Ares' Scripting Page

Amateurs (Beginner)

Professionals (Intermediate)

Masters (Advanced)

ARES' SCRIPTING PAGE
•CTRL.Make your men uncontrollable
MOVE.Archers forward
MOVE.Berserks forward
GEOM.Test for arrows around Rathas
<<container for arrows>>
SOUN.Rathas expresses his displeasure
MELE.Enemy attacks your position
•Name (name_string)
•Monsters (subj_monster Identifier)
Link (link_action identifier)
Activates On Success (acos_action identifier)
Deactivates On Success (deos_action identifier)
Activates On Failure (acof_action_identifier)

SCRIPTING is the language by which MAP ACTIONS are coded into a MESH file. The practical upshot of scripting is that it makes the map interesting! Meshes are dead. Scripting brings them to life. Almost nothing on the map can happen without Scripting.

SCRIPTING IS EASY

You just have to understand the components, and then you can combine those components in ways the authors of the scripting language never foresaw, to make almost literally anything you want happen on the battlefield.

These tutorials cover SINGLE PLAYER SCRIPTING. If you're a beginner, I will start with a background of how to create a single player level in the first place, so that you can script it. Some knowledge of FEAR and AMBER will be necessary.

First, you have to have your colormap. You know, the big thing with all the pretty colors arranged in such a manner as to have some semblance of a landscape. Make sure it is in the right format as dictated by the guidelines of RTFM*. Then, you open Loathing, select "New," and, when prompted, you select your colormap. Then you use all the various little buttons and tools and things to make the map look like you want.

Very good. Now you should have two tags in your LOCAL folder, one in the COLLECTIONS sub-folder, and one in the MESHES sub-folder. Open FEAR and then open the STRING LISTS folder. Create a new STRING LIST. Type the name of your level. Then, in Fear, go into Meshes. Open your mesh file. Attach the string list you made in the pop-up menu that says "Map Description."

Now go back into Loathing. Open the Monster Palette, and add monsters to your map. Player units are team 0, enemy units are team 1, mutual enemy units are team 2, ambient life is team -1.

Now you're ready to script.

Click on the flag icon. This brings up a list of Map Actions. This list should be blank.

When you click "NEW" you will create a new action. This action will have a name and a type. That's all you need to worry about.

Creation a new action of type "none."

Now mess around with the "custom" button just to see what you can do. Try to create the following

monster_identifier subj - This will allow you to create a list of the subjects of the actions, the units that do the action

monster_identifier obje - This will allow you to create a list of the objects of the actions, the units that the action is done to.

monster_identifier enem - This will allow you to create a list of monsters that will be tested for in proximity tests (covered later)

You have no idea what the hell you're doing, do you?

Well, no matter - continue to Intermediate level

NOTE - Intermediate is nothing more than a straight cut and paste of my scripting tutorial that has been floating around on the net for years. It has been updated slightly and a few sections moved to Advanced. If you want to skip it, go straight to advanced.

*Read The F---ing Manual

Ares' Scripting Page

Professionals

TABLE OF CONTENTS

ARES' GUIDE TO SINGLE PLAYER SCRIPTING

So, you want to make a single player level, eh? I assume you are already familiar with how to make a map - inserting a color map, placing units, that sort of thing. This is a scripting guide, not a map-making guide :) Make sure you have your map all ready to go, with all the units you want placed in advance. You can change the location of a unit by control clicking, so don't worry about making it perfect now. Make sure all the player units are team 0 and the enemy units are team 1.

Forgive me if I oversimplify this, and repeat myself, but I want to be as clear as possible. This is mainly for people who don't even know how to begin scripting, but it may also help people with a little scripting knowledge. You'll note I stop being so meticulous as the document advances and stop giving examples for everything, because I assume the reader will begin to be more familiar with how scripting works.

First, have everything you want to do mapped out in your head. If you want to add or delete units later, you'll have some trouble realigning the scripting. Got it all ready? Good. Lets rock! The fictional map I will use as an example is a map similar in layout to "willow creek." The player has warriors, bowmen, and dwarves. There is an initial thrall army you must fight, an army at the stream, at the road, and in the town.

Click on the little flag icon tool to bring up a list of scripted actions. There are three boxes, the top is the box that lists the scripted actions. The second lists the components of a selected action, and the third lists the components of one of those components. Sound complicated? Its really not. Just do everything I say :)

Here is generally what happens: You create an action by clicking "add" in the first box. Depending on what type of action you create, different options will be available for you in the second box. The third box specifies where what or whom to affect. As an example, the first box might have a movement action. When you select this, the second box allows you to add strings for the general words "monsters" (to move), "waypoints" (locations to move to), "final facing" (in degrees, the direction for them to face at the end) and "formation" (the formation they move in.) Clicking on "monsters" will give you a list of the monsters to select in the third box, clicking on "waypoints" will give you the X-Y coordinates to move to in the third box, etc, etc.

Get the general idea? I'll get into specifics later.

You'll get the hang of it after less than two hours of scripting, I guarantee it.

LISTING UNITS ON THE MAP

The first thing you have to do is organize your scripting so it flows better. When you create a new action, it has a name and a type (from the pull down menu.) If you select type "none," and do nothing else, then what you have is simply an organizational piece of text that in no way affects gameplay - it could even be a space to seperate groups of related actions, or a secret message for someone who opens your map and pokes around.

The first thing you want to do is create one of these organizational texts. Type in big caps for the title of the action "PLAYER FORCES." Hit OK. That's it, you now have a piece of text in your action list that doesnt do anything. (NOTE: It is not necessary to have these organizational texts, but it makes the scripting read much easier. I really suggest you do it.)

This action is NOT organizational text, because you are about to do something to it. Make sure it is selected, and under the SECOND box, click "add custom." From the pull-down menu, select to add a monster-identifier. You are going to identify your player forces wth this action, so later on, if you want your men to do something, you can do so very easily.

Now you have two lines of text in the second box, these are

•Name (name,string)

monster_identifier subj

Click on "monster_identifier subj" and in the world view, click once on each unit the player controls to select them. As you click, a list of this pattern will start to appear in the third box:

#1 Warrior
#2 Warrior
#3 Warrior
Etc, etc

Now you have created an action that identifies all your men. It may turn out that you never use these links, as player forces are not always scripted to do things in the map - sometimes, the player issues all the commands, and the A.I doesn't for the entire level - but you should still have them for aesthetic purposes, and if you want to change something to include a scripted action for them later on.

Now you have to do the same thing for the enemy units. This is very important, as ALL the things the enemy does is scripted by the A.I. After the last player unit is listed, create a couple blank lines for organizational purposes, and a little organizational divider, and do the same thing for all the enemy units. Say the enemy consists of ghols, soulless, and thrall.

For this, things are a little different. You dont divide the creatures by type, but by group AND by type. Say you have a group of enemies in a town, another by a stream, etc, as I noted above. You want to group these guys as follows:

Instead of lumping all the units together, divide them into very specific groups, each with an action of its own, like "thrall at stream," "souls at stream," "thrall at town," etc.

Now, after you create a list of all the enemy units, you should have something like this in the first, main box:

PLAYER FORCES

ENEMY FORCES
Initial Thrall Army
Stream thrall
Stream souls
Town thrall

etc.

After you have this all set up, DUPLICATE the player forces action, and name it PLAYER FORCES AS TARGETS

Now where you see "monster identifier subj" double click and change "subj" to "obje." Now THESE actions will be the "object" of other actions, while the first were the subjects. I'll get into that later. Do the same thing AGAIN and name the action "PLAYER FORCES AS ENEMIES." Go through and change from subj or obje to "enem." You'll learn why later.

INITIAL OBSERVER MOVEMENTS

The first thing we will learn is an initial observer movement - this is when the camera moves by itself at the beginning of the level. To make an observer movement action, create a new action of type "Observer Movement." Lets say now that you want the camera at the beginning of the level to show a scene somewhere on the map, and you want it to move to show your forces. Click "initially active" and give the action a trigger time of around 2.00 to show the scene for two seconds before the camera starts moving. Now, the camera will move as soon as the level starts, but will wait two seconds before doing so. In the secondary box, you can add a list of strings that include the destination, final facing, and speed of the camera (among other things. You also MUST include a string that says the camera is controled by map actions. Thats prety self explanatory when you add it. You should have something that looks like this:

•OBMO.<whatever you named this action>

Whenever you see a • next to an action, that means it is initially active.

When you have an observer movement at the very beginning of the level, the camera starts out at the starting location for the Team 0 observer and moves to the destination. If you script an observer movement to happen during gameplay, the camera will move from wherever the player has the camera at the time to the destination, but for an initial observer movement, it just starts out wherever you put that eyeball icon.

Add a destination string in the second box, and, with "destination" selected, click on a place in the world view. The X-Y coordinates of where you clicked will appear in the third box. The camera will now start out at the team 0 observer, stay there for two seconds, and move to this place!

But you probably want to specify the direction for your camera to face. In the second box, add a "Final facing" string. Now, click and drag anywhere in the world view, and an arrow appears. Drag the arrow until it is facing the way you want, and let go. The degree measure will automatically appear in the third box.

The default observer camera speed of 0.0625 is very slow. To speed it up, add a "Linear Interpolant Seperation" string, and click "add" under the third box to specify the speed. I recommend at least a speed of 0.30, or as
high as 1.00 for a fast observer.

Congratulations! You just scripted your first action. Save your map and go try it out.

THE TUNI ACTION - YOUR BEST FRIEND

Now we move on to perhaps the most important action in Loathing: The Test Unit action, hereby refered to as a TUNI action. The TUNI action tests a unit or group of units, and if their state is the same as is specified by the action, it can trigger anything else to happen. TUNI actions are always initially active in simple maps unless there are special circumstances. In more complex maps, some of the units are not initially present on the level, and you cant test for them until they appear.

Lets say you have a large force of thrall at the start of the level, the group you identified as the Initial Thrall Army. If you dont script anything, your archers can kill them all and the thrall will just stand there. But with the TUNI action in conjunction with an attack (ATTA) action, you can script all the Thrall to attack as soon as one of them takes a little damage.

First create the TUNI action. Name it "Initial Thrall Damaged?" Add a "link" string in the second box. In the third box, have this link be "Initial Thrall Army" from the list of enemies you made in the very beginning. The TUNI will now test these units. (When you click "add" in the third box with a link selected in the second, you get a pull down menu that lets you link to any of the other actions you created)

Now add a "Vitality Less Than" string in the second box, and in the third box, make this 1.00

For some reason, it shows up in the third box as 1.000001 most of the time, but this makes no difference.

Now, this action will test the initial thrall army to see if their cumulative vitality is less than 100%

Create an "activates on success" string. This means that if the TUNI action sucessfully tests the initial thrall army to be less than 100%, it will activate all the actions in the third box under this "activates on success" string. The action we want to activate is for the Thrall to attack.

Except you haven't scripted that yet.

Create a new scripted action, of type "Attack." Change it so it deactivates "never." Name it "Initial Thrall Attack." Add a link in the second box, and link it to the Initial Thrall Army in the units list you created in the very beginning. These are the units that are going to be doing the attacking. You could also add a "Monsters" string in the second box, and individually click on each monster you want to have attack you, but its a lot easier to simply make a link. Now in that same link, add a second action - "PLAYER FORCES AS TARGETS." The subject of the action (attackers) will now attack the object of the action (attackees)

Now you have scripted the Thrall to attack your army, and by default, this is not initially active and happens on a trigger. What is the trigger?

The TUNI action. Under the "activates on success" string you made for the TUNI action, add the attacking action you just made into the third box. When the A.I. successfully tests for the initial thrall army to have less than 100% cumulative vitality, it will execute the thrall attack action. You should now have a set of actions in the main list that looks like this:

•TUNI.Initial Thrall Damaged?
ATTA.Initial Thrall Attack.

Note that this looks misleading, it seems to imply that the fact that these two actions are congruous, one after the other, means that one triggers the other. We gave these actions the names we did and put them right next to each other for organizational purposes, a TUNI action can trigger an action anywhere else on your list.

Congratulations! You just scripted the Thrall to attack when first damaged.

You did it wrong.

Sorry about that but I just wanted to show you the ATTA action. It is what you use for RANGED units. For MELEE units you need a slightly more complex action - the MELE action. Change the ATTA action you just made to a Melee or MELE action. You have the ability to add a new element - "Power over distance threshhold." In an ATTA, the enemy attacks the most powerful units. This is great for, say, soulless, but bad for melee units - they'll go for your dwarves while your warriors hack away at their backs. With melee you can make the Thrall give priority to close units that are within the number you specify in the power over distance threshhold. Make this number two.

You can also have an army attack as soon as the level starts by simply making an ATTA action and making it be initially active, or you can make a bunch of units attack after a set amount of time by clicking initially active and setting the trigger time.

You can also use the TUNI action to test for other things to get them to attack. Another way is with a radius. You can test for if any enemy units (units identified as monster_identifier enem) are within a certain radius, and if so, you can put an activates on success link to get them to attack. Do this with the string flag "enemy closer than radius" and a number you specify. A typical TUNI - ATTA/MELE combo is like this:

•TUNI.Damage test
link (link_action identifier)
Initial Thrall Army
Vitality Less Than
0.99000
Activates On Success
MELE.Thrall attack
Deactivates on success
TUNI.Proximity test

•TUNI.Proximity test
link (link_action adentifier)
Initial thrall army
PLAYER FORCES AS ENEMIES
Enemy closer than radius
15.00
Activates On Success
MELE.Thrall attack
Deactivates On Success
TUNI.Damage test

MELE.Thrall attack
link (link_action identifier)
Initial thrall army
PLAYER FORCES AS TARGETS
Power over Distance Threshhold
2.00

You can test any unit or group for a change in vitality, whether or not they are dead, and any number of things, and link any action you want to a TUNI action. You can link movement, observer movements, and attack actions among other things.

Or you can link an endgame action. Say you want the level to end with a Dark victory when all the player units are dead. Create a TUNI action, link all the player forces to it, and test for unit count equal to zero. Then link an ENDG (endgame) action to it with an "activates on success" string. You can do the same for the Dark forces to create a Light victory. Example:

•TUNI.Player forces dead?
link (link_action identifier)
PLAYER FORCES
Unit count equal to
0
Activates on success
ENDG.Dark victory

ENDG.Dark victory
Dark victory flag

ACTION-TRIGGERS-ACTION

TUNI actions usually trigger an action to occur, but a chain reaction can also be done. When any action is successfully completed, you can put an "activates on success" string to start another action. For example, a certain army getting killed my Myrkridia (TUNI) can trigger a warrior to go to a certain destination. When this MOVE action is executed, an OBMO action can be triggered that moves the observer to show this warrior. When the OBMO is completed, the sound file "Those Beasts... They're Unstoppable... We haven't a prayer!" from level nine can be triggered to play with a SOUN action. When this is successful, the observer moves back to your army with another OBMO action. You can also have a MOVE action trigger another MOVE action to give the appearance of a patrol making its way around the map, trigered to attack if you get too close. This "chaibn reaction" type of thing is a very useful tool for experienced map-makers

RE-INFORCEMENTS FOR EITHER SIDE

To make units appear as re-inforcements or as another wave of enemies that enters the map, put a bunch of units all clumped together at the very very edge of the map, or in a corner. Make them all invisible with the unit tool. You can then use any action that, upon success, triggers these men to appear via a "visible flag" under the action type Unit Control (CTRL) When they appear, you can make that trigger them to move in formation to a location.

OTHER CTRL ACTIONS

You can trigger units to teleport into a world knot with a CTRL action. Just place the world knot pylon models and put invisible units in there. When one action is completed, it can trigger the units to beam in via the world knot visibility flag. Note that for all these actions, you must specify which units to afect with a link to the original units list or a "monsters" flag.

Very simple CTRL actions include setting a unit's name to a specific name, setting their health level, making them veterans, and giving them kills. These can all be done with initially active CTRL actions with the appropriate strings attached.

OTHER USEFUL ACTIONS

The other actions which I haven't given their own section are Attack (ATTA) and Movement (MOVE) actions. These actions let you make enemy units attack or move around as necessary. They are generally not initially active, and are triggered by other actions. They are both pretty intuitive. For an ATTA action, you only really need two links - the list of creatures that will be doing the attacking, and the list of targets (normally all the player
forces.) When triggered, the attackers will charge the specified targets and attack them

Movement actions are just as intuitive. You need to specify what monsters to move, add a destination and click in the world view, add a final facing screen and click-and-drag in the world view, and add a formation index string. Then trigger the movement to occur. You can even have multiple waypoints and have the monsters cycle them.

Sound actions are mainly used for in-game conversations. Bungie has a wide variety of things that units say, and you can mix 'n match to create unique conversations. Add a SOUN (Sound Action) action. There should be a string in the second box that looks like so:

•Sound Tag (soun, sound)

Click on this and click add in the third box. You get a list of all the Bungie sounds you can play, many of which have subtitle which play automatically. You can use an activates on success string to link this sound to play when any action successfully occurs. To make a conversation, have the successful completetion of one sound trigger another sound to play. In a converstion, its best to give soun tage a small, two or three second triger time to have pauses between the phrases, and not make it sound like they are interrupting each other.

LEGION

Legion should mainly be used for soulless on hills defending things. Make a subj list for soulless. For group count, say "1." For line, put a point on the hill and a point right in front of the hill. For group centerpoint array, put a point on the hill. For group facing array, click and drag towards the direction player units will come from. Add a formation with Group Formation Array. if you have other nearby soulless legions you can add an Allied Legions string and attach the other LEGI actions. Very simple and very useful.

SPECIALIZED ACTIONS

There are special actions for Ghols and Wights - Suicide and Harass. Both are very self explanatory in Loathing - just experiment with the values a bit to customize these actions to your maps. Suicide makes wights sneaky, so they try to run away from archers and run at your back. Harass makes ghols pick up detritus, try to lure your dwarves and archers away, and run at their backs. Neat stuff.

MUNGER

Munger is the action that Bungie says you "should never, EVER use." Its OK to use Munger to delete an action. Thats fine. If there is an action that is giving you trouble later on aftr its usefulness is finished and you cant figure it out, simply delete it with a Munger after its is used.

Dont do anything else. Even if you think you understand it and you've studied the examples. Trust me. I know. In the words of Nathan "It seemed to be the Munger itself that caused random, destructive violence on people's machines." Ever seen a house model tip over onto its side and slide horizontally across the mesh for no reason and then you suffer a complete system meltdown and your machine crashes and you have to unplug it? I have. Dont do advanced Mungering.

ORGANIZING YOUR SCRIPT

The Command key (On Macs) is extremely useful for indenting and changing the order of scripts. Just select a script and use command + the arrow keys to indent or move the script up or down in the list.

CONCLUSION

I hope this guide has helped you learn the basics of scripting. Once you know these things, you can figure out lots of other cool stuff to do. Happy scripting! If you have any questions about this guide or any problems you encounter while scripting, or want to do something more complicated than what I've described, I'll do what I can to help.

CONTINUE to ADVANCED

Ares' Scripting Page

Masters

Here's where I impart my knowledge of Advanced Scripting.

Actions covered here are ACLI, GEOM, MOMA, and DELA among others.

TABLE OF CONTENTS

- Basic explanation

• SECRETS of LOATHING
- monster_identifier targ
- flag onii
- world_point_2d pnts
- others?

ACLI

Action Lists can activate several actions at once. "So what? You can do that with multiple links in the acos!" Well, it is often easier to just have ONE action in the acos, and make sure all the actions you want to happen are in the ACLI. It is less confusing.

But there is another merit of Action Lists. This is Randomization.

One of the abilities of the ACLI is something called the Random Selection flag. When this is used, only ONE of the actions in the ACLI will be activated. One, randomly.

Aside from making this randomize two actions, it can also randomize order. Say waves of attacks are activated by CTRL actions. If you have two waves, you can duplicate the CTRL's and give the second group trigger times. Then make an ACLI that activates the first wave and the delayed second wave. Make another ACLI that activates the second wave and the delayed first wave. Make a third ACLI that randomly chooses between these two ACLI's. BAM, you've got yourself a randomized attack order. The same principle can be used to randomize any number of waves.

SQUADS, PLATOONS, AND PLATOON MOVEMENTS

Squads, Platoons, and Platoon Movements are GREAT actions that no mapmapker can do without. They allow patrols or groups to move around the map in a formation or combination of formations (even a smiley face with the circle, box, and deep arc formations), and attack the player (although other triggers are also usually used.)

Squads are groups of units. These groups can be specified to move in a certain formation, and in any relation to each other. Platoons are groups of squads. Platoon movements are the route the platoons take.

In a squad, first add a link to a subj list of the units you want to be in the squad. There are several other things you have to specify - the formation via its corresponding number (0 = short line, 1 = long line, 2 = loose line, etc.) and the centerpoint. Dont put anything for facing. Listen closely: The

PLATOON will also have centerpoint when we get to that. The centerpoint of the SQUAD in relation to the centerpoint of the PLATOON determines where it walks. If you want Fetch to walk in front of the thrall, have the Fetch squad's centerpoint be in front of the centerpoint for the thrall squad. if you want the Soulless to go to the left of the thrall, have the centerpoint for the Soulless squad be to the left of the centerpoint of the thrall squad. So you could have any number of elaborate formation combinations in a platoon - a circle of thrall, a couple of boxes of soulless, a deep arc of myrmidons to have a smiley face of enemys advancing on you =)

The PLATOON has four things you must specify: Under Initial State link the Platoon Movement you are about to make. Under Initial Squads list all your squads. Under Centerpoint put a point in the middle of the men somewhere - its the squad's centerpoints that matter, this could theoretically be on the other end of the map. For facing, just click and drag an arrow in the world view the direction they will initially be facing although this doesnt really matter.

For PLATOON MOVEMENT, click on a route you want the units to follow across the map. Put a small radius of 2 for Waypoint Radii to add a little randomness. For "Radii" put two values, both the same: 20 and 20. These dont really matter as they wont be what triggers your PLAT to attack if you do what I'm about to tell you. Add a loop flag if you want them to follow a route. If your waypoints are far apart and you have different types of units with different speeds, you may want to interpolate the waypoints - this means that if you add an intrpolate flag and interpolate spacing of, say, 5, then every 5 World Units the fast guys will stop and wait for the rest to catch up. Very useful. Now GO BACK AND ATTACH THIS ACTION TO THE PLATOON. Dont forget like I always do, its damn annoying!

Now, add some damage and proximity TUNI tests like I specified under TUNI and you have a patrol that goes around the map and attacks!

NOTE - If you want the platoon to abandon the chase when units lead it too far away from its route, DONT use damage and proximity tests! Rely on the "Radius" and make the radius big enough - say 28 or so - such that archers cannot attack them without being attacked.

GEOM actions (Geometry Filters)

These are some of the most complicated actions in Myth. Listen VERY closely. Geometries are used to filter out certain units and place these units into an empty "container action" as subj's, obje's, or anything else (which is way too complicated to even get into. Subj and obje are what is used the vast majority of the time.)

You can then trigger another action to occur once some units are in this container action and which applies only to them.

Say you want the enemy to attack ONLY your units which go into a certain area. Obviously, TUNI will not work. You can test for units in that area. But the enemy will not know which guys to attack. First you need a subject of the GEOM actions. These are the units the GEOM looks for. Put a subj list that includes all your guys ("PLAYER FORCES"). Now you can either have a "circle centerpoint," "circle centerpoint (monster)," and "circle radius," or you can have "polygon points," and a "polygon closed flag" to specify the area within which the geometry tests. For the circle, select a point or monster as the centerpoint, and an appropriate radius. For the polygon, simply put points on the map under the "polygon points."

Now the GEOM will filter out any of the PLAYER FORCES that stray into this area. Now add the string "tested items inside field name." Put "obje" because you want these guys to be the object (targets) of the attack.

Now make your container action. Name it <<<player unit container>>>. Thats it. Now go back to the GEOM and add the string "Results action identifier." Select the container you just made.

Now make and ATTA (or MELE) action. Add a link. Put the subject list for the attackers and in the same link, the unit container which has the object list of the units that go into the forbidden zone. Go back to the GEOM. Add an "Activates on success" string. Attach the ATTA (or MELE) you just made.

Make the GEOM initially active.

Congrats!

Other things a Geom can be used for include making units turn "blue" (comp controlled) on the overhead when they enter a certain area and do stuff under comp control (think of the final exit of the Myth1 level "Flight from Covenant" or when a dwarf gets close to a magic hole thingy with a glowing stone in "The Summoner"). Do this by linking a CTRL to the GEOM instead of an ATTA and then have the CTRL trigger whatever it is you want them to do.

Another example of a VERY ADVANCED Geom is on the level "Shiver" and is duplicated in the upcoming CoD level "Raiders of the Ermine." It is actually used to have Shiver target satchel charges with her explosive spell and blow them up! This is done with neat effects like having the tested items inside field name be "wayp" instead of obje or subj, meaning an attack ground location for an attack, having the circle centerpoint be herself so she looks for satchels around her, and looking for object type "dwsc" the code for satchel charge, instead of looking for units. Check it out in the Shiver scripting! There are limitless things you can do with GEOM if you have enough imagination.

MORE GEOM

Geom can do even better things. It can test for projectiles around a unit, as was shown with the last paragraph of the above. The thing is, it even works with very small radii. So you can actually tell what projectile caused damage. This is very significant! It means you can actually tell which type of unit did damage or attacked another unit. If you want the enemy to retreat when arrows are shot at him, but not to retreat if attacked by a different projectile, this is the way to do it. The radius of testing can be very small, even as small as 0.2, to ensure greater accuracy.

There is also a string called 'Tested Items Inside Location Field Name.' This is used to put a world_point_2d into a container action instead of a monster_ or object_identifier, which is what 'Tested Items Inside Field Name' does. If you want to place the location of a unit or object in a container, this is your man! World_point_2d's are the numbers that appear when you click on the mesh in the world move to put points for a MOVE action, for a SQUAD or PLATOON MOVEMENT, and for a POLYGON. Using this string allows you to look up the correct 4-letter index for that particular world_point, like wayp (for waypoints). You can feed the central point for a radial test, the waypoints for a movement, the location for a platoon to move to, or even the points for a polyon into its action. So if you want a platoon to "hunt" you and follow you around, use the containter of a GEOM with this string as the location to move to in the PLMO.

Continuous feed GEOMS can be made very simply by changing the GEOM so it deactivates Never. Experiment - its not as easy as it sounds.

MOMA

MOMA will move an invisible unit to a location. The only thing you need to know that you can't figure out at this stage is that the unit must have been visible at some time before it is moved. So make the unit visible, and have an initially-active CTRL make it invisible. Then its marker can be moved. Use this for teleportation effects, or other instances where units disappear and are moved to a specific location. Also use it to move units to a location searched for by a GEOM and then teleport in. This is what Soulblighter does on "A Murder of Crows."

DELA

Cool action. It feeds the subjects to the results action one at a time with the "even distribution" flag over the time specified. So if you want units to attack one at a time, like exploding deer so they aren't all clumped, use this. You can also feed it to a GENE or something to make units in a line do the wave.

SECRETS OF LOATHING

There is a secret monster_identifier type, called monster_identifier targ, that can be used in the MOMA. When used, the subj of the action will replace the targ of the action. What does this mean? Well, use a CTRL that is triggered by the MOMA to delete the original unit, and you can have the subj, an invisible unit somewhere off the map, take on all the characteristics of the targ! THE MONSTER TAGS ARE SWAPPED IN-GAME! Wow, no? What does this mean?

Well, it means that a unit can be changed into a new unit with different attributes, maintaining the same name, the same flavor, the same health level, the same experience, and the same facing of the original - it is a seamless switch. Let me give a couple examples:

A "Borg" unit that becomes resistant to whatever hits it. When hit with a certain projectile,, say a molotov doing "explosive" damage, tested for by a GEOM projectile test, a new unit resistant to that type of damage is targed over onto the original seamlessly. Now, explosives will hardly hurt it. Also triggered will be a new set of GEOM tests for damage types that will trigger the targing of a unit resistant to explosives AND the new type! So keep hitting it in order with everything you've got!

A unit that mounts a horse or a wolf or something. As long as you have the animations for the subj and the targ all in one collection, you can have units go up to horses, and targ into a horseback rider, CTRLing away the old units. A "mounting" animation with the "entrance projectile group" in Fear can make it seamless. You can also have the death of a mounted unit NOT result in death for the rider - like when you test for the unit being at 0.2 health, the new unmounted unit is targed onto the old mounted unit, and the exit projectile group for the old unit contains the horse's death and dead body. Now you've got an unmounted unit with the same name, health, facing, experience etc. of the old.

A unit that can switch weapons and switch his entire set of animations - when he picks up something (tested for with a GEOM), the unit using that weapon is targed on.

Cool beans, no?

There is a secret flag in the CTRL action called "onii." Create a custom onii flag and units will flicker like ghosts.

There is a world_point_2d type called "pnts." If you use this as the "location field name" for a GEOM, you can make the points that a unit or object is on impassible.