, the same name as the folder and model file, followed by the type of script. For example, the file name for the script run when a thug dies would be thugDie.bsl.

The following is a list of the 16 optional object scripts:

TAGInit.bsl :: Script run when object is loaded (used for default attributes).
TAGCol.bsl :: Script run when object collides with another object.
TAGUse.bsl :: Script run when object is used (before damage/effects).
TAGUse2.bsl :: Script run when object is used (after damage/effects).
TAGDam.bsl :: Script run when object is damaged.
TAGDie.bsl :: Script run when object is killed or destroyed.
TAGHeal.bsl :: Script run when object is healed.
TAGTim.bsl :: Script run when object's timer is set and runs out.
TAGHit.bsl :: Script run when object attack successfully hits.
TAGAtt.bsl :: Script run when object attacks (sentients only).
TAGWin.bsl :: Script run when object destroys/kills another object (sentients only).
TAGMov.bsl :: Script run when object moves (sentients only).
TAGAI.bsl :: Script run every 1.5 seconds (sentients only).
TAGGet.bsl :: Script run when object is obtained (obtainables only).
TAGDrop.bsl :: Script run when object is dropped (obtainables only).
TAGEmp.bsl :: Script run when object is out of charges/meta (obtainables only).
TAGRel.bsl :: Script run when object is reloaded (obtainables only).

Default Skill Effects (home)

Besides effects dealt with in scripts and object govSkill properties, some skills also have default effects, listed here. Skills 7 to 11 have no default effects for sentient objects, and are mostly made useful in ability scripts or as governing skill for weapons ("govSkill" command in the Obtainable Objects Commands section).

Skill 0: Maximum health.
Skill 1: Maximum meta.
Skill 2: Jump height, move speed, swim speed.
Skill 3: Melee damage, throw distance, swim speed.
Skill 4: Lessens the noise of sentient's footsteps.
Skill 5: Hold breath underwater for longer periods of time.
Skill 6: Improve the reaction of non-hostiles (player only).
Skill 12: Increases maximum attempts for minigame 1 (Minesweeper).
Skill 13: Increases maximum attempts for minigame 2 (Hangman).

Script Variables (home)

Script variables range from 1 to 500, and they are kept track of in a separate TXT file that is usually found in the story's root assets folder (for modding references). They can be set to positive or negative integers, or 0 (default). The variable commands can be found below under the Basic Commands section.

Variables are used for things as basic as the day/night cycle (variable #2 in DEFAULT assets and most stories, #8 in the original OAXACA story), or as specific as quest dialogue.

To set a variable to a given value, use the "var" command (listed in next section), and to increase or decrease a variable, use the "varUp" command, negative 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 money or faction respect. For example, if you want to set variable #20 to the player's money, use the command "var=20,money".

You can set variables (#1-500) to the following in-game values:

money :: set given variable to player's money.
points :: set given variable to player's skill points.
react1 :: set given variable to faction #1's respect.
react2 :: set given variable to faction #2's respect.
react3 :: set given variable to faction #3's respect.
react4 :: set given variable to faction #4's respect.
react5 :: set given variable to faction #5's respect.
skill0 (to 13) :: set given variable to current object's skill (ID 0 to 13).
resist0 (to 6) :: set given variable to current object's resistance (ID 0 to 6).

Basic Commands (home)

Basic commands that deal with saving game, script variables, displaying messages, etc.

newStory :: delete all maps in the current story and start over with a new name.
saveStory=*string$* :: save current game as the initial setup for a new story of given name.
saveStoryTxt=*string$* :: saves story in plain text so it can be edited with notepad.
saveGame=*string$* :: save current game as character name in the *saves* folder;
if you set a string, it will rename the character and save file;
setting the string to "Autosave" will retain the character's name and make a backup.
saveGameTxt=*string$* :: saves game in plain text so it can be edited with notepad.
savePlayer=*string$* :: save player as a *.CHR file (if no string, save as player name);
can set to “response” to save as player’s response in input window.
loadGame=*string$* :: load a *.gam file from the *saves* folder (no file extension).
loadStory=*string,int* :: load a *.gam file from the *stories* folder (no file extension);
set optional final parameter to 1 to import player, 2 to import player and variables.
loadPlayer=*string* :: load *.CHR file and overwrite current player (can set to “response”).
storyName=*string$* :: change the name of the story (*.gam file and assets folder);
all assets and scripts will now be in the 'DEFAULT' folder if no story folder exists.
react=*string$,int* :: set reaction percent of given faction (0 to 99), use "all" for all factions;
set 2nd parameter to "v" followed by 3 digits to set reaction to variable (v001-v500);
reactUp=*string$*,*int* :: increase reaction percent of given faction (can be negative).
var=*ID,int* :: set variable of given ID (1 to 500) to given integer (see variable notes above).
varUp=*ID,int* :: increase variable of given ID by given integer (can be negative).
varRand=*ID,int* :: set variable of given ID to a random number from 1 to given integer.
resetVars=*ID,ID* :: reset all 200 variables to 0;
if optional two parameters are set, only reset variables of ID in given range.
resetStats=*tag$* :: resets stats of all objects on map with given tag to default init script;
leave the parameter blank to reset every object.
msg=*string$* :: print a one-line message to the screen, only if player initiated the script.
msgAll=*string$* :: prints a one-line message regardless of the source object.
points=*int* :: give a certain amount of skill points to the player.
moneyUp=*int* :: increase player's money (can be negative);
set parameter to "v" followed by 3 digits to set money to variable (v001-v500).
headShotBonus=*int* :: set headshot multiplier (default 0=x2).
chapter=*int* :: set the current chapter (changes the loading screen).
callScript=*string$* :: call a script from the main scripts folder (no file extension).
callMapScript :: call script associated with current map (normally called every 3 seconds).
newObjective=*ID* :: add objective of given ID to objectives list.
eraseObjective=*ID* :: erase objective of given ID from objectives list.
eraseAllObjectives :: erase all objectives (before finale, for example).
compass=*ID* :: set to 1 to turn on the compass in the HUD.
stopAllSound=*filename$* :: stop all sounds currently playing with given file name.
stopAllAnim=*filename$* :: stop all 2d anims with given file name.
stopAllTargets :: stop all sentient object targets (stop all combat on map).
deleteParty :: delete all party members except the main player.
regroupParty :: instantly regroup all party members (if no collision).
camShake :: shake the camera for given number of ticks (0 to 60).
goto=*string$* :: go to given label in same script file (line that ends with a colon).
steam=*string$* :: unlock a Steam achievement with given API name.
timePass=*int* :: pass given number of ticks (each tick is 3 seconds).
playCredits :: scroll credits and reset the game.
gameover :: show gameover screen and reset the game.
reset :: reset the current script.
exit :: exit the current script.

Constant Commands (home)

These commands set the labels of the game's damage types, factions, skills, etc. They should never be altered during actual gameplay, but they can be if you want. Normally, these commands will only be found in the Globals.bsl system script in the scripts folder.

factionName=*ID,string$* :: rename faction of given ID (0 to 6) to given string.
damageName=*ID,string$* :: rename damage type of given ID (0 to 6) to given string;
set to "none" for no name.
skillName=*ID,string$* :: rename skill of given ID (0 to 13) to given string (see skill chart below);
set to "none" for no name (if skills 7 to 13 are set to none, disable second skill tab).
skillDesc=*ID,string$* :: change the skill descriptions to given string (visible in upgrade window).
healthName=*string$* :: set name of sentients' health stat to string.
metaName=*string$* :: set name of sentients' meta stat to string.
moneyName=*string$* :: set name of game currency to string.
pointsName=*string$* :: set name of skill points to string.
statusName=*string$* :: set the caption on the status window to string.
equipName=*string$* :: set the caption on the equip window to string.
effectsName=*string$* :: set the caption on the status effects window to string.
skillsName=*string$* :: set the caption on the skills window to string.
resistName=*string$* :: set the caption on the resist window to string.
upgradeName=*string$* :: set the caption on the skill upgrade window to string.
abilitiesName=*string$* :: set the caption on the abilities window to string.
partyName=*string$* :: set the caption on the party member window to string.
chapterName=*string$* :: set the caption for chapter number to string.
infoName=*string$* :: set the caption for the basic character info window to string.
respectName=*string$* :: set the caption for respect window to string.
nullName=*string$* :: set the caption for null strings to string.
minigame1Name=*string$* :: set the caption on the first minigame (minesweeper).
minigame2Name=*string$* :: set the caption on the second minigame (hangman).
minigame3Name=*string$* :: set the caption on the third minigame (Chinese dice).
objective=*ID,string$* :: set objective titles (1 to 99) to given string.
objectiveDesc=*ID,string$* :: set objective descriptions (1 to 99) to given string.
tip=*ID,string$* :: set the loading screen messages (1 to 99) to given string;
tips 1-10 are for chapter 1, 11-20 for chapter 2, etc.
renameAll=*string$,string$* :: rename all objects in game of given name to second given name.

Ability Commands (home)

These commands alter the properties of every ability (ID from 0 to 99). They should never be altered during actual gameplay, but they can be if you want. Normally, these commands will only be found in the Globals.bsl system script in the script folder.

abilityName=*ID,string$* :: change given ability ID to given name (or "none" to disable).
abilityDesc=*ID,string$* :: change given ability's description to given string.
abilityIcon=*ID,string$* :: set ability icon to alternate pic in the "hud" folder (no extension).
abilityScript=*ID,filename$* :: change script of ability to file name (no extension).
abilityReq=*ID,string$* :: change required skill of ability to skill name.
abilityCost=*ID,int* :: change meta cost of ability to given amount (also required skill level).
abilityRange=*ID,int* :: change the range of ability to given number of units.
abilityTarget=*ID,int* :: targeting type when using ability ("self", "other", "option", or "none").

Status Effect Commands (home)

These commands alter the properties of every status effect (ID from 0 to 6). They should never be altered during actual gameplay. They should never be altered during actual gameplay, but they can be if you want. Normally, these commands will only be found in the Globals.bsl system script in the script folder.

effectName=*ID,string$* :: rename effect of given ID (0 to 6) to given string.
effectDesc=*ID,string$* :: change description of effect, displayed when first inflicted.
effectConfuse=*ID,int* :: does the effect blur vision (1=yes)?
effectGhost=*ID,int* :: does the effect conceal you (1=yes)?
effectInfra=*ID,int* :: does effect cause infra vision (1=yes, 2=see through walls)?
effectCure=*ID,float* :: change amount of effect cured every 3 seconds;
effect curing can be negative, decimal, or a negative decimal.
effectMax=*ID,int* :: change maximum amount of effect.
effectDrain=*ID,int* :: change amount of meta drained with effect every 3 seconds.
effectDam=*ID,int* :: change damage of given effect every 3 seconds.
effectDType=*ID,string$* :: change damage type of above damage to damage name.
effectPen=*ID,string$* :: change skill name to which the effect gives a penalty;
"none" will have no penalties, "all" will apply to all skills except the first two.

Window Commands (home)

These commands will open various menus. To set the choice labels in the option window, use the "choice" command first before you use the window command. After using the "window" command or one of its variants, use the "if_response" command to test which option the player picked.

scene=*string$* :: display background image (must call a window afterwards for visible effect).
closeUp=*int* :: camera close up on currently-selected object (must call window afterwards).
if optional parameter is 1, the close up is instant (no camera movement).
window=*string$*,*string$* :: call a 5-option window with given title and message respectively;
use "/n" in the window message to substitute the current player's name;
use "/m" in the window message to substitute the player's money;
use "/r" in the window message to automatically start a new line;
use "/v" in the window message, followed by 3 digits to substitute variable value;
the player's response (1 to 6) can be tested with the "if_choice" command.
altWin=*string$*,*string$* :: same as *window*, but uses "windowB.png" as the skin.
editWin=*string$*,*string$* :: same as *window*, but will only appear if called using the world editor.
silentWin=*string$,string$* :: same as *window*, but there will be no window sounds.
leftWin=*string$,string$* :: same as *window*, but always positioned on left side, even in a scene.
inputWin=*string$*,*string$* :: call a text input window, with given title and message respectively;
player's text response can be tested with the "if_choice" command.
choice=*int,string$* :: set one of the six options (1 to 6) in window to given message string;
use this command immediately before using the window command;
choices are reset to blank after the next window command is used;
choices ending in "(cancel)", "(continue)", "(done)", or "(back)" get chosen with ESC.
choicePic=*int,string$* :: set picture for each of the 6 choices, displayed in bottom-right of window.
pauseGame :: pause the game until player presses a key (used right using a background scene).
upgrade :: call the player upgrade window (use current object's skills as upgrade limits).
displayStats :: display (most) statistics of the currently-selected sentient object.
findObj=*string$* :: find all objects of given name on all maps (list map and position).

Anim Commands (home)

These commands play 2D sprite animations at a point in 3D space (explosions, etc.). To have a permanent 2D animation playing, set the animLife to 97 in the map script. This is because anims consist of 16 frames, so it will loop 6 times till the map script is called again.

makeAnim=*filename$* :: make a new anim with the given picture file;
    the anim will appear at the currently-selected object's location;
    if no object is selected, the anim will appear at previously-selected anim.
clearAnim :: unselect the current anim so any new anims have no parent anim.
animScript=*int* :: set the script run when an object collides with the anim.
animLife=*int* :: set the life in frames of most recently-created animation (16=default);
    for permanent anims, set parameter to "perm" in map script (perm=97).
animDam=*int,string$* :: set damage of anim to given amount and damage type.
animEffect=*int,string$* :: set status effect inflicted by anim to given amount and effect name.
animRadius=*int* :: set radius of anim to given amount.
animGlue :: permanently glue anim to its parent object;
    if optional 2nd parameter is above zero, anim will be glued that far in front of object.
animTarget :: set anim's destination to its parent object's target coordinants.
animHoming :: set anim's destination to current object and automatically follow it.
animSpeed=*int* :: set anim's speed when moving towards a target.
animTemp :: force anim to dissipitate as soon as it hits a target.
animAngleXFix=*int* :: fix anim's x axis to given angle (instead of facing the camera).
animAngleYFix=*int* :: fix anim's y axis to given angle (instead of facing the camera).
animX=*int* :: move anim to given position on the X axis.
animY=*int* :: move anim to given position on the Y axis.
animZ=*int* :: move anim to given position on the Z axis.
animXMove=*int* :: move anim given units along the X axis, positive or negative.
animYMove=*int* :: move anim given units along the Y axis.
animZMove=*int* :: move anim given units along the Z axis.
animXDest=*int* :: set anim's destination to given position along the X axis.
animYDest=*int* :: set anim's destination to given position along the Y axis.
animZDest=*int* :: set anim's destination to given position along the Z axis.
animGround :: move anim down to the ground instantly (for fire, etc.).

Map Commands (home)

These commands deal with map variables. They can be changed in the map scripts for day/night cycles, etc. A map must have a name if it is to be saved in the GAM file. You can have a max of 50 maps.

mapName=*string$* :: set the currently-loaded map's name to given string.
mapMusic=*filename$* :: set music played on current map to file name without file extension;
set to "none" to turn off music.
mapAmb=*filename$* :: set looped sound effect played on map to file name without extension;
set to "none" to turn off ambiance.
mapAmbIn=*filename$* :: set looped sound effect played on map while player is indoors;
set to "none" to turn off ambiance.
mapScript=*filename$* :: set the script in the *scripts* folder that runs every 3 seconds on map.
mapShading=*int* :: set natural shading of all objects on map (0 to 100).
mapSky=*tag$* :: set sky tag on current map (file name without last letter or extension);
sky images have 6 sides, which are labeled A through F (last letter of file name).
mapWater=*filename$* :: set water texture on current map to file name without file extension.
mapGround=*filename$* :: set ground texture for current map to file name without file extension;
all terrain images are located in the images\terrain folder and must be PNG;
also include a single-tile PNG image of the same name in the textures folder.
mapCliff=*filename$* :: set cliff texture for current map to file name without file extension;
all terrain images are located in the images\terrain folder and must be PNG;
also include a single-tile PNG image of the same name in the textures folder.
fogDist=*int,int* :: number of units of visibility in fog (300 to 10000);
if the optional second parameter is set to 1, the change will be instant, not smooth.
fogRed=*int,int* :: set the amount of red in the fog (1 to 255);
if the optional second parameter is set to 1, the change will be instant, not smooth.
fogGreen=*int,int* :: set the amount of green in the fog (1 to 255);
if the optional second parameter is set to 1, the change will be instant, not smooth.
fogBlue=*int,int* :: set the amount of blue in the fog (1 to 255);
if the optional second parameter is set to 1, the change will be instant, not smooth.
rain=*int* :: set number of raindrops falling on screen (0 to 100).
snow=*int* :: set number of snowflakes falling on screen (0 to 100).
waterFlowX=*int* :: flow of water on X axis (0 to 50, can be negative).
waterFlowZ=*int* :: flow of water on Z axis (0 to 50, can be negative).
waterLvl=*int,int* :: set water level on current map.
if optional 2nd parameter is set to 1, the water level change is instant.
elev=*int* :: set ground elevation at player's location to integer.
elev5x5=*int* :: set 5x5 ground elevation centered at player's location to integer.
elevSmooth=*int* :: set 5x5 ground elevation with automatically-smooth edges.
elevAll=*int* :: fill entire map with given ground elevation.
elevCliff=*int* :: set cliff elevation (impossible to climb) at player's location to integer.
elevCliff5x5=*int* :: set 5x5 cliff elevation centered at player's location to integer.
elevCliffSmooth=*int* :: set 5x5 cliff elevation with automatically-smooth edges.
elevCliffAll=*int* :: fill entire map with given cliff elevation.
groundTile=*int* :: set ground tile at player's location from 1 to 16.
ground terrain images are divided into a 4x4 grid (16 tiles total).
groundTile5x5=*int* :: set 5x5 ground tiles at player's location from 1 to 16.
groundTileSmooth :: set 5x5 ground tiles with smooth edges;
uses tiles 5, 6, 7, 9, 10, 11, 13, 14, and 15.
groundTileAll=*int* :: set all ground tiles on map from 1 to 16.
groundTileRand :: randomize all the map's ground tiles.

cliffTile=*int* :: set cliff tile at player's location from 1 to 4;
cliff terrain images are divided into a 2x2 grid (4 tiles total).
cliffTile5x5=*int* : set 5x5 cliff tiles at player's location from 1 to 16.
cliffTileAll=*int* :: set all cliff tiles on map from 1 to 4.
cliffTileRand :: randomize all the map's cliff tiles.
linkNorth=*ID* :: set map's north link to map of given ID (1 to 30).
linkSouth=*ID* :: set map's south link to map of given ID (1 to 30).
linkEast=*ID* :: set map's east link to map of given ID (1 to 30).
linkWest=*ID* :: set map's west link to map of given ID (1 to 30).
make=*tag$* :: make object with given tag (name of folder) at current object's position;
box :: create a 100x100x100 unit box in front of the player;
cone :: create a 100x100x100 unit cone.
tube :: create a 100x100x100 unit cylinder.
ball :: create a 100x100x100 unit sphere.
plane :: create a flat 100x100 unit surface.
destroyAll=*str$* :: destroy all objects of given name.
deleteAll=*str$* :: delete all objects of given name.
if optional 2nd parameter is set to 1, exclude dead/destroyed objects.
deleteAllTag=*str$* :: delete all objects of given tag.
if optional 2nd parameter is set to 1, exclude dead/destroyed objects.
deleteCorpses :: delete all corpses from map.
factionChange=*string$,string$* :: change faction of all non-sentient objects on map;
first string is initial faction, second string is faction to change to.
deleteMap :: completely erase all map properties and objects on current map.
copyMap=*ID* :: erase current map and replace with map from different story file.
copyMapFrom=*filename$,ID* :: erase map and replace with map from different story file;
if map script is in story assets folder, copy it to current story's assets folder.

Temp Commands (home)

These commands are not saved with the game, and reset every time player moves to a new map. They should be called in the map script, which is automatically run when the game or map is loaded.

waterGhost=*int* :: set to 1 to make water plane transparent.
gravityMod=*int* :: set gravity to percentage (1 to 1000, default 100, set in map script).
deathScript=*str$* :: set script to run when player dies ("none" for standard gameover).
disableAI=*int* :: set to 1 to temporarily disable AI (does not reset on map load).
disableRegroup=*int* : set to 1 to disable auto team regroup.

Boolean Conditionals (home)

These commands begin a conditional branch for every command up to "endif" like other programming languages. In many commands, the equals sign may be replaced with either ">" or "<" to represent greater or less than.

if_storyName=*string$* :: test the name of the initial story file.
if_chapter=*int* :: test the current chapter.
if_rain=*int* :: test amount of rain on screen (generally used with ">" or "<", not "=");
use negative values to test for snow.
if_outside :: test whether current object is outside (no roof above).
if_map=*string$* :: test the name of the current map.
if_var=*int* :: test the value of a game variable, set with the *var* and *varup* commands.
if_money=*int* :: test the player's money.
if_points=*int* :: test the player's skill points.
if_load=*int* :: test how many items the currently-selected object is carrying.
if_type=*string$* :: test type of currently-selected object;
object types include normal, passable, usable, obtainable, and sentient.
if_tag=*string$* :: test the tag of the currently-selected object (name of folder).
set to "none" to test if there is no currently-selected object.
if_name=*string$* :: test the name of the currently-selected object;
set to "none" to test for objects with no names.
if_skin=*string$* :: test the file name of current object's texture (without file extension)
if_material=*string$* :: test the material of the current object.
if_faction=*string$* :: test the faction name of the current object.
if_react=*string$,int* :: test the reaction of given faction;
if the first parameter is omitted, use the faction of current object.
if_health=*int* :: test the health of current object;
you can also use if_health=full to test if health is at maximum (based on skill level).
if_healthPer=*int* :: test the percentage of health of current object.
if_meta=*int* :: test the meta of current object.
you can also use if_meta=full to test if meta is at maximum (based on skill level).
if_effect=*string$,int* :: test the given status effect level of the current object.
if_skill=*string$,int* :: test the given skill level of the current object.
if_skillBase=*string$,int* :: test the given base skill level (without temp bonuses).
if_know=*string$* :: test whether current object knows the given ability.
if_have=*string$* :: test if object name is in current object's inventory.
if_haveSpace :: test if object has an empty slot available in inventory.
if_delay=*int* :: test delay duration of object (weapon or sentient delay).
if_fall=*int* :: amount of time current object has spent falling (negative for jumping).
if_waiting :: test whether current object has been standing still for a second.
if_xPos=*int* :: test object's position on the x axis.
if_yPos=*int* :: test object's position on the y axis.
if_zPos=*int* :: test object's position on the z axis.
if_xAngle=*int* :: test object's angle around the x axis.
if_yAngle=*int* :: test object's angle around the Y axis.
if_zAngle=*int* :: test object's angle around the Z axis.
if_collision :: test whether current object is colliding with another object (1=yes).
if_isDead :: test whether current object is dead.
if_inWater :: test whether current object is in the water;
0=not in water, 1=touching water, 2=swimming in water, 3=underwater.
if_thrown :: test whether current object is being thrown at the moment.
if_noWeapon :: test whether current object has no weapon or ammunition.

if_player :: test whether current object is the player.
if_playing :: test whether current object is playing its animation.
if_sound :: test whether object is currently playing a sound effect.
if_rangeAttack :: test whether current object has a ranged attack (either sentient or obtainable).
if_atPosition=*int,int* :: test whether current object is within 25 units of given x and z coordinants.
if_nearPosition=*int,int* :: test whether current object is within 400 units of given coordinants.
if_objCount=*string$,int* :: test the amount of objects with given name on the current map.
does not count dead or destroyed objects.
if_objNear=*string$,int* :: test if object name is within given radius of current object;
switches selection to the object if one is found;
does not count dead or destroyed objects.
if_objNearLOS=*string$,int* :: same as *if_objNear* but also tests for line of sight.
if_objTagNear=*string$,int* :: test if object tag is within given radius of current object;
switches selection to the object if one is found;
does not count dead or destroyed objects.
if_sentientNear=*int* :: test if a sentient object is within given radius of current object;
does not count dead sentient objects, switches selection.
if_hostileNear=*int* :: test if a hostile sentient object is within given radius of current object;
switches selection to the object if one is found.
if_corpseNear=*int* :: test if a dead sentient object is within given radius of current object;
switches selection to the object if one is found.
if_itemNear=*int* :: test if a useful obtainable object is within given radius of current object;
ignores items that have targetType and ammoType both set to none;
ignores items owned by any faction, as well as "coin", "money", and "text" tags;
also tests for line of sight, switches selection to the object if one is found.
if_targetNear=*int* :: test whether current object's target is within given radius of current object;
returns false if current object has no target or if target is not in line of sight;
does NOT switch selection.
if_playerNear=*int,int* :: test whether player is within given radius of current object, and visible;
if second parameter is 1 then bypass line of sight check, does not switch selection.
if_partySize=*int* :: test how many teammates are in the party.
if_rand=*int* :: test with a random percent probability of passing (1% to 99%).
if_choice=*string$* :: use after a window or *inpWin* command to test for the player's input;
strings can be either 1, 2, 3, 4, 5, 6, or a response from an *inputWin* (see above;.
set to "savegames" to test if response is the same as any saved characters;
set to "none" to test for blank string.
if_minigame1=*int* :: test player in a game of minesweeper with a board of given size.
if_minigame2=*string$* :: test player in a game of hangman with the given word.
if_minigame3=*int* :: test player in a game of Chinese dice;
honesty parameter is 0 to 5 (with 5 being the most honest).
if_charLvl=*int* :: test sentient level (all skills, abilities, and skill points combined).
if_playerLvl :: test player level (all skills, abilities, and skill points combined).
if_combat :: test if any sentient objects are engaged in combat on the current map.
if_storyExists :: test if story exists in *stories* folder (no file extension).
if_importPlayer :: test if player has just been imported.
else :: begin the opposite conditional branch of previous boolean command.
endif :: end the current conditional branch.

Pick Commands (home)

These commands set the currently-selected object so you can then use object commands.

pick=*ID* :: select object of given ID number (max objects on map=5000).
pickName=*string$,int* :: pick a random object with the given name on the map;
if optional 2nd parameter is 1, do not pick player object;
never picks dead or destroyed objects.
pickTag=*string$* :: pick object of given tag (folder name), regardless of condition;
automatically picks the lowest object ID, unlike *pickName*.
pickParty=*int* :: pick member of the player's party (1 to 4, 1 being the main player).
pickDefault :: select the object that initiated the current script (selected by default).
pickTrigger :: select the other object that caused the current script to start;
in death and damage scripts, this will pick the attacker of the default object;
in the collision script, this will pick the object that collided with the default object;
in the use script, this will pick the object that is using the default object;
in the get script, this will pick the object that obtained the default object;
in the reload script, it will pick the ammo source (clips or other guns);
in the timer script, it will pick the player if called automatically (restore or teleport).
pickTarget :: select the current target of a weapon or a sentient object.
pickParent :: select the holder of the current object (the guy with the gun).
pickEquip :: select the current object's equipment (opposite of *pickparent*).
pickPlayer :: select the current player object.
pickMainPlayer :: select the main character created by the player.
pickLast :: select the object that was previously selected for scripts.

Object Commands (home)

You must select an object with a pick command or have one selected on the screen in the world editor for object commands to work.

type=*string$* :: set object's type to "normal", "passable", "usable", "obtainable", or "sentient".
name=*string$* :: set name of current object to given string;
set to "none" for no name or set to "response" to set to custom inputwin response.
desc=*string$* :: set object's description to given string;
set to "none" for no description.
skin=*filename$* :: set skin (texture) to file name without file extension;
DDS files take priority (fastest), then PNG, then BMP;
if image is a BMP, black colors will be transparent;
if set to “editor”, only show object in editor mode.
icon=*filename$* :: set inventory icon of object to file name in items folder without file extension;
if setting a sentient object's face icon, file will be in the faces folder.
material=*string$* :: set material of object to string (determines anim file when object is shot);
if material is "animal" various blood hit anims are shown when shot;
if material is "animal" it will appear red through fog and darkness with infravision;
if material is "wood" bullets can pass through it with decreased damage.
polyCollision=*int* :: set whether object uses polygon collision, which is slower (0=no, 1=yes);
if set to 1, only player can climb and there is automatic sliding on steep slopes;
if set to 2, any sentient can climb and there is no auto-sliding;
if set to 3, object functions like a ladder, which the player can climb vertically.
flying=*int* :: set whether object is unaffected by gravity (0=no, 1=yes, 2=no camera bob).
aquatic=*int* :: set whether object can breathe or be used underwater (0=no, 1=yes, 2=water only)
faction=*string$* :: set object's faction to one of the 6 faction names.
drawPriority=*int* :: set whether object has priority if a face overlaps with another (0=no, 1=yes)
losRadius=*int* :: set radius of line of sight check (for large projectiles like rockets);
for sentient objects, the check is for unarmed attacks;
if set to 0 (default), the radius is 0.2.
scale=*int* :: scale object to given percentage of default.
xScale=*int* :: scale object's X axis to given percentage.
yScale=*int* :: scale object's Y axis to given percentage.
zScale=*int* :: scale object's Z axis to given percentage.
scaleSkin=*int* :: scale texture's width and height to given percentage.
xScaleSkin=*int* :: scale texture's width to given percentage.
yScaleSkin=*int* :: scale texture's height to given percentage.
xDest=*int* :: set X destination for object to gradually move towards.
yDest=*int* :: set Y destination for object to gradually move towards.
zDest=*int* :: set Z destination for object to gradually move towards.
xDestChange=*int* :: set X destination for object, relative to current position.
yDestChange=*int* :: set Y destination for object, relative to current position.
zDestChange=*int* **::** set Z destination for object, relative to current position.
rotateTo=*int* :: set destination Y angle for object to gradually rotate to.
moveForward=*int* :: move object straight forward by given units.
xMove=*int* :: move object given units along the X axis, positive or negative.
yMove=*int* :: move object given units along the Y axis.
zMove=*int* :: move object given units along the Z axis.
xPos=*int* :: move object to given position on the X axis (map size=5000x5000).
yPos=*int* :: move object to given position on the Y axis.
zPos=*int* :: move object to given position on the Z axis.

resist=*string$,int* :: set object's resistance to given damage name to given percentage;
    set 1st parameter to "all" to set resistance to all 7 damage types;
    set 2nd parameter to "v" followed by 3 digits to set resistance to variable (v001-v500).
resistUp=*string$,int* :: add integer to object's damage resistance (can be negative).
resistBon=*string$,int* :: grant a temporary bonus to given resistance (effects don't stack).
groundPos :: move an object instantly to the ground.
teleport=*int* :: teleport object to given map ID (1 to 50);
    only works in object use or collision scripts.
xAngle=*int* :: rotate object to given angle around the X axis (loop at 360).
yAngle=*int* :: rotate object to given angle around the Y axis.
zAngle=*int* :: rotate object to given angle around the Z axis.
fixPivot :: set object's default rotation (0 degrees) to its current angle.
playObject :: play object's animation from start to finish.
loopObject :: loop object's animation repeatedly.
rewindObject :: reset object's animation frame to its original and stop animation.
setObjectFrame=*int* :: set current animation frame of object and stop animation.
curDelay :: set current delay of object, before it can be used again;
    not to be confused with *delay* command below (sets rate of fire).
damage=*int,string$* :: deal given amount of damage (the string denotes the type of damage).
crosshairKick=*int* :: increase the crosshair of the current object (or the holder of the object);
    the maximum crosshair size is 100.
heal=*int* :: restore given amount of target object's health or durability.
recharge=*int* :: restore given amount of target's meta (this reloads guns, too).
health=*int* :: set health to exact amount.
meta=*int* :: set meta to exact amount.
useScript :: automatically run the object's special "use" script.
playSound=*filename$,int* :: play sound at current object with given volume (1 to 500);
    if no object is selected, the sound will play at the most recent 2D animation;
    if file name is "fail", the system fail sound will be played instead of a 3D sound;
    if sound is located in the "voices" sub-folder, it won't play if object is underwater.
stopSound :: stop the current sound playing at current object, allowing for another one to play.
setTimer=*int* :: set object timer (to run "tim" script when countdown ends).
muzzle=*filename$* :: set file name of object's 2d use animation (without file extension);
    for sentient objects, this animation plays during unarmed attacks.
copy :: duplicate the current object at precisely the same coordinates.
destroy :: destroy the current object (if it's not invincible).
delete :: permanently delete the current object from the current map.
deusEx :: max out all skills, abilities, and resistances of current object.

Sentient Object Commands (home)

These commands are only for the "sentient" object type.

setPlayer :: set current player to selected sentient object.
headLimb=*ID* :: set limb number of object's head (rotates when looking);
also, if there is no head limb, object is immune to headshots.
gripLimb=*ID* :: set limb number that hold's object's current equipment.
hideLimb=*ID* :: hide the limb number given in the value.
showLimb=*ID* :: show the limb number given in the value.
collisionWidth=*int* :: set the width of object's collision box (default, min=25).
collisionHeight=*int* :: set the height of object's collision box (default, min=64).
corpseBack=*int* :: how far object's corpse is knocked back for collision purposes.
unarmed=*int* :: set whether object can perform unarmed attacks;
0=no, 1=melee, 2=ranged scripted, 3=ranged gun, 4=no ranged attack animation;
if unarmed is 3 (ranged gun), accuracy is determined by skill#1 (max meta).
natDType=*string$* :: set sentient's unarmed damage type to one of the 7 damage names.
camOverlay=*int* :: set camera overlay image if sentient object is player (0 to 3).
skill=*string$,int* :: set object's skill of the given label to given integer (0 to 50);
set 1st parameter to "all" to set all 14 skills;
set 2nd parameter to "v" followed by 3 digits to set skill to variable (v001-v500).
skillUp=*string$,int* :: add integer to object's skill (can be negative).
skillBon=*string$,int* :: grant a temporary bonus to given skill (effects don't stack).
ability=*string$,int* :: learn or forget the given ability (0=forget, 1=learn);
set the parameter to "none" to unlearn all abilities.
useAbility=*string$* :: force sentient to use given ability name.
effect=*string$,int* :: set given status effect to amount (max=50).
effectUp=*string$,int* :: add amount to given status effect (min=-50, max=50).
switchTo=*string$* :: switch sentient's equipment to given item name (if sentient has it);
set to "none" to switch to an empty inventory slot (if it exists).
give=*tag$* :: make object with given tag (folder) and give to current object;
to modify it before giving to the player, use the "giveToPlayer" command (below).
drop :: force object to drop its current equipment.
dropAll :: drop all the items in current object's inventory.
shoot=*int* :: force object to use its current equipment;
if optional parameter is set to 1, gunshot is an automatic hit, instant rotation.
task=*string$* :: play one loop of a sentient task ("idle", "hurt", "attack", "walk", or "fall").
throw :: throw object's current equipment.
jump :: force object to jump forward.
resetFall :: reset the object's fall distance (and the resulting damage).
equipAngleX :: change the default X angle of object's equipment.
equipAngleY :: change the default Y angle of object's equipment.
equipAngleZ :: change the default Z angle of object's equipment.

Sentient Object AI Commands (home)

These commands are only for the "sentient" object type, dealing specifically with behavior.

aiTarget=*ID* :: set current object's target to a specific object ID (or 0 for none).
aiTargetName=*string$,int* :: set current object's target to a nearby object with given name.
if optional parameter >0, only target objects within that range.
aiTargetPlayer :: set current object's target to the player (becomes hostile).
aiTargetParty :: set current object's target to a random party member.
aiTargetDefault :: set current object's target to whatever object initiated the script.
aiTargetLast :: set current object's target to previously-selected object.
aiFollow=*ID* :: make current object follow a specific object ID (or 0 for none).
aiFollowName=*string$* :: make current object follow object with given name.
aiFollowPlayer :: make current object follow the player.
aiFollowTarget :: make current object follow whatever object it is targeting.
aiFollowDefault :: make current object follow whatever object initiated the script.
aiFlee=*int* :: set the max health at which object flees away from battle.
aiGuard=*int* :: current object stands ground, won't move unless ordered by a script (1=yes);
set to 2 if you don't want the sentient to rotate either.
aiNoiseCheck=*int* :: current object follows nearby hostile noises (1=yes).
aiStopAdv=*int* :: current object stops advancing towards target at given distance.
aiJoinParty :: current object joins the player's party.
aiLeaveParty :: current object leaves the player's party.

Obtainable Object Commands (home)

These commands are only for objects of type "obtainable".

reqStrength=*int* :: set required strength skill to obtain object (skill 3).
noExtract=*int* :: set to 1 to disallow ammo extraction for the reloading of other weapons.
durability=*int* :: set maximum health of obtainable.
maxCharge=*int* :: set maximum uses (-1=infinite, 0=disposable, >0=rechargable).
range=*int* :: set maximum range of target when sentients use object;
set to 0 for melee weapons, -1 for ranged weapons with no crosshairs.
power=*int* :: set damage dealt by obtainable (negative for heal);
if range>0, damage is also affected by range, weapon condition, and headshots;
range affects damage from +3 to -3, condition from +2 to -2, and headshots are x2.
dType=*string$* :: set damage type of object to one of the 7 damage names.
accuracy=*int* :: set accuracy of obtainable (skill bonus).
delay=*int* :: set delay of obtainable every time it's used or equipped (20=default).
scatter=*int* :: set number of shots fired on a single use (1=default).
ammoType=*ID* :: set object's ammo ID (0=not ammo)
ammoReq=*ID* :: set the ID of ammo required to reload this object.
zoom=*int* :: set the zoom magnification when the examine key is pressed.
windUp=*int* :: set number of ticks it takes to wind up weapon before shooting.
govSkill=*string$* :: set skill that affects the object's accuracy to one of the 14 skill names.
targType=*string$* :: targeting type when using object ("self", "other", "option", or "none").
muzzleY :: correct Y position of object muzzle flash.
equipPosX=*int* :: correct X position of object when held by someone.
equipPosY=*int* :: correct Y position of object when held by someone.
equipPosZ=*int* :: correct Z position of object when held by someone.
giveToPlayer :: give current object to player;
with this command, you can make an object, modify it, then give it to the player.
giveToDefault :: give current object to the default object (that initiated the current script).

System Sound Effects (home)

These sounds are played by default, and should exist in the sounds folder. They can be replaced for mods in the mod's assets folder, just like any other resource.

select.wav	:: Played in menu selections.
select2.wav	:: Played in menu selections.
bonus.wav	:: Played when you earn points and other benefits.
money.wav	:: Played when you find money.
fail.wav	:: Played when a check is failed.
miss.wav	:: Played when you miss in melee or throw an item.
obtain.wav	:: Played when you pick up an item.
drop.wav	:: Played when you drop an item.
thump1.wav	:: Played when you land after falling.
thump2.wav	:: Played when you land after falling (alternative).
swim1.wav	:: Played when you are swimming.
swim2.wav	:: Played when you are swimming (alt).
splash.wav	:: Played when you are walking in water.
splashBig.wav	:: Played when you fall into water.
fallDie.wav	:: Played when you die from falling.
slip.wav	:: Played when you are sliding down cliffs.
hitWood.wav	:: Played when you shoot wooden objects (penetration).
hit(Material).wav	:: Played when you shoot other materials.
dam3.wav:	:: Played when you take damage type #3.
dam4.wav	:: Played when you take damage type #4.
dam5.wav	:: Played when you take damage type #5.
dam6.wav	:: Played when you take damage type #6.
dam7.wav	:: Played when you take damage type #7.
step1.wav	:: Played when you walk.
step2.wav	:: Played when you walk (alt).
step3.wav	:: Played when you walk (alt).
step4.wav	:: Played when you walk (alt).
stepCliff1.wav	:: Played when you walk on level cliff surface.
stepCliff2.wav	:: Played when you walk on level cliff surface (alt).
step(Material)1.wav	:: Played when you walk on surface of material.
step(Material)2.wav	:: Played when you walk on surface of material (alt).

Additional Notes (home)

-Maps must be named to be saved. Don't forget to **NAME YOUR MAP** or you will lose all your work when you quit.
-The "coin" or "money" object tag (folder name) will always run the "get" script and delete the object in place of actually moving the object to the player's inventory. This is for coins to give the player money and bypass the inventory slot check. The player will also automatically pick up "coin" or "money" objects on the ground with no owner instead of just targeting them.
-Faction 0 is always neutral, faction 6 is always the player's party.
-Guns can shoot through materials set to "wood", but the damage is reduced.
-Obtainables with 0 *durability* are indestructible.
-Obtainables with 0 *maxcharge* are labeled "disposable", -1 *maxcharge* is "infinite", but there is no difference in gameplay (you must handle that in the object scripts).
-Only objects with 0 *durability* and 0 or -1 *maxcharge* can be stacked (up to 10 in one slot).
-You can find tutorial videos in the Stuff folder (EXTRAS DLC).
-You can play the Mod Tutor story for an interactive tutorial (EXTRAS DLC).

Modding Permissions (home)

You can sell your own mods for Brigand on any site you want and pocket all the money, as long as you include a link to the Brigand 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).