EGOBOO

2.7.7

Development Guide

Egoboo - (Ee-go-boo) *is a OpenGL game, and requires a 3D graphics card to play. You can try it in software mode, but it's not supported. If you have problems, run either of the Setup files. Egoboo is freeware, open source, but it is not finished and is still being developed by a small team. So check the websites for updates, editors and etc..*

http://egoboo.sourceforge.net/

Original Game Idea: Aaron Bishop and Ben Bishop

Manual By: Zefz aka Johan Jansen

Contents

System Minimum Requirements

These are the minimum we know it works on, but it might very well work on less. Tune down some graphical effects if the frame rate is low.

Processor:	PentiumIII/Athlon with 2.0 GHz
Memory:	256 MB RAM (512 MB RAM Recommended)
Video:	Any graphical 128 MB graphics card supporting OpenGL graphics
Other:	For Hotseat, Egoboo requires gamepads or a connected mouse. Internet or LAN/UDP is required for multiplayer. (Egoboo 2.3.6 and later does not have networked multiplayer)

Introduction

Most data files for Egoboo are stored as .TXT files for easy modification, to encourage the growth and expansion of the game. If you are interested in making changes, see the HOW2EDIT.TXT file in the DEV bundle. The map files are .MPD files, and can be made with the included map editor. Yes, the map editor is bad, but it works. Well, it works sometimes. The image files are .BMP file, and can be made with the included paint program (nice for seamless tiles), or any commercial program. The model files are .MD2 files, and can be made with the included modeller. This program isn't mine, but it's what I used for everything. It'll grow on you. The sound files are .WAV files. I don't know of any free programs for doing full sound manipulation, but many sound cards come with programs that are good enough. Well they're good enough anyway. EgoMap is a map editor specially designed for Egoboo, you can download it from the Official Egoboo site. Whenever you have any questions about editing Egoboo, just go to the Egoboo forum (http://egoboo.informe.com/forum/).

Download the DEV package for all newest version Egoboo tools and unfinished material that you can work on!

http://sourceforge.net/project/showfiles.php?group_id=12221

Below there are different tutorials and information on how to edit and develop (Internet will be very handy if you plan to read some of these documents, as some contain links to various sites):

Objects - (Characters, items, monsters, etc.)

Scripts - (Monster AI, spawning points, passages, etc.)

Models - (Modelling, skinning and animation of objects)

Modules - (Dungeons, levels, cities containing lots of objects)

Making Objects

Objects in Egoboo modules are fairly simple, once you know a few little things about them. This topic will attempt to describe the basics of how this works to give the necessary background information.

Two major concepts

Modules basically contain two major parts related to objects: An "objects" directory which contains all the possible objects for that module, and a spawn.txt file, directing Egoboo where to put these objects.

Object slots:

The main thing to remember is that the "Objects" directory is used like a factory, to create objects in the module. Having just an object folder in the "objects" directory will do pretty much nothing unless it's also being spawned.

Each object directory (ex: sword.obj) contains several files, but the important one as far as modules are concerned is "data.txt". This file has a "Slot Number" as it's first entry. This is how the spawn file refers to any particular object. Each object must have a unique ID in the ranges of 36 to 255. (NOTE: I am not positive of the upper limit) The first 36 slots are reserved for importing of characters. Since Egoboo supports 4 Hero's (Players) at a time, each Hero can have 8 items. 0 would be player one, and 1 through 8 would be his/her items. 9 would then be player 2, and so on. If an object has a conflicting Slot Number, egoboo will exit immediately because it will not be able to tell them apart.

Whenever an object is manually copied into a different module, the "Slot Number" must be unique. The module editor EgoMap does this automatically if an object is imported from inside the editor.

Spawning:

The file "spawn.txt" is used to create objects. It use's the object's Slot Number to figure out which object to spawn, and makes an instance of it at the positions specified (in "spawn.txt"). Spawn.txt spawn.txt, located in the module.mod\gamedat folder, allows you to edit spawns for items, monsters, the hero and his\her items. It also allows you to give NPCs items. The spawn.txt file already has a basic bit of information included in its header.

Spawn.txt Basic Info This is a simple example of a spawn in a module. It will explain what the values and such mean.

```
// Name Slot Xpos Ypos Zpos Dir Mon Skn Pas Con Lvl Stt Gho Team
LUMPKIN 0: NONE 38 10.0 13.0 1.0 N 0 0 0 0 0 F F Evil
TORCH: NONE 40 0 0 0 R 0 0 0 0 0 F F NULL
CLAWS: NONE 42 0 0 0 L 0 0 0 0 0 F F NULL
```

Name - A name for the object. It seems it should be none for decorative items (ie: stools, tables, rugs, bookcases, etc), and NPCs may have a name associated with them (ie: in town.mod, the guards are named "a_Guard").

Slot - This is where the slot id number for an object goes. The slot ID number for items can be found in its data.txt. The object's slot IDs should start at 36, to leave room for the Heros to spawn. The hero's slots should be like so:

```
HERO 0: NONE 0 12.5 6.5 0.0 N 0 0 0 0 0 T F Good
HERO 1: NONE 1 0 0 0 L 0 0 0 0 0 F F Null
HERO 2: NONE 2 0 0 0 R 0 0 0 0 0 F F Null
HERO 3: NONE 3 0 0 0 I 0 0 0 0 0 F F Null
HERO 4: NONE 4 0 0 0 I 0 0 0 0 0 F F Null
HERO 5: NONE 5 0 0 0 I 0 0 0 0 0 F F Null
HERO 6: NONE 6 0 0 0 I 0 0 0 0 0 F F Null
HERO 7: NONE 7 0 0 0 I 0 0 0 0 0 F F Null
HERO 8: NONE 8 0 0 0 I 0 0 0 0 0 F F Null

HERO 9: NONE 9 11.5 6.5 0.0 N 0 0 0 0 0 T F Good
HERO 10: NONE 10 0 0 0 L 0 0 0 0 0 F F Null
HERO 11: NONE 11 0 0 0 R 0 0 0 0 0 F F Null
HERO 12: NONE 12 0 0 0 I 0 0 0 0 0 F F Null
HERO 13: NONE 13 0 0 0 I 0 0 0 0 0 F F Null
HERO 14: NONE 14 0 0 0 I 0 0 0 0 0 F F Null
HERO 15: NONE 15 0 0 0 I 0 0 0 0 0 F F Null
HERO 16: NONE 16 0 0 0 I 0 0 0 0 0 F F Null
HERO 17: NONE 17 0 0 0 I 0 0 0 0 0 F F Null

HERO 18: NONE 18 12.5 5.5 0.0 N 0 0 0 0 0 T F Good
HERO 19: NONE 19 0 0 0 L 0 0 0 0 0 F F Null
HERO 20: NONE 20 0 0 0 R 0 0 0 0 0 F F Null
HERO 21: NONE 21 0 0 0 I 0 0 0 0 0 F F Null
HERO 22: NONE 22 0 0 0 I 0 0 0 0 0 F F Null
HERO 23: NONE 23 0 0 0 I 0 0 0 0 0 F F Null
```

```
HERO 24: NONE 24 0 0 0 I 0 0 0 0 0 F F Null
HERO 25: NONE 25 0 0 0 I 0 0 0 0 0 F F Null
HERO 26: NONE 26 0 0 0 I 0 0 0 0 0 F F Null

HERO 27: NONE 27 11.5 5.5 0.0 N 0 0 0 0 0 T F Good
HERO 28: NONE 28 0 0 0 L 0 0 0 0 0 F F Null
HERO 29: NONE 29 0 0 0 R 0 0 0 0 0 F F Null
HERO 30: NONE 30 0 0 0 I 0 0 0 0 0 F F Null
HERO 31: NONE 31 0 0 0 I 0 0 0 0 0 F F Null
HERO 32: NONE 32 0 0 0 I 0 0 0 0 0 F F Null
HERO 33: NONE 33 0 0 0 I 0 0 0 0 0 F F Null
HERO 34: NONE 34 0 0 0 I 0 0 0 0 0 F F Null
HERO 35: NONE 35 0 0 0 I 0 0 0 0 0 F F Null
```

This ensures that the same hero won't spawn twice, or two seperate hero's items won't be the same, etc. EgoMap doesn't do this correctly, so you must manually edit the hero spawn slots (as of April 11 2005, it may have been fixed since).

Xpos + **Ypos** - Xpos and Ypos are the cordinates of the item's\NPCs\etc spawn. It seems heroes should have .5 after the cordinate (does it really have too?). You can find the cordinates by using EgoMap and Cartman.

Zpos - The start elevation for the item\NPC\hero\etc. It also allows you to put an item atop of another item too (ie, you want a book on a table, so you set the Zpos for the book at 0.5, and the table at 0.0, so the book will fall unto the table when the module loads)

Dir - This is where the item should spawn in\on an item\NPC\hero\etc. THe different settings are:

- **L** -The item will spawn in the npc's\hero's\etc left hand. (weapons, books, and shields are what usually is spawned here. It also is used for mounts with only one grip)
- **R** - The item will spawn in the npc's\hero's\etc right hand. (weapons, books, and shields are what usually is spawned here)
- **I** - The item will spawn in the NPC's\hero's\etc inventory. (usefull for keys) Mon: Gives an item\NPC\hero\etc extra money.

Skn - The skin that should be used for this level. It can be 0-3, or 4 if you want the NPC's\hero's (does it work with a hero?)\item\etc skin to be random. (note: to make braziers alight, set its skin to 20...)

Pas - The passage the item refers too. This can be used for braziers\switches\etc to open doors.

Con - Content setting for the character. It is used for armor chests.

Lvl - This defines what level an NPC will start at in this module. It does not work on players.

Stt - This will show the the status bar for an NPC\Hero if set to T (ie, in the adventurer module, the Mother Mim and Brom are set to 'T'), otherwise it will not show the status bar for the NPC\Hero if it is set to 'F'.

Gho - It has the same as the book of the Unseen, (as of now, April 11, 2005) inverting the Hero's\NPC\etc skin... It should make them invisible actually.

Team: The NPC\Hero\etc team. The first letter is used (so Evil can also be E, and Frog and be F, etc). On capital letters are used, so e and E are the same ('e' and 'E' and 'Evil' and 'Elf' and 'elf' are the same as far as spawn.txt is concerned). 'N' or Null is already set to be nuetral (it should be used on items, and NPCs (only if NPCs will not be a part of any team!).

MAKING MODELS

Overview

Egoboo uses the Quake 2 .MD2 model format for object models.

Character Models

Character (Humanoid) models must have a left and right grip. Grips are where weapons get attached to the character. Each grip is represented by 4 vertices. the grip vertices should be put into the character last! Just cut and paste from some other model and you'll be fine. Basically the grip vertices represent a matrix. 4 vertices = x rotation, y rotation, z rotation, and translation(position). It is kind of hard to visualize when you are looking at the vertices in the model, but you can kind of see it. The vertex at the hand is the translation, the vertex in the positive x direction is the x rotation, etc...

Weapon Models

Weapon models must use a SPAWNLAST vertex. It is an extra vertex that floats at the tip of the weapon. Attack particles get spawned at this point. the spawnlast vertex should be put into the model last!

Textures and Skinning

The latest version of Egoboo supports almost any texture format, Although we still use mostly the BMP file format and in some special cases the PNG (Which supports transparency).

Model Editors

GOLEM - Open GL Model Editor - Designed with Egoboo in mind. Designed to be cross-platform (Linux, Mac OS, Windows) Still in development.

Currently can Load and Save the Quake 2 model format that Egoboo uses. Will support model modes which disable unsupported features and add validation. (One mode will be Egoboo) Will support tag (grip) editing which will make editing models easier.

AZTEC - A 3d modeler which can be used to create egoboo models, not sure if it can be used for Egoboo yet

QUAKEME - Another modeler, you can download the full version here: http://www.edgefiles.com/dirs/965.html

MDL - Quake 2 Modeler - Origial modeler used to create egoboo models. Part of the extras download at http://egoboo.sourceforge.net

Q2MTutorial1

Q2MTutorial2

Q2MTutorial3

FRAME NAMES

The most important thing about creating a new model is to name your frames correctly. The name consists of a 2 letter action type, followed by a the 2 digit frame number, followed by any special codes. It looks something like KA03SP.

Action Types

DA - Dance (Typical standing)
DB - Dance (Bored)
DC - Dance (Bored)
DD - Dance (Bored)
UA - Unarmed Attack (Left)
UB - Unarmed Attack (Left)
UC - Unarmed Attack (Right)
UD - Unarmed Attack (Right)
TA - Thrust Attack (Left)
TB - Thrust Attack (Left)
TC - Thrust Attack (Right)
TD - Thrust Attack (Right)
CA - Chop Attack (Left)
CB - Chop Attack (Left)
CC - Chop Attack (Right)
CD - Chop Attack (Right)
SA - Slice Attack (Left)
SB - Slice Attack (Left)
SC - Slice Attack (Right)
SD - Slice Attack (Right)
BA - Bash Attack (Left)
BB - Bash Attack (Left)
BC - Bash Attack (Right)
BD - Bash Attack (Right)
LA - Longbow Attack (Left)
LB - Longbow Attack (Left)
LC - Longbow Attack (Right)
LD - Longbow Attack (Right)
XA - Crossbow Attack (Left)
XB - Crossbow Attack (Left)
XC - Crossbow Attack (Right)
XD - Crossbow Attack (Right)
FA - Flinged Attack (Left)
FB - Flinged Attack (Left)
FC - Flinged Attack (Right)
FD - Flinged Attack (Right)
PA - Parry or Block (Left)
PB - Parry or Block (Left)
PC - Parry or Block (Right)

PD - Parry or Block (Right)
EA - Evade
EB - Evade
RA - Roll
ZA - Zap Magic (Left)
ZB - Zap Magic (Left)
ZC - Zap Magic (Right)
ZD - Zap Magic (Right)
WA - Sneak
WB - Walk
WC - Run
WD - Push
JA - Jump
JB - Falling (End of Jump) (Dropped Item left)
JC - Falling [Dropped item right]
HA - Hit
HB - Hit
HC - Hit
HD - Hit
KA - Killed
KB - Killed
KC - Killed
KD - Killed
MA - Misc (Drop Left Item)
MB - Misc (Drop Right Item)
MC - Misc (Cheer/Slam Left)
MD - Misc (Show Off/Slam Right/Rise from ground)
ME - Misc (Grab Item Left)
MF - Misc (Grab Item Right)
MG - Misc (Open Chest)
MH - Misc (Sit)
MI - Misc (Ride)
MJ - Misc (Object Activated)
MK - Misc (Snoozing)
ML - Misc (Unlock)
MM - Misc (Held Left)
MN - Misc (Held Right)

SPECIAL CODES

```
AL - Spawn an attack particle on the left weapon
DL - Drop the item in the left grip
GL - Grab an item with the left grip
CL - Grab a character with the left grip
AR - Spawn an attack particle on the right weapon
DR - Drop the item in the right grip
GR - Grab an item with the right grip
CR - Grab a character with the right grip
I - Make the character invincible
S - Stop the character from from accelerating
F - Play a footfall sound
P - Poof the character
```

GRIPS

Grips are the last vertices placed in the model. They typically represent a character's (or creature's) left and right hand. Grips are also used for mounts, you put the grip where the character will ride... it can be thought of as the saddle for the mount. The scale, position and rotation of each set of grip vertices controls the scale, position and rotation of whatever other object the character is holding in that grip.

Grips have been notoriously difficult to work with. If the vertices are not in exactly the right order, the object you are holding appears stretched, squashed, distorted or worse.

You are allowed to use up to two grips per model. If you only need one, it will be considered the left grip in your data.txt file for the object.

The simplest way to add grips to your character is to cut and paste the grips from an existing model.

Make sure that you animate the grip vertices when you animate the rest of the model. You wouldn't want your character's hand to move, but whatever he's holding to remain still.

Here we have prepared two models. They are both included in the following archive:

· grip.zip (Requires internet connection to download)

grip.mdl is only one grip, for use with mounts and such. grips.mdl is a set of left and right grip vertices. You can extract the archive into your QME geometry folder, to easily add grips

to your models. Or, you can just open the grip models in your favourite model editor, select the grip, copy it and paste it into your new model. Again, it's important that you add the grips last to your model, due to the nature of how Egoboo reads model data.

AI Scripting

EgoScript

EgoScript is a fairly simple scripting language that is very much like Assembly language programming. It has a very limited amount of variables that can be used, and reused, for calling functions or checking IF statements. Despite it's limitations, it's still possible to make some very creative behaviour for both objects and monsters. To learn more about EgoScript functions, check the Text directory that came with Egoboo. It contains several text files about the subject, including a full list of functions available.

The most important of those text files, as far as EgoScript is concerned, would be the AiDocs. These are included with the DEV package (Downloaded under Tools on http://egoboo.sourceforge.net/).

Escape Sequences

The MESSAGE.TXT files use cryptic little escape sequences to display certain types of information. Each sequence consists of a % followed by a letter or number. Here's a nice little list...

- %n - The character's name (or class if name is unknown)
- %t - The target's name (or class if name is unknown)
- %o - The owner's name (or class if name is unknown)
- %c - The character's class
- %s - The target's class
- %0 - The target's skin 0 name (some_Leather_Armor)
- %1 - The target's skin 1 name (some_Chainmail)
- %2 - The target's skin 2 name (a_Breastplate)
- %2 - The target's skin 3 name (a_Hero_Costume) (Most powerful level)
- %d - The value of tmpdistance
- %x - The value of tmpx

- %y - The value of tmpy
- %D - The value of tmpdistance (Use for stat display of pets)
- %X - The value of tmpx (Use for stat display of pets)
- %Y - The value of tmpy (Use for stat display of pets)
- %a - The character's ammo (or ? if unknown)
- %k - The character's kurse state (kursed or unkursed)
- %p - The character's possessive (his, her, or its)
- %g - The target's possessive (his, her, or its)
- %m - The character's gender (male_, female_, or _)

Expansions

The data files for Egoboo started out simple. Then they got complex, and I updated them. More complexity, and more updates, until I had so many files that updating them all individually took too long. That's why there are Expansions. Expansions are those cryptic four letter IDSZ things at the bottom of certain data files. This document lists the different expansions and says a little bit about each one.

PARTICLE EXPANSIONS

Particle expansions go in PART?.TXT files, and are followed by a single number. Be sure to

precede each with a colon, or it won't be read.

```
:[TURN] 0 // This isn't very useful... Don't turn with character
:[ARMO] 0 // Particle is armor piercing ( Base defense of 0 )
:[BLOC] 0 // Particle can't be blocked by a shield
:[ARRO] 0 // Particle only damages the character to whom it is
attached
:[TIME] 0 // Don't set invincibility time on hit
:[PUSH] 0 // Don't push the enemy back on hit
:[ZSPD] 20 // 0 to 20 or so, Allows up/down aiming at targets
:[FSND] 2 // 0 to 15, Sound file to play when particle hits the floor
:[WSND] 4 // 0 to 15, Sound file to play when particle hits a wall
:[WEND] 1 // 0 or 1, Particle will end if it hits a wall
:[DLEV] -10 // -50 to 50 or so, Dynamic light level change, *1000
notation
:[DRAD] -10000 // -10000 to 10000 or so, Dynamic light radius change,
*1000
:[IDAM] 1 // Add 25% of the owners intelligence as bonus damage
:[WDAM] 1 // Add 25% of the owners wisdom as bonus damage
```

CHARACTER EXPANSIONS

Character expansions go in DATA.TXT files, and are followed by a single number. Be sure to precede each with a colon, or it won't be read.

```
:[ICON] 1 // Force the icon to show
:[STUK] 0 // Don't allow particles to stick to character ( arrows, etc. )
:[PACK] 0 // Don't let character be put in pack... For large items
:[VAMP] 1 // Character has no reflection
:[DRAW] 1 // Character is always drawn, even if far away from camera
:[RANG] 1 // Character is flagged as a ranged weapon
:[EQUI] 1 // Character is flagged as equipment, For worn items
:[SQUA] 1 // Character has a square base, Most are circular
:[SHAD] 1 // Draw the shadow, even if partially transparent
:[SKIN] 2 // 0 to 3, Override the skin given in SPAWN.TXT, For imports
:[DRES] 2 // 0 to 3, Flag a skin as "Dressy", May be used multiple times
:[CONT] 100 // 0 to 100 or so, Set the AI content to this
:[STAT] 50 // 0 to 100 or so, Set the AI state to this
:[HIDE] 3 // 0 to 100 or so, Don't draw when the AI state is this
:[GOLD] 200 // 0 to 9999, Set starting gold for the character
:[LEVL] 0 // Set the level to this +1, 0 is 1st level...
:[PLAT] 1 // 0 or 1, Character can use platforms?
:[RIPP] 0 // 0 or 1, Character makes ripples?
```

All the characters skills are also saved as expansions:

```
:[CKUR] 1 // Character can see kurses
:[POIS] 1 // Character can use poison without err
:[WMAG] 1 // Character can use Arcane magic
:[HMAG] 1 // Character can use divine magic
:[STAB] 1 // Character can backstab
:[READ] 1 // Character can read (Literate)
:[DISA] 1 // Character can find and disarm traps
:[AWEP] 1 // Character can use advanced weapons
:[TECH] 1 // Character can use technological items
:[JOUS] 1 // Character can joust with a lance
```

ENCHANT EXPANSIONS

Enchant expansions go in ENCHANT.TXT files, and are followed by a single number. Be sure to precede each with a colon, or it won't be read.

```
:[STAY] 1 // Stay around even after the owner has died
:[AMOU] 8 // 0 to 15, Amount to spawn each time
:[TIME] 25 // 1 to 100, Time until next spawn
:[TYPE] 0 // 0 to 7, Type of particles to spawn
:[FACE] 8192 // 0 to 65535, Facing add for each one...
:[SEND] 0 // 0 to 15, The end sound
:[OVER] 1 // 1 to 100, Create a child overlay object with this state

// The overlay follows the target around, and is killed when
// the enchantment ends...
:[CKUR] 1 // This allows the target to see kurses
```

MODULE EXPANSIONS

These go in the MENU.TXT files. These aren't entered by hand, rather a character in the module must use the AddIDSZ function to automagically append it to the file. Any given module may check another module to see if it has a given IDSZ, for hiding certain modules until a secret is found or the other module is beaten.

IDSZ

The IDSZ system is a way of describing objects using 4 letter words. For example, a crossbow has an IDSZ of [XBOW], and a sword has an IDSZ of [SWOR]. Why? It lets you add generic flags to the game, so other people can build on them. For example, I might want to make a sword that gets a bonus against Dragons. Since Dragons have a [DRAG] IDSZ, the sword's scripted AI can check if it hit a dragon, and do a little extra damage if it did. The nice part is that the weapon will also work against other [DRAG] creatures, like Dragonmen and Drag Queens, that I'm sure people will add in the future. The only way it will work though, is if everyone uses the same ID strings. So, before you make a [MINO] [TAUR], consult the lists located in IDSZ.txt (Located in the Egoboo Development package).

Making Modules

Common Problems

Egoboo Exits when attempting to load the module:

The first thing to check is the file Egoboo outputs. It creates a text file called "error.txt", which will contain the reason.

Slot Problems:

The most common cause for this error is a slot that is used twice. Look at Object Slots under Making Objects for more information if this is the problem.

Missing Start Point:

Another common cause is a missing start point. The start point must be added to either the spawn.txt file, or through EgoMap. In EgoMap, use the Objects mode, and pick a start point, and add it. For more information, see EgoMap's web page or ask in the Egoboo Forums.

MAKING PASSAGES

Passages are Egoboo's way of marking an area. The main uses for them are: Having a shop Having doors that can open or close Detecting a character's presence in an area.

Passages are contained in the file passage.txt in a module's gamedat folder. It looks like this (this is from the adventurer starter's passage file).

// TopleftXY BottomrightXY Open ShootThrough SlippyClose

0 Deadend: 3 51 4 52 F T F

1 Deadend: 53 7 54 8 F T F

2 Door: 40 53 41 54 F F F

3 Armor: 48 27 49 28 F F F

4 Trog: 17 18 18 19 F F F

5 MamaLeft: 24 25 25 26 F F F

Here's what it all means:

The number at the left is the number of that passage. You'll need this number later for scripting. The name is just to help you remember which passage is which. Neither of these are read by Egoboo and are just there to help you out.

The TopleftXY and BottomrightXY are the boundaries of the passage. Unfortunately, as of right now you can't set the coordinates in EgoMap, so you'll have to do it manually. The easiest way is like this.

1. Open your module in Egoboo

2. Hold F5. This will display the coordinates of your character.

3. Go to the top left and bottom right corners of each passage you want and write down the coordinates.

Zefz Notes: Personally, I just use EgoMap to edit and add passages. Set Show Passages to True (Done in EgoMap) and modify passage.txt. Now reload the module and see how the passage changed.

Open is T or F. This is asking whether the passage is open or closed when the module begins. Set it to whichever is true. Open or closed refers to whether the tiles the passage is over are passable or impassable; set this by changing flags in Egomap.

Shoot Through refers to whether or not you can shoot through the passage. If set to T, projectiles will go through; if F, projectiles will be stopped when they hit the passage.

I've never used slippy closing, but I think what it does is set the tiles in the passage as slippy tiles instead of regular tiles. Try it and find out (or just leave it F and play it safe).

So, now you've got a file, passage.txt, which has the info for all the passages in your module. Now open up spawn.txt. You'll notice that one of the values that can be set for each object is the Pas value, which refers to passage. For each object that uses a passage, set that object's Pas value to the number of the passage that is relevant (you remember those numbers from before, right?). One object can interact with any number of passages; the Pas value is just an easy way of remembering which object goes with which passage, and the scripts for doors and buttons use it, so if you don't want to modify the scripts, set the value.

Now, on to the uses of passages: For doors, set the door to its associated passage. The standard ones are all scripted to use that, and will work without any further need for changes. All you have to do to get the door to open and close is call OpenPassage and ClosePassage (Egoscript commands) on the associated passage. Note that OpenPassage will only work if the passage is closed, and ClosePassage will only work if the passage is open. Buttons can do this automatically; for a standard button, just make sure that it's associated with the same passage as the door you want and it will open/close the door with no further need for changes (just

make sure you have the right button; some require keys, some can close/open, and some can only either close or open). Any object can close and open doors/passages.

For shops. Please, don't script your own shop. Open the shopkeep object from Zippy City and base your shop on that. Basically, there is an Egoscript command, AddShopPassage, that turns a passage into a shop where you get paid for any item you drop and must pay for any item you pick up. The shop will function as long as the object that called it into existence is still alive.

Detecting character presence: The Egoscript function SetTargetToWhoeverIsInPassage does exactly what it sounds like it does. From that, you can test if the target is a player, is alive/dead, etc. for whatever you want.

Making your first passage will probably be annoying, but once you've made one it should be

pretty intuitive.

Other

Here are some definitions of short-words and terms (Where * is a number or a string):

STR = The STR stat is the character's strength.

WIS = The WIS stat is the character's wisdom.

INT = The INT stat is the character's intelligence.

DEX = The DEX stat is the character's dexterity

DEF = The DEF stat is the character's defence.

EXP = The EXP stat is the character's experience.

Mana = Spellpoints or spiritual energy, needed for using spells.

Mana Return = Rate of mana regeneration.

Mana Flow = How much mana the character can channel at the time

Cursed item = A cursed item gives bad effects on a character, only a wizard spell can remove cursed items.

Magical item = Magical items gives good effects and benefits to you character.

Potion = A liquid that gives an effect on your character. Example: Refill health.

Health/Hitpoints = Also called HP. the HP of a character determinates how much hits a character can take before he dies.

Zorkmids = Money ($), used to buy items in the sheep village.

*.OBJ = Items, characters and monsters are all named *.obj, which means object.

Example: Healer.Obj or Sword.Obj

data.txt = All objects have their own data.txt. In that text file you can edit every thing about them, such as health, exp and strength. Try experimenting with the text files, it is very easy editing them.

sound*.wav = Every sound a object makes is a wav. These wav files are played trough data.txt, enchant.txt or script.txt (The last one is the most common one).

part*.txt = These txt files describes how to draw different particles the object spawns. Fine examples are spell effects or blood (All weapons use particles to determine their damage). Example: part0.txt, part1.txt or part8.txt

naming.txt = This is the current name or the random name for a object.

message.txt = These are the different messages the object can show in the game. tris*.bmp = The skin which is drawn upon the model of the object. Every object can have up

up to four different skins (0, 1, 2 or 3). Example: tris0.bmp or tris3.bmp

Spell = A magical effect some characters can create trough the use of Mana

Various Egoboo sites:

http://egoboo.sourceforge.net/ – Official Egoboo website

http://egoboo.informe.com/forum/ -Egoboo Community forum

http://egoboo.informe.com/wiki/index.php/Main_Page -The EgoWiki

http://www.geocrawler.com/lists/3/SourceForge/6669/0/ -an archive of the old egoboo discussions.

http://zippy-egoboo.sourceforge.net -Old Egoboo community devolpment website

http://www.artistcollaboration.com/users/aworkinprogress/Software.htm -Egoboo C++

CONTACT:

ZefzSoftwares@hotmail.com -Lead designer

jonathan@bishopia.org -Coder

aworkinprogress@artistcollaboration.com -Coder

Feel free to report any errors or bugs you find and thank you for downloading Egoboo! If you have any problems running the game, dont hesitate to contact the forums.

Credits

Here are the credits to those who have contributed something to the game (List may not be

fully up to date and has no specific order):

ORIGINAL PROGRAMMERS

Aaron Bishop

Benjamin Bishop

MODULES

John Dick aka Booger

Johan Jansen aka Zefz

Arakon

Pteromys

CODING

Matty Noble aka Elmin

Arakon

Aleous

Jonathan Fischer

Ben Birdsey

Denis Jaimes

Johan Jansen aka Zefz

Gary Clark

Morgan

Ollipekka

Paco

Hans de Goede

SOUND AND MUSIC

Aleous

Klastek Timrak

OBJECTS

PurpleSquerkle

Johan Jansen aka Zefz

John Dick aka Booger

Hunna

Spyro

AiletheAlien

Pteromys

Ptapasu

ARTWORK

Hiroy

Chainsaw

ALSO WORTH MENTIONING

Arne Kristian Jansen

eatsonlyheads

Icezd

dolny

the happy pink elephant

Golem

Alazamir

Veiva

Clokinator aka Florian Natterer

Agent of Dread (Have a cookie!)

PurpleSquerkle

All the other supporters from the Egoboo Community!

By: Zefz – Egoboo Game Manual – Manual version 2.10 – Last updated 25.03.08

Work in progress

Divine Power

Holy Weapon

Touch of Death

Symbol of Death/Shock/Daze

Curse

Seeing

Mass Heal

Destroy Undead

Turn Undead

Recharge

Fire Rain

Prayer (Heal self)

Adventurer

Armor Defence Slashing Bashing Piercing Fire Holy Evil Ice Zap

Leather Tunic (50$) 30 Normal Normal Normal Normal Normal Normal Normal Normal

Chain Shirt (100$) 60 Resistant Normal Normal Normal Normal Normal Normal Normal

Breastplate (300$) 90 Resistant Normal Resistant Normal Normal Normal Normal Normal

Robin Costume (600$) 110 Resistant Normal Resistant Normal Normal Resistant Normal Normal

Armor Defence Slashing Bashing Piercing Fire Holy Evil Ice Zap

Leather Tunic (50$) 30 1 1 1 1 1 1 1 1

Chain Shirt (100$) 60 2 1 1 1 1 1 1 1

Breastplate (300$) 90 2 1 2 1 1 1 1 1

Robin Costume (600$) 110 2 1 2 1 1 2 1 1

Setup Information

You can change graphical and audio settings in-game trough the options menu. If you want to change input controls tough, you will need to edit controls.txt (See the section about controls for more information).

GRAPHICS

Texture Filtering - This manages the quality in which manner textures are filtered.

Fullscreen - Should the game be played fullscreen or not?

Z Bit - Enhances the color depth quality if set higher (32 is maximum).

Screensize - Overall graphical quality – higher number increases graphic quality.

Max Number of Text Messages - How many information messages can be displayed at the same time (Reduce this only if they cover too much of your screen – in case of low screensize).

Antialasing - This will enable antialiasing which increase model detail and sharper borders.

Particle Effects - This manages which types of particles the game is to load. Normal is standard, Smooth looks better, but loses some detail while Fast has low color bits and are faster to render.

Autoturn Camera - Adjust the autoturn function for the camera which follows the players.

Reflection - This decides the quality of floor reflections or disable reflections.

Shadows - This sets shadow quality or disables it altogether.

3D Effects - This option will increase 3D quality and enable special 3D effects.

Pretty Water - Enabling this will increase water graphic quality.

Vertical Sync - Wait for vertical synchronization? Enable this if you see tearing on the screen.

SOUND

Sound - Allow sounds to be played?

Sound Volume - The volume of all sound effects.

Music - Allow interactive music to be played?

Music Volume - The sound volume of all musical tracks that are played.

Buffer Size - This will reduce lags and stutters in music, but will cause a overall delay for all sounds played. You will need to find a balance for your pc (So that sounds arent too much delayed and music works normally).

Max Sound Channels - The maximum number of sounds that can be played at the same time.

Egoscript Reference

Here is a reference to all the egoscript functions that are implemented at the time of the last update 09-30-2009.

For each function, you will see the egoscript function name, an approximation to how this function would be called in Lua, and a text description of what the function does.

fSpawned = "*if(Spawned()) then*"
This function proceeds if the character was spawned this update

IfTimeOut = "*if(TimeOut()) then*"
This function proceeds if the character's aitime is 0... Use in conjunction with set_Time

IfAtWaypoint = "*if(AtWaypoint()) then*" -
This function proceeds if the character reached its waypoint this update

IfAtLastWaypoint = "*if(AtLastWaypoint()) then*" -
This function proceeds if the character reached its last waypoint this update

IfAttacked = "*if(Attacked()) then*" -
This function proceeds if the character (an item) was put in its owner's pocket this update

IfBumped = "*if(Bumped()) then*" -
This function proceeds if the character was bumped by another character this update

IfOrdered = "*if(Ordered()) then*" -
This function proceeds if the character got an order from another character on its team this update

IfCalledForHelp = "*if(CalledForHelp()) then*" -
This function proceeds if one of the character's teammates was nearly killed this update

SetContent – *"self.content = self.argument"*
This function sets the content variable... Used in conjunction with GetContent... Content is preserved from update to update

IfKilled = "*if(Killed()) then*" -
This function proceeds if the character was killed this update

IfTargetKilled = "*if(TargetKilled()) then*" -
This function proceeds if the character's target from last update was killed during this update

ClearWaypoints = *"ClearWaypoints()"*
This function is used to move a character around... Do this before AddWaypoint

AddWaypoint = *"AddWaypoint(tmpx = "x position", tmpy = "y position")"*
This function tells the character where to move next

FindPath = *"FindPath()"*
NOTE: This function doesn't work yet

Compass = *"Compass(tmpturn = "rotation", tmpdistance = "radius")"*
This function modifies tmpx and tmpy, depending on the setting of tmpdistance and tmpturn. It acts like one of those Compass thing with the two little needle legs

GetTargetArmorPrice = *"tmpx = GetTargetArmorPrice(tmpargument = "skin")"*
This function returns the cost of the desired skin upgrade, setting tmpx to the price

SetTime = *"SetTime(tmpargument = "time")"*
This function sets the character's time... 50 clicks per second... Used in conjunction with _TimeOut

GetContent = *"tmpargument = self.content"*
This function sets tmpargument to the character's content variable... Used in conjunction with set_Content, or as a NOP to space out an Else

JoinTargetTeam = "JoinTargetTeam()"
This function lets a character join a different team... Used mostly for pets

SetTargetToNearbyEnemy = *"SetTargetToNearbyEnemy()"*
This function sets the target to a nearby enemy, failing if there are none

SetTargetToTargetLeftHand = *"SetTargetToTargetLeftHand()"*
This function sets the target to the item in the target's left hand, failing if the target has no left hand item

SetTargetToTargetRightHand = *"SetTargetToTargetRightHand()"*
This function sets the target to the item in the target's right hand, failing if the target has no right hand item

SetTargetToWhoeverAttacked = *"SetTargetToWhoeverAttacked()"*
This function sets the target to whoever attacked the character last, failing for damage tiles

SetTargetToWhoeverBumped = *"SetTargetToWhoeverBumped()"*
This function sets the target to whoever bumped the character last. It never fails

SetTargetToWhoeverCalledForHelp = *"SetTargetToWhoeverCalledForHelp()"*
This function sets the target to whoever called for help last...

SetTargetToOldTarget = *"self.target = self.old_target"*
This function sets the target to the target from last update, used to undo other SetTarget* functions

SetTurnModeToVelocity = *"SetTurnModeToVelocity()"*
This function sets the character's movement mode to the default

SetTurnModeToWatch = *"SetTurnModeToWatch()"*
This function makes the character look at its next waypoint, usually used with close waypoints or the Stop function

SetTurnModeToSpin = *"SetTurnModeToSpin()"*
This function makes the character spin around in a circle, usually used for magical items and such

SetBumpHeight = *"SetBumpHeight(tmpargument = "height")"*
This function makes the character taller or shorter, usually used when the character dies

IfTargetHasID = "*if(TargetHasID(tmpargument = "idsz")) then*"
This function proceeds if the target has either a parent or type IDSZ matching tmpargument...

IfTargetHasItemID = "*if(TargetHasItemID(tmpargument = "idsz")) then*"
This function proceeds if the target has a matching item in his/her pockets or hands.

IfTargetHoldingItemID = "*if(TargetHoldingItemID(tmpargument = "idsz")) then*"
This function proceeds if the target has a matching item in his/her hands. It also sets tmpargument to the proper latch button to press to use that item

IfTargetHasSkillID = "*if(TargetHasSkillID(tmpargument = "skill idsz")) then*"
This function proceeds if ID matches tmpargument

Else = *"else"*
This function proceeds if the last function failed... Need padding before it... See the above function for an example...

Run = *"Run()"*
This function sets the character's maximum acceleration to its actual maximum

Walk = *"Walk()"*
This function sets the character's maximum acceleration to 66% of its actual maximum

Sneak = *"Sneak()"*
This function sets the character's maximum acceleration to 33% of its actual maximum

DoAction = *"DoAction(tmpargument = "action")"*
This function makes the character do a given action if it isn't doing anything better. Fails if the action is invalid or

if the character is doing something else already

KeepAction = *"KeepAction()"*
This function makes the character's animation stop on its last frame and stay there... Usually used for dropped items

IssueOrder = *"IssueOrder(tmpargument = "order")"*
This function tells all of the character's teammates to do something, though each teammate needs to interpret the order using _Ordered in its own script...

DropWeapons = *"DropWeapons()"*
This function drops the character's in-hand items... It will also buck the rider if the character is a mount

TargetDoAction = *"TargetDoAction(tmpargument = "action")"*
The function makes the target start a new action, if it is valid for the model It will fail if the action is invalid or if the target is doing something else already

OpenPassage = *"OpenPassage(tmpargument = "passage")"*
This function allows movement over the given passage area. Fails if the passage is already open Passage areas are defined in passage.txt and set in spawn.txt for the given character

ClosePassage = *"ClosePassage(tmpargument = "passage")"*
This function prohibits movement over the given passage area, proceeding if the passage isn't blocked. Crushable characters within the passage are crushed.

IfPassageOpen = "*if(PassageOpen(tmpargument = "passage")) then*" -
This function proceeds if the given passage is valid and open to movement Used mostly by door characters to tell them when to run their open animation.

GoPoof = *"GoPoof()"*
This function flags the character to be removed from the game entirely. This doesn't work on players

CostTargetItemID = *"CostTargetItemID(tmpargument = "idsz")"*
This function proceeds if the target has a matching item, and poofs that item... For one use keys and such

DoActionOverride = *"DoActionOverride(tmpargument = "action")"*
This function makes the character do a given action no matter what

IfHealed = "*if(Healed()) then*" -
This function proceeds if the character was healed by a healing particle

SendPlayerMessage = *"SendPlayerMessage(tmpargument = "message number")"*
This function sends a message to the players

CallForHelp = *"CallForHelp()"*
This function calls all of the character's teammates for help... The teammates must use IfCalledForHelp in their scripts

AddIDSZ = *"AddIDSZ(tmpargument = "idsz")"*
This function slaps an expansion IDSZ onto the module description... Used to show completion of special quests for a given module

SetState = *"self.state = tmpargument"*
This function sets the character's state.
VERY IMPORTANT... State is preserved from update to update

GetState =*"tmpargument = self.state"*
This function reads the character's state variable

IfStateIs = "*if(StateIs(tmpargument = "state")) then*" -
This function proceeds if the character's state equals tmpargument

IfTargetCanOpenStuff = "*if(TargetCanOpenStuff()) then*" -
This function proceeds if the target can open stuff (set in data.txt) . Used by chests and buttons and such so only "smart" creatures can operate them

IfGrabbed = "*if(Grabbed()) then*" -
This function proceeds if the character was grabbed this update... Used mostly by item characters

IfDropped = "*if(Dropped()) then*" -
This function proceeds if the character was dropped this update... Used mostly by item characters

SetTargetToWhoeverIsHolding = *"SetTargetToWhoeverIsHolding()"*
This function sets the target to the character's holder or mount, failing if the character is not held

DamageTarget = *"DamageTarget(tmpargument = "damage")"*
This function damages the target damage;

IfXIsLessThanY = "*if(XIsLessThanY(tmpx, tmpy)) then*" -
This function proceeds if tmpx is less than tmpy...

SetWeatherTime = *"SetWeatherTime(tmpargument = "time") "*
This function can be used to slow down or speed up or stop rain and other weather effects

GetBumpHeight = *"tmpargument = GetBumpHeight()"*
This function sets tmpargument to the character's height

IfReaffirmed = "*if(Reaffirmed()) then*" -
This function proceeds if the character was damaged by its reaffirm damage type... Used to relight the torch...

UnkeepAction = *"UnkeepAction()"*
This function undoes KeepAction

IfTargetIsOnOtherTeam = "*if(TargetIsOnOtherTeam()) then*" -
This function proceeds if the target is on another team

IfTargetIsOnHatedTeam = "*if(TargetIsOnHatedTeam()) then*" -
This function proceeds if the target is on an enemy team

PressLatchButton = *"PressLatchButton(tmpargument = "latch bits")"*
This function emulates joystick button presses

SetTargetToTargetOfLeader = *"SetTargetToTargetOfLeader()"*
This function sets the character's target to the target of its leader, or it fails with no change if the leader is dead

IfLeaderKilled = "*if(LeaderKilled()) then*" -
This function proceeds if the team's leader died this update

BecomeLeader = *"BecomeLeader()"*
This function makes the character the leader of the team

ChangeTargetArmor = *"ChangeTargetArmor(tmpargument = "armor")"*
This function sets the armor type of the target... Used for chests Sets tmpargument as the old type and tmpx as the new type

GiveMoneyToTarget = *"GiveMoneyToTarget(tmpargument = "money")"*

This function increases the target's money, while decreasing the character's own money. tmpargument is set to the amount transferred

DropKeys = *"DropKeys()"*
This function drops all of the keys in the character's inventory. This does NOT drop keys in the character's hands.

IfLeaderIsAlive = "*if(LeaderIsAlive()) then*" -
This function proceeds if the team has a leader

IfTargetIsOldTarget = "*if(TargetIsOldTarget()) then*" -
This function proceeds if the target is the same as it was last update

SetTargetToLeader = *"SetTargetToLeader()"*
This function sets the target to the leader, proceeding if their is a valid leader for the character's team

SpawnCharacter = *"SpawnCharacter(tmpx = "x", tmpy = "y", tmpturn = "turn", tmpdistance = "speed")"*
This function spawns a character of the same type as the spawner... This function spawns a character, failing if x,y is invalid This is horribly complicated to use, so see ANIMATE.OBJ for an example tmpx and tmpy give the coodinates, tmpturn gives the new character's direction, and tmpdistance gives the new character's initial velocity

RespawnCharacter = *"RespawnCharacter()"*
This function respawns the character at its starting location... Often used with the Clean functions

ChangeTile = *"ChangeTile(tmpargument = "tile type") "*
This function changes the tile under the character to the new tile type, which is highly module dependent

IfUsed = "*if(Used()) then*" -
This function proceeds if the character was used by its holder or rider... Character's cannot be used if their reload time is greater than 0

DropMoney = *"DropMoney(tmpargument = "money")"*
This function drops a certain amount of money, if the character has that much

SetOldTarget() -= "self.old_target = self.target"
This function sets the old target to the current target... To allow greater manipulations of the target

DetachFromHolder = *"DetachFromHolder()"*
This function drops the character or makes it get off its mount Can be used to make slippery weapons, or to make certain characters incapable of wielding certain weapons... "A troll can't grab a torch"

IfTargetHasVulnerabilityID = "*if(TargetHasVulnerabilityID(tmpargument = "vulnerability idsz")) then*" -
This function proceeds if the target is vulnerable to the given IDSZ...

CleanUp = *"CleanUp()"*
This function tells all the dead characters on the team to clean themselves up... Usually done by the boss creature every second or so

IfCleanedUp = "*if(CleanedUp()) then*" -
This function proceeds if the character is dead and if the boss told it to clean itself up

IfSitting = "*if(Sitting()) then*" -
This function proceeds if the character is riding a mount

IfTargetIsHurt = "*if(TargetIsHurt()) then*" -
This function passes only if the target is hurt and alive

IfTargetIsAPlayer = "*if(TargetIsAPlayer()) then*" -
This function proceeds if the target is controlled by a human (may not be local)

PlaySound = *"PlaySound(tmpargument = "sound")"*
This function plays one of the character's sounds. The sound fades out depending on its distance from the viewer

SpawnParticle = *"SpawnParticle(tmpargument = "particle", tmpdistance = "character vertex", tmpx = "offset x", tmpy = "offset y")"*
This function spawns a particle, offset from the character's location

IfTargetIsAlive = "*if(TargetIsAlive()) then*" -
This function proceeds if the target is alive

Stop = *"Stop()"*
This function sets the character's maximum acceleration to 0... Used along with Walk and Run and Sneak

DisaffirmCharacter = *"DisaffirmCharacter()"*
This function removes all the attached particles from a character (stuck arrows, flames, etc)

ReaffirmCharacter = *"ReaffirmCharacter()"*
This function makes sure it has all of its reaffirmation particles attached to it. Used to make the torch light again

IfTargetIsSelf = "*if(TargetIsSelf()) then*" -
This function proceeds if the character is targeting itself

IfTargetIsMale = "*if(TargetIsMale()) then*" -
This function proceeds only if the target is male

IfTargetIsFemale = "*if(TargetIsFemale()) then*" -
This function proceeds if the target is female

SetTargetToSelf() - *"self.target = self.index"*
This function sets the target to the character itself

SetTargetToRider = *"SetTargetToRider()"*
This function sets the target to whoever is riding the character (left/only grip), failing if there is no rider

GetAttackTurn = *"tmpturn = GetAttackTurn()"*
This function sets tmpturn to the direction from which the last attack came. Not particularly useful in most cases, but it could be.

GetDamageType = *"tmpargument = GetDamageType()"*
This function sets tmpargument to the damage type of the last attack that hit the character

BecomeSpell = *"BecomeSpell()"*
This function turns a spellbook character into a spell based on its content.
TOO COMPLICATED TO EXPLAIN... SHOULDN'T EVER BE NEEDED BY YOU...

BecomeSpellbook = *"BecomeSpellbook()"*
This function turns a spell character into a spellbook and sets the content accordingly.
TOO COMPLICATED TO EXPLAIN. Just copy the spells that already exist, and don't change them too much

IfScoredAHit = "*if(ScoredAHit()) then*" -
This function proceeds if the character damaged another character this update... If it's a held character it also sets the target to whoever was hit

IfDisaffirmed = "*if(Disaffirmed()) then*" -
This function proceeds if the character was disaffirmed... This doesn't seem useful anymore...

TranslateOrder = *"tmpx,tmpy,tmpargument = TranslateOrder()"*
This function translates a packed order into understandable values... See CreateOrder for more... This function sets tmpx, tmpy, tmpargument, and possibly sets the target (which may not be good) -

SetTargetToWhoeverWasHit = *"SetTargetToWhoeverWasHit()"*
This function sets the target to whoever was hit by the character last

SetTargetToWideEnemy = *"SetTargetToWideEnemy()"*
This function sets the target to an enemy in the vicinity around the character, failing if there are none

IfChanged = "*if(Changed()) then*" -
This function proceeds if the character has changed shape. Needed for morph spells and such

IfInWater = "*if(InWater()) then*" -
This function proceeds if the character has just entered into some water this update (and the water is really water, not fog or another effect)

IfBored = "*if(Bored()) then*" -
This function proceeds if the character has been standing idle too long

IfTooMuchBaggage = "*if(TooMuchBaggage()) then*" -
This function proceeds if the character tries to put an item in his/her pockets, but the character already has 6 items in the inventory. Used to tell the players what's going on.

IfGrogged = "*if(Grogged()) then*" -
This function proceeds if the character has been grogged (a type of confusion) this update

IfDazed = "*if(Dazed()) then*" -
This function proceeds if the character has been dazed (a type of confusion) this update

IfTargetHasSpecialID = "*if(TargetHasSpecialID(tmpargument = "special idsz")) then*" -
This function proceeds if the character has a special IDSZ (in data.txt)

PressTargetLatchButton = *"PressLatchButton(self.target, tmpargument = "latch bits")"*
This function mimics joystick button presses for the target. For making items force their own usage and such

IfInvisible = "*if(Invisible()) then*" -
This function proceeds if the character is invisible

IfArmorIs = "*if(ArmorIs(tmpargument = "skin")) then*" -
This function proceeds if the character's skin type equals tmpargument

GetTargetGrogTime = *"tmpargument = GetTargetGrogTime()"*
This function sets tmpargument to the number of updates before the character is ungrogged, proceeding if the number is greater than 0

GetTargetDazeTime = *"tmpargument = GetTargetDazeTime()"*
This function sets tmpargument to the number of updates before the character is undazed, proceeding if the number is greater than 0

SetDamageType = *"SetDamageType(tmpargument = "damage type")"*
This function lets a weapon change the type of damage it inflicts

SetWaterLevel = *"SetWaterLevel(tmpargument = "level")"*
This function raises or lowers the water in the module

EnchantTarget = *"Enchant(self.target)"*
This function enchants the target with the enchantment given in enchant.txt. Make sure you use SetOwnerToTarget before doing this.

EnchantChild = *"Enchant(self.child)"*
This function enchants the last character spawned with the enchantment given in enchant.txt. Make sure you use set_OwnerToTarget before doing this.

TeleportTarget = Teleport(self.target, tmpx = "x", tmpy = "y") -
This function teleports the target to the X, Y location, failing if the location is off the map or blocked

GiveExperienceToTarget = *"GiveExperience(self.target, tmpargument = "amount", tmpdistance = "xptype")"*
This function gives the target some experience, xptype from distance, amount from argument...

IncreaseAmmo = IncreaseAmmo() -
This function increases the character's ammo by 1

UnkurseTarget = Unkurse(self.target) -
This function unkurses the target

GiveExperienceToTargetTeam = *"GiveExperienceToTeam(self.target, tmpargument = "amount", tmpdistance = "type")"*
This function gives experience to everyone on the target's team

IfUnarmed = "*if(Unarmed()) then*" -
This function proceeds if the character is holding no items in hand.

RestockTargetAmmoIDAll = RestockTargetAmmoIDAll(tmpargument = "idsz") -
This function restocks the ammo of all of the target's items, if those items have a matching parent or type IDSZ

RestockTargetAmmoIDFirst = RestockTargetAmmoIDFirst(tmpargument = "idsz") -
This function restocks the ammo of the first item the character is holding, if the item matches the ID given (parent or child type) -

FlashTarget = *"Flash(self.target)"*
This function makes the target flash

SetRedShift = SetRedShift(tmpargument = "red darkening") -
This function sets the character's red shift (0 - 3), higher values making the character less red and darker

SetGreenShift = SetGreenShift(tmpargument = "green darkening") -
This function sets the character's green shift (0 - 3), higher values making the character less red and darker

SetBlueShift = SetBlueShift(tmpargument = "blue darkening") -
This function sets the character's blue shift (0 - 3), higher values making the character less red and darker

SetLight = SetLight(tmpargument = "lighness") -
This function alters the character's transparency (0 - 254), = no transparency

SetAlpha = SetAlpha(tmpargument = "alpha") -
This function alters the character's transparency (0 - 255) , 255 = no transparency

IfHitFromBehind = "*if(HitFromBehind()) then*" -
This function proceeds if the last attack to the character came from behind

IfHitFromFront = "*if(HitFromFront()) then*" -
This function proceeds if the last attack to the character came from the front

IfHitFromLeft = "*if(HitFromLeft()) then*" -
This function proceeds if the last attack to the character came from the left

IfHitFromRight = "*if(HitFromRight()) then*" -
This function proceeds if the last attack to the character came from the right

IfTargetIsOnSameTeam = "*if(TargetIsOnSameTeam()) then*" -
This function proceeds if the target is on the character's team

KillTarget = Kill(self.target)
This function kills the target

UndoEnchant = UndoEnchant() -
This function removes the last enchantment spawned by the character, proceeding if an enchantment was removed

GetWaterLevel = tmpargument = GetWaterLevel() -
This function sets tmpargument to the current douse level for the water * 10. A waterlevel in wawalight of 85 would set tmpargument to 850

CostTargetMana = *"CostMana(self.target, tmpargument = "amount")"*
This function costs the target a specific amount of mana, proceeding if the target was able to pay the price... The amounts are * 256

IfTargetHasAnyID = "*if(TargetHasAnyID(tmpargument = "idsz")) then*" -
This function proceeds if the target has any IDSZ that matches the given one

SetBumpSize = SetBumpSize(tmpargument = "size") -
This function sets the how wide the character is

IfNotDropped = "*if(NotDropped()) then*" -
This function proceeds if the character is kursed and another character was holding it and tried to drop it

IfYIsLessThanX = "*if(YIsLessThanX()) then*" -
This function proceeds if tmpy is less than tmpx

SetFlyHeight = SetFlyHeight(tmpargument = "height") -
This function makes the character fly (or fall to ground if 0)

IfBlocked = "*if(Blocked()) then*" -
This function proceeds if the character blocked the attack of another character this update

IfTargetIsDefending = "*if(TargetIsDefending()) then*" -
This function proceeds if the target is holding up a shield or similar defense

IfTargetIsAttacking = "*if(TargetIsAttacking()) then*" -
This function proceeds if the target is doing an attack action

IfStateIs0 = *"if(self.state = 0) then"*

IfStateIs1 = *"if(self.state = 1) then"*

IfStateIs2 = *"if(self.state = 2) then"*

IfStateIs3 = *"if(self.state = 3) then"*

IfStateIs4 = *"if(self.state = 4) then"*

IfStateIs5 = *"if(self.state = 5) then"*

IfStateIs6 = *"if(self.state = 6) then"*

IfStateIs7 = *"if(self.state = 7) then"*

IfContentIs = "*if(self.content = tmpargument) then*" -
This function proceeds if the content matches tmpargument

SetTurnModeToWatchTarget = *"SetTurnModeToWatchTarget()"*
This function makes the character face its target, no matter what direction it is moving in... Undo this with set_TurnModeToVelocity

IfStateIsNot = "*if(StateIsNot(tmpargument = "test")) then*" -
This function proceeds if the character's state does not equal tmpargument

IfXIsEqualToY = *"if(self.x == self.y) then"*
These functions proceed if tmpx and tmpy are the same

DebugMessage = DebugMessage() -
This function spits out some useful numbers

BlackTarget = Black(self.target) -
The opposite of FlashTarget, causing the target to turn black

SendMessageNear = SendMessageNear(tmpargument = "message") -
This function sends a message if the camera is in the nearby area

IfHitGround = "*if(HitGround()) then*" -
This function proceeds if a character hit the ground this update... Used to determine when to play the sound for a dropped item

IfNameIsKnown = "*if(NameIsKnown()) then*" -
This function proceeds if the character's name is known

IfUsageIsKnown = "*if(UsageIsKnown()) then*" -
This function proceeds if the character's usage is known

IfHoldingItemID = "*if(HoldingItemID(tmpargument = "idsz")) then*" -
This function proceeds if the character is holding a specified item in hand, setting tmpargument to the latch button to press to use it

IfHoldingRangedWeapon = "*if(HoldingRangedWeapon()) then*" -
This function passes if the character is holding a ranged weapon, returning the latch to press to use it. This also checks ammo/ammoknown.

IfHoldingMeleeWeapon = "*if(HoldingMeleeWeapon()) then*" -
This function proceeds if the character is holding a specified item in hand, setting tmpargument to the latch button to press to use it

IfHoldingShield = "*if(HoldingShield()) then*" -
This function proceeds if the character is holding a specified item in hand, setting tmpargument to the latch button to press to use it. The button will need to be held down.

IfKursed = "*if(Kursed()) then*" -

This function proceeds if the character is kursed

IfTargetIsKursed = "*if(IsKursed(self.target)) then*" -
This function proceeds if the target is kursed

IfTargetIsDressedUp = "*if(IsDressedUp(self.target)) then*" -
This function proceeds if the target is dressed in fancy clothes

IfOverWater = "*if(OverWater()) then*" -
This function proceeds if the character is on a water tile

IfThrown = "*if(Thrown()) then*" -
This function proceeds if the character was thrown this update.

MakeNameKnown = MakeNameKnown() -
This function makes the name of the character known, for identifying weapons and spells and such

MakeUsageKnown = MakeUsageKnown() -
This function makes the usage known for this type of object For XP gains from using an unknown potion or such

StopTargetMovement = *"Stop(self.target)"*
This function makes the target stop moving temporarily Sets the target's x and y velocities to 0, and sets the z velocity to 0 if the character is moving upwards. This is a special function for the IronBall object

SetXY = SetXY(tmpargument = "index", tmpx = "x", tmpy = "y") -
This function sets one of the 8 permanent storage variable slots (each of which holds an x,y pair) -

GetXY = tmpx,tmpy = GetXY(tmpargument = "index") -
This function reads one of the 8 permanent storage variable slots, setting tmpx and tmpy accordingly

AddXY = tmpx,tmpy = AddXY(tmpargument = "index", tmpx = "x", tmpy = "y") -
This function alters the contents of one of the 8 permanent storage slots

MakeAmmoKnown = MakeAmmoKnown() -
This function makes the character's ammo known (for items)

SpawnAttachedParticle = SpawnAttachedParticle(tmpargument = "particle", tmpdistance = "vertex") -
This function spawns a particle attached to the character

SpawnExactParticle = SpawnExactParticle(tmpargument = "particle", tmpx = "x", tmpy = "y", tmpdistance = "z") -
This function spawns a particle at a specific x, y, z position

AccelerateTarget = AccelerateTarget(tmpx = "acc x", tmpy = "acc y") -
This function changes the x and y speeds of the target

IfdistanceIsMoreThanTurn - *"if(self.distance > sefl.turn) then"*
This function proceeds tmpdistance is greater than tmpturn

IfCrushed = "*if(Crushed()) then*" -
This function proceeds if the character was crushed in a passage this update...

MakeCrushValid = MakeCrushValid() -
This function makes a character able to be crushed by closing doors and such

SetTargetToLowestTarget = SetTargetToLowestTarget() -
This function sets the target to the absolute bottom character. The holder of the target, or the holder of the holder of the target, or the holder of the holder of ther holder of the target, etc.

IfNotPutAway = "*if(NotPutAway()) then*" -
This function proceeds if the character couldn't be put into another character's pockets for some reason. It might be kursed or too big or something

IfTakenOut = "*if(TakenOut()) then*" -
This function proceeds if the character is equiped in another's inventory, and the holder tried to unequip it (take it out of pack), but the item was kursed and didn't cooperate

IfAmmoOut = "*if(AmmoOut()) then*" -
This function proceeds if the character itself has no ammo left. This is for crossbows and such, not archers.

PlaySoundLooped = PlaySoundLooped(tmpargument = "sound", tmpdistance = "frequency") -
This function starts playing a continuous sound

StopSound = StopSound(tmpargument = "sound") -
This function stops the playing of a continuous sound!

HealSelf = HealSelf() -
This function gives life back to the character. Values given as * 256 This does NOT remove [HEAL] enchants (poisons)

Equip = Equip() -
This function flags the character as being equipped. This is used by equipment items when they are placed in the inventory

IfTargetHasItemIDEquipped = "*if(TargetHasItemIDEquipped(tmpargument = "item idsz")) then*" -
This function proceeds if the target already wearing a matching item

SetTurnModeToWatchTarget = *"SetTurnModeToWatchTarget()"*
This function must be called before enchanting anything. The owner is the character that pays the sustain costs and such for the enchantment

SetTargetToOwner *"self.target = self.owner"*
This function sets the target to whoever was previously declared as the owner.

SetFrame = SetFrame(tmpargument = "frame") -
This function sets the current .MD2 frame for the character... Values are * 4

BreakPassage = *"BreakPassage(tmpargument = "passage", tmpturn = "tile type", tmpdistance = "number of frames", tmpx = "animate ?", tmpy = "tile fx bits")"*
This function makes the tiles fall away (turns into damage terrain). This function causes the tiles of a passage to increment if stepped on. tmpx and tmpy are both set to the location of whoever broke the tile if the function passed...

SetReloadTime = SetReloadTime(tmpargument = "time") -
This function stops a character from being used for a while... Used by weapons to slow down their attack rate... 50 clicks per second...

SetTargetToWideBlahID = SetTargetToWideBlahID(tmpargument = "idsz", tmpdistance = "blah bits") -
This function sets the target to a character that matches the description, and who is located in the general vicinity of the character

PoofTarget = "Poof(self.target)"
This function removes the target from the game, failing if the target is a player

ChildDoActionOverride = *"DoActionOverride(self.child, tmpargument = action)"*
This function lets a character set the action of the last character it spawned. It also sets the current frame to the

first frame of the action (no interpolation from last frame).

SpawnPoof = *"SpawnPoof ()"*
This function makes a poof at the character's location. The poof form and particle types are set in data.txt

SetSpeedPercent = SetSpeedPercent(tmpargument = "percent") -
This function acts like Run or Walk, except it allows the explicit setting of the speed

SetChildState = "self.child.state = tmpargument"
This function lets a character set the state of the last character it spawned

SpawnAttachedSizedParticle = SpawnAttachedSizedParticle(tmpargument = "particle", tmpdistance = "vertex", tmpturn = "size") -
This function spawns a particle of the specific size attached to the character. For spell charging effects

ChangeArmor = ChangeArmor(tmpargument = "time") -
This function changes the character's armor. Sets tmpargument as the old type and tmpx as the new type

ShowTimer = ShowTimer(tmpargument = "time") -
This function sets the value displayed by the module timer... For races and such... 50 clicks per second

IfFacingTarget = "*if(FacingTarget()) then*" -
This function proceeds if the character is more or less facing its target

PlaySoundVolume = PlaySoundVolume(argument = "sound", distance = "volume") -
This function sets the volume of a sound and plays it

SpawnAttachedFacedParticle = SpawnAttachedFacedParticle(tmpargument = "particle", tmpdistance = "vertex", tmpturn = "turn") -
This function spawns a particle attached to the character, facing the same direction given by tmpturn

IfStateIsOdd = "*if(StateIsOdd()) then*" -
This function proceeds if the character's state is 1, 3, 5, 7, etc.

SetTargetToDistantEnemy = SetTargetToDistantEnemy(tmpdistance = "distance") -
This function finds a character within a certain distance of the character, failing if there are none

Teleport = *"Teleport(self, tmpx = "x", tmpy = "y")"*
This function teleports the character to a new location, failing if the location is blocked or off the map

GiveStrengthToTarget = *"GiveStrength(self.target)"*
Permanently boost the target's strength

GiveWisdomToTarget = *"GiveWisdom(self.target)"*
Permanently boost the target's wisdom

GiveIntelligenceToTarget = *"GiveIntelligence(self.target)"*
Permanently boost the target's intelligence

GiveDexterityToTarget = *"GiveDexterity(self.target)"*
Permanently boost the target's dexterity

GiveLifeToTarget = *"GiveLife(self.target)"*
Permanently boost the target's life

GiveManaToTarget = *"GiveMana(self.target)"*
Permanently boost the target's mana

ShowMap = *"ShowMap()"*
This function shows the module's map. Fails if map already visible

ShowYouAreHere = ShowYouAreHere() -
This function shows the blinking white blip on the map that represents the camera location

ShowBlipXY = ShowBlipXY(tmpx = "x", tmpy = "y", tmpargument = "color") -
This function draws a blip on the map, and must be done each update

HealTarget = *"HealOther(self.target, tmpargument = "amount")"*
This function gives some life back to the target. Values are * 256 Any enchantments that are removed by [HEAL], like poison, go away

PumpTarget = *"Pump(self.target, tmpargument = "amount")"*
This function gives some mana back to the target. Values are * 256

CostAmmo = CostAmmo() -
This function costs the character 1 point of ammo

MakeSimilarNamesKnown = MakeSimilarNamesKnown() -
This function makes the names of similar objects known. Checks all 6 IDSZ types to make sure they match.

SpawnAttachedHolderParticle = SpawnAttachedHolderParticle(tmpargument = "particle", tmpdistance = "vertex") -
This function spawns a particle attached to the character's holder, or to the character if no holder

SetTargetReloadTime = *"SetReloadTime(self.target, tmpargument = "time")"*
This function sets the target's reload time This function stops the target from attacking for a while.

SetFogLevel = SetFogLevel(tmpargument = "level") -
This function sets the level of the module's fog. Values are * 10
NOTE: Fog has been disabled

GetFogLevel = tmpargument = GetFogLevel()
This function sets tmpargument to the level of the module's fog... Values are * 10

SetFogTAD =*"SetFogColor(self.turn, self.argument, self.distance)"*
This function sets the color of the module's fog. TAD stands for <turn, argument, distance> == <red, green, blue>. Makes sense, huh?
NOTE: Fog has been disabled

SetFogBottomLevel = SetFogBottomLevel(tmpargument = "level") -
This function sets the level of the module's fog. Values are * 10 float fTmp;
NOTE: Fog has been disabled

CorrectActionForHand = CorrectActionForHand(tmpargument = "action") -
This function changes tmpargument according to which hand the character is held in It turns ZA into ZA, ZB, ZC, or ZD...
USAGE: wizards casting spells

IfTargetIsMounted = "*if(TargetIsMounted()) then*" -
This function proceeds if the target is riding a mount

SparkleIcon = SparkleIcon(tmpargument = "color") -
This function starts little sparklies going around the character's icon

UnsparkleIcon = UnsparkleIcon() -
This function stops little sparklies going around the character's icon

GetTileXY = “tmpargument = GetTileXY(tmpx = "x", tmpy = "y")”
This function sets tmpargument to the tile type at the specified coordinates

SetShadowSize = *“SetShadowSize(tmpargument = "size")”*
This function makes the character's shadow bigger or smaller

OrderTarget = *“IssueOrder(self.target, tmpargument = "order")”*
This function issues an order to the given target Be careful in using this, always checking IDSZ first

SetTargetToWhoeverIsInPassage = SetTargetToWhoeverIsInPassage() -
This function sets the target to whoever is blocking the given passage This function lets passage rectangles be used as event triggers

IfCharacterWasABook = “*if(CharacterWasABook()) then*” -
This function proceeds if the base model is the same as the current model or if the base model is SPELLBOOK USAGE: USED BY THE MORPH SPELL. Not much use elsewhere

SetEnchantBoostValues = SetEnchantBoostValues(tmpargument = "owner mana regen", tmpdistance = "owner life regen", tmpx = "target mana regen", tmpy = "target life regen") -
This function sets the mana and life drains for the last enchantment spawned by this character. Values are * 256

SpawnCharacterXYZ = SpawnCharacterXYZ(tmpx = "x", tmpy = "y", tmpdistance = "z", tmpturn = "turn") -
This function spawns a character at a specific location, using a specific model type, failing if x,y,z is invalid NOTE: DON'T USE THIS FOR EXPORTABLE ITEMS OR CHARACTERS, AS THE MODEL SLOTS MAY VARY FROM MODULE TO MODULE...

ChangeTargetClass = ChangeTargetClass(tmpargument = "slot") -
This function changes the target character's model slot.
NOTE: DON'T USE THIS FOR EXPORTABLE ITEMS OR CHARACTERS, AS THE MODEL SLOTS MAY VARY FROM MODULE TO MODULE. USAGE: This is intended as a way to incorporate more player classes into the game...

PlayFullSound = PlayFullSound(tmpargument = "sound", tmpdistance = "frequency") -
This function plays one of the character's sounds... The sound will be heard at full volume by all players

SpawnExactChaseParticle = SpawnExactChaseParticle(tmpargument = "particle", tmpx = "x", tmpy = "y", tmpdistance = "z") -
This function spawns a particle at a specific x, y, z position, that will home in on the character's target

CreateOrder = tmpargument = CreateOrder(tmpx = "value1", tmpy = "value2", tmpargument = "order") -
This function compresses tmpx, tmpy, tmpargument (0 - 15), and the character's target into tmpargument... This new tmpargument can then be issued as an order to teammates... TranslateOrder will undo the compression

OrderSpecialID = OrderSpecialID(tmpargument = "compressed order", tmpdistance = "idsz") -
This function orders all characters with the given special IDSZ... Note that the IDSZ is set in tmpdistance...

UnkurseTargetInventory = *“UnkurseInventory(self.target)”*
This function unkurses all items held and in the pockets of the target

IfTargetIsSneaking = “*if(TargetIsSneaking()) then*” -
This function proceeds if the target is doing ACTION_WA or ACTION_DA

DropItems = DropItems() -
This function drops all of the items the character is holding

RespawnTarget = *“Respawn(self.target)”*
This function respawns the target at its current location

TargetDoActionSetFrame = *"DoActionSetFrame(self.target, tmpargument = "action")"*
This function starts the target doing the given action, and also sets the starting frame to the first of the animation (so there is no interpolation 'cause it looks awful in some circumstances) It will fail if the action is invalid

IfTargetCanSeeInvisible = "*if(TargetCanSeeInvisible()) then*" -
This function proceeds if the target can see invisible

SetTargetToNearestBlahID = SetTargetToNearestBlahID(tmpargument = "idsz", tmpdistance = "blah bits") -
This function finds the NEAREST (exact) character that fits the given parameters, failing if it finds none
TARGET_TYPE blahteam;

SetTargetToNearestEnemy = SetTargetToNearestEnemy() -
This function finds the NEAREST (exact) enemy, failing if it finds none

SetTargetToNearestFriend = SetTargetToNearestFriend() -
This function finds the NEAREST (exact) friend, failing if it finds none

SetTargetToNearestLifeform = SetTargetToNearestLifeform() -
This function finds the NEAREST (exact) friend or enemy, failing if it finds none

FlashPassage = FlashPassage(tmpargument = "passage", tmpdistance = "color") -
This function makes the given passage fully lit. Usage: For debug purposes

FindTileInPassage = *"tmpx, tmpy = FindTileInPassage(tmpargument = "passage", tmpdistance = "tile type", tmpx, tmpy)"*
This function finds all tiles of the specified type that lie within the given passage. Call multiple times to find multiple tiles. tmpx and tmpy will be set to the middle of the found tile if one is found, or both will be set to 0 if no tile is found. tmpx and tmpy are required and set on return

IfHeldInLeftHand = "*if(HeldInLeftHand()) then*" -
This function passes if another character is holding the character in its left hand. Usage: Used mostly by enchants that target the item of the other hand

NotAnItem = NotAnItem() -
This function makes the character a non-item character.
Usage: Used for spells that summon creatures

SetChildAmmo = *"SetAmmo(self.child, tmpargument = "none")"*
This function sets the ammo of the last character spawned by this character

IfHitVulnerable = "*if(HitVulnerable()) then*" -
This function proceeds if the character was hit by a weapon of its vulnerability IDSZ. For example, a werewolf gets hit by a [SILV] bullet.

IfTargetIsFlying = "*if(TargetIsFlying()) then*" -
This function proceeds if the character target is flying

IdentifyTarget = *"Identify(self.target)"*
This function reveals the target's name, ammo, and usage Proceeds if the target was unknown

BeatModule = BeatModule() -
This function displays the Module Ended message

EndModule = EndModule() -
This function presses the Escape key

DisableExport = DisableExport() -
This function turns export off

EnableExport = EnableExport() -
This function turns export on

GetTargetState = *"tmpargument = GetTargetState()" = "tmpargument = sefl.target.state"*
This function sets tmpargument to the state of the target

GetTargetContent = *"tmpargument = GetTargetContent() " = "tmpargument = sefl.target.content"*
This sets tmpargument to the current Target's content value

ClearEndMessage = ClearEndMessage() -
This function empties the end-module text buffer

AddEndMessage = AddEndMessage(tmpargument = "message") -
This function appends a message to the end-module text buffer

SetMusicPassage = SetMusicPassage(tmpargument = "passage", tmpturn = "type", tmpdistance = "repetitions") -
This function makes the given passage play music if a player enters it tmpargument is the passage to set and tmpdistance is the music track to play...

MakeCrushInvalid = MakeCrushInvalid() -
This function makes doors unable to close on this object

StopMusic = StopMusic() -
This function stops the interactive music

FlashVariable = FlashVariable(tmpargument = "amount") -
This function makes the character flash according to tmpargument

AccelerateUp = AccelerateUp(tmpargument = "acc z") -
This function makes the character accelerate up and down

FlashVariableHeight = FlashVariableHeight(tmpturn = "intensity bottom", tmpx = "bottom", tmpdistance = "intensity top", tmpy = "top") -
This function makes the character flash, feet one color, head another...

SetDamageTime = SetDamageTime(tmpargument = "time") -
This function makes the character invincible for a little while

IfStateIs8 = *"if(self.state = 8) then"*

IfStateIs9 = *"if(self.state = 9) then"*

IfStateIs10 = *"if(self.state = 10) then"*

IfStateIs11 = *"if(self.state = 11) then"*

IfStateIs12 = *"if(self.state = 12) then"*

IfStateIs13 = *"if(self.state = 13) then"*

IfStateIs14 = *"if(self.state = 14) then"*

IfStateIs15 = *"if(self.state = 15) then"*

IfTargetIsAMount = "*if(TargetIsAMount()) then*" -
This function passes if the Target is a mountable character

IfTargetIsAPlatform = "*if(TargetIsAPlatform()) then*" -
This function passes if the Target is a platform character

AddStat = AddStat() -
This function turns on an NPC's status display

DisenchantTarget = *"Disenchant(self.target)"*
This function removes all enchantments on the Target character, proceeding if there were any, failing if not

DisenchantAll = DisenchantAll() -
This function removes all enchantments in the game

SetVolumeNearestTeammate = SetVolumeNearestTeammate(tmpargument = "sound", tmpdistance = "distance") -
This function lets insects buzz correctly... The closest Team member is used to determine the overall sound level.

AddShopPassage = AddShopPassage(tmpargument = "passage") -
This function makes a passage behave as a shop area, as long as the character is alive.

TargetPayForArmor = *"tmpx, tmpy = PayForArmor(self.target, tmpargument = "skin")"*
This function costs the Target the appropriate amount of money for the given armor type. Passes if the character has enough, and fails if not. Does trade-in bonus automatically. tmpy is always set to cost of requested skin tmpx is set to amount needed after trade-in (0 for pass).

JoinEvilTeam = JoinEvilTeam() -
This function adds the character to the evil Team.

JoinNullTeam = JoinNullTeam() -
This function adds the character to the null Team.

JoinGoodTeam = JoinGoodTeam() -
This function adds the character to the good Team.

PitsKill = PitsKill() -
This function activates pit deaths for when characters fall below a certain altitude...

SetTargetToPassageID = SetTargetToPassageID(tmpargument = "passage", tmpdistance = "idsz") -
This function finds a character who is both in the passage and who has an item with the given IDSZ

MakeNameUnknown = MakeNameUnknown() -
This function makes the name of an item/character unknown. Usage: Use if you have subspawning of creatures from a book...

SpawnExactParticleEndSpawn = SpawnExactParticleEndSpawn(tmpargument = "particle", tmpturn = "state", tmpx = "x", tmpy = "y", tmpdistance = "z") -
This function spawns a particle at a specific x, y, z position. When the particle ends, a character is spawned at its final location. The character is the same type of whatever spawned the particle.

SpawnPoofSpeedSpacingDamage = SpawnPoofSpeedSpacingDamage(tmpx = "xy speed", tmpy = "xy spacing", tmpargument = "damage") -
This function makes a lovely little poof at the character's location, adjusting the xy speed and spacing and the base damage first Temporarily adjust the values for the particle type

GiveExperienceToGoodTeam = GiveExperienceToGoodTeam(tmpargument = "amount", tmpdistance = "type") -
This function gives experience to everyone on the G Team

DoNothing = ""
This function does nothing Use this for debugging or in combination with a Else function

GrogTarget = GrogTarget(tmpargument = "amount") -
This function grogs the Target for a duration equal to tmpargument

DazeTarget = DazeTarget(tmpargument = "amount") -
This function dazes the Target for a duration equal to tmpargument

EnableRespawn = EnableRespawn() -
This function turns respawn with JUMP button on

DisableRespawn = DisableRespawn() -
This function turns respawn with JUMP button off

IfHolderBlocked = "*if(HolderBlocked()) then*" -
This function passes if the holder blocked an attack

IfTargetHasNotFullMana = "*if(TargetHasNotFullMana()) then*" -
This function passes only if the Target is not at max mana and alive

EnableListenSkill = EnableListenSkill() -
This function increases sound play range by 25%

SetTargetToLastItemUsed = SetTargetToLastItemUsed() -
This sets the Target to the last item the character used

FollowLink = FollowLink(tmpargument = "index of next module name") -
Skips to the next module!

IfOperatorIsLinux = "*if(OperatorIsLinux()) then*" -
Proceeds if running on linux

IfTargetIsAWeapon = "*if(TargetIsAWeapon()) then*" -
Proceeds if the AI Target Is a melee or ranged weapon

IfSomeoneIsStealing = "*if(SomeoneIsStealing()) then*" -
This function passes if someone stolen from it's shop

IfTargetIsASpell = "*if(TargetIsASpell()) then*" -
Proceeds if the AI Target has any particle with the [IDAM] or [WDAM] expansion

IfBackstabbed = "*if(Backstabbed()) then*" -
Proceeds if HitFromBehind, target has [DISA] skill and damage dealt is physical automatically fails if character has code of conduct

GetTargetDamageType = *"tmpargument = GetDamageType(self.target)"*
This function gets the last type of damage for the Target

AddQuest = AddQuest(tmpargument = "quest idsz") -
This function adds a quest idsz set in tmpargument into the Targets quest.txt

IfBeatQuestAllPlayers = "*if(BeatQuestAllPlayers()) then*" -
This function marks a IDSZ in the targets quest.txt as beaten

IfTargetHasQuest = "*if(tmpdistance = TargetHasQuest(tmpargument = "quest idsz)) then*" -
This function proceeds if the Target has the unfinIshed quest specIfied in tmpargument and sets tmpdistance to the Quest Level of the specIfied quest.

SetQuestLevel = SetQuestLevel(tmpargument = "idsz", distance = "adjustment") -

This function modIfies the quest level for a specIfic quest IDSZ tmpargument specIfies quest idsz and tmpdistance the adjustment (which may be negative) -

AddQuestAllPlayers = AddQuestAllPlayers(tmpargument = "quest idsz") -
This function adds a quest idsz set in tmpargument into all local player's quest logs The quest level Is set to tmpdistance if the level Is not already higher or QUEST_BEATEN

AddBlipAllEnemies = AddBlipAllEnemies() -
show all enemies on the minimap who match the IDSZ given in tmpargument it show only the enemies of the AI Target

PitsFall = PitsFall(tmpx = "teleprt x", tmpy = "teleprt y", tmpdistance = "teleprt z") -
This function activates pit teleportation...

IfTargetIsOwner = "*if(TargetIsOwner()) then*" -
This function proceeds only if the Target is the character's owner

SpawnAttachedCharacter = SpawnAttachedCharacter(tmpargument = "profile", tmpx = "x", tmpy = "y", tmpdistance = "z") -
This function spawns a character defined in tmpargument to the characters AI target using the slot specified in tmpdistance (ATK_LEFT, ATK_RIGHT or INVENTORY). Fails if the inventory or grip specified is full or already in use.
NOTE: DON'T USE THIS FOR EXPORTABLE ITEMS OR CHARACTERS, AS THE MODEL SLOTS MAY VARY FROM MODULE TO MODULE...

End = "?"
This Is the last function in a script

TakePicture = TakePicture() -
This function proceeds only if the screenshot was successful

SetSpeech = SetSpeech(tmpargument = "sound") -
This function sets all of the RTS speech registers to tmpargument

SetMoveSpeech = SetMoveSpeech(tmpargument = "sound") -
This function sets the RTS move speech register to tmpargument

SetSecondMoveSpeech = SetSecondMoveSpeech(tmpargument = "sound") -
This function sets the RTS movealt speech register to tmpargument

SetAttacksSpeech = SetAttacksSpeech(tmpargument = "sound") -
This function sets the RTS attack speech register to tmpargument

SetAssistSpeech = SetAssistSpeech(tmpargument = "sound") -
This function sets the RTS assist speech register to tmpargument

SetTerrainSpeech = SetTerrainSpeech(tmpargument = "sound") -
This function sets the RTS terrain speech register to tmpargument

SetSelectSpeech = SetSelectSpeech(tmpargument = "sound") -
This function sets the RTS select speech register to tmpargument

IfOperatorIsMacintosh = "*if(OperatorIsMacintosh()) then*" -
Proceeds if the current running OS is mac

IfModuleHasIDSZ = "*if(ModuleHasIDSZ(tmpargument = "message number with module name", tmpdistance = "idsz")) then*" -

Proceeds if the specified module has the required IDSZ specified in tmpdistance The module folder name to be checked is a string from message.txt

MorphToTarget = MorphToTarget() -
This morphs the character into the target Also set size and keeps the previous AI type

GiveManaFlowToTarget = *"GiveManaFlow(self.target)"*
Permanently boost the target's mana flow

GiveManaReturnToTarget = "GiveManaReturn(self.target)"
Permanently boost the target's mana return

SetMoney = *"SetMoney(tmpargument = "amount")"*
Permanently sets the money for the character to tmpargument

IfTargetCanSeeKurses = "*if(TargetCanSeeKurses()) then*" -
Proceeds if the target can see kursed stuff.

DispelEnchantID = DispelEnchantID(tmpargument = "idsz") -
This function removes all enchants from the target who match the specified RemovedByIDSZ

KurseTarget = *"Kurse(self.target)"*
This makes the target kursed

SetChildContent = "self.child.content = tmpargument"
This function lets a character set the content of the last character it spawned last