Introduction of Ice XP

What is Ice ?

- A Starcraft Iscript.bin editor,

This program hows you to edit how Starcraft treats the various graphic files throughout the game. Advanced users can use this knowledge to change the speed of units,

allow building graphics to be used for units and vice versa.

Advised for usage only by those with experience with the other more common SC editors (Arsenal, StarDraft, etc.).

The edited .bin files must be moved into a "script" folder either in a MPQ file.

What is Ice XP?

- Basically.. Ice XP is modified layout and charaters's color from origianl Ice for **Windows XP user** and **Ice Beginners**

When if you use original Ice, the character's color in text field might displayed same to background field color so you might hacked on that clocked characters



To work out that trobles, i've modify these stuffs

- Ice layout codes and some background color of objects are modified to White color

- Program layout size is be widen to seen very comfortable, especially GRP preview windows size

- Added some hint text and description

Actually in this Ice XP is no any big different functions except changed Layout and Colors

Take look these Screenshots

Modified background field color to White color

🐚 ICE XP – modified	d by NJ								
<u>F</u> ile <u>O</u> ptions <u>H</u> elp									
Main Headers GRP Prev	iew SC Sound hea	ring							
Header of Unit/etc:									
Protoss Nexus									
<u>C</u> reate New Header <u>D</u> elete Header <u>R</u> ename Header									
Iscript ID : Search by Iscript ID									
Header Info Entry Heade	 Animation Scripts 	1							
Animation Typ	e 21 Edit	Return to basic pose from Ground Attack:	0						
Initial Animation	12948	Return to basic pose from Air Attack:	0						
Death Animation	31107	Spell and Special Ability							
Initial Ground Attack:	0	Walking Animation:	0						
Initial Air Attack:	0	Return to basic pose from Walking:	0						
change unit call/effect:	0	Special Ability?:	0						
State burrowed:	0	Repeated Ground Attack:	0						
Construct and mind :	0	Repeated Air Attack:	0						
Research tech/upgrade:	12959	Special Ability?:	0						
Land:	0	Liff off:	0						
Unknown 6:	31162	Unknown 7:	31162						
Unknown 8:	12967	Unknown 9:	0						
Unknown 10:	0	Unknown 11:	0						
Burrow:	0	Unburrow:	0						
Unknown 14:	0	*hint: black 0 :no animation data/ gray 0	: none entered						

🍓 New Opcode	e	
Opcode:		-
Play frame Play frame UNKNOW Shift grand	e %1 e %1 according to tileset. May require a LOS file /N OP: 0x02 (Assuming 1 params: 2 bytes): %1 bic position %1 vertical units	
Registered S Wait %1 ti Wait eithe Go to offs	VN OP: 0x04 (Assuming no params) icks er %1 or %2 ticks (random) et %1	~
🐚 Edit Type		
Animation Number:		
	1 Animation [0] 2 Animations [1] 4 Animations [2] 14 Animations [12] 14 Animations [13] 16 Animations [14] 16 Animations [15] 22 Animations [20]	

GRP preview window is widened

🐚 ICE XP – modified	d by NJ 📃 🗖 🔀
<u>F</u> ile <u>O</u> ptions <u>H</u> elp	
Main Headers GRP Previ	view SC Sound hearing
Main GRP ID#:	Frame No#0 Secondary GRP ID# : ID Tag: 275 Frame No#0
Header Info	Animation Scripts
Registered Scripts:	Initial Animation
Place active underlay Play frame 0 Go to offset 31162	Initial Animation Death Animation Initial Ground Attack Animation Initial Air Attack Animation Special Ability Animation 1? Repeat Ground Attack Animation Repeat Air Attack Animation Special Ability Animation 2?
New Opcode Copy Delete Cut Note: To paste a line you may click twice [Paste	Paste Up Jump Hex Down Back Current Offset Eind ! Current Offset Eind ! Selected line Offset.7456

🐚 ICE XP - modified by NJ
<u>File Options Help</u>
Main Headers GRP Preview SC Sound hearing
Main GRP ID#: Frame No#0 Secondary GRP ID# : ID Tag: 275 Frame No#0
▲ 240 view Image C Sprite
Header Info Entry Header Animation Scripts
Registered Scripts: Initial Ground Attack Animation
Play frame 34 Wait 1 ticks Begin unbreakable code section Play sound (sfxdata.dat) 69 Attack with 1 (1 = Ground, else Air) Play frame 51 Wait 1 ticks Play frame 34 Wait 1 ticks Play frame 51 Wait 1 ticks
New Opcode Copy Paste Up Jump Delete Cut Hex Down Back Note: To paste a line you may click twice [Paste] at first time for copying Search Offset Eind !

Reference: ICE written by DI

Of all the file utilities discussed so far ICE is probably the most powerful (this is excluding the Stardraft and Stargraft patch loaders of course, since without them none of this would be possible :). While ICE is designed to be user friendly (much, much more user friendly than if you had to edit the iscript.bin manually, which, believe it or not, is what a lot of people did until now), it is still critical to understand the basic structure of the iscript.bin before learning how to use it. It will make it a lot easier to understand, and help you diagnose bugs (with your scripts and with ICE itself). With that said, I will reluctantly tell you that you *probably* won't have to worry about much of the pointers (goto line XXX, etc.) that were ranted about in the last section. Pointers are still important and will still cause many crashes if they aren't handled right, but ICE tries to handle them for you. For example, you normally can't just insert an opcode in the middle of some animation routine because it will throw off every pointer that is below it. However, in ICE you normally *can* just insert (add) an opcode in the middle of a script. This is because when you do this, ICE will automatically move the script you are editing to the end of the iscript, and adjust the appropriate script pointer in the header to match.

File Menu

Open File - This will let you open an iscript.bin file for editing.

Load Default - This will load the default iscript.bin from the MPQ for editing (it will not edit the iscript.bin that is in the MPQ specifically, it will just extract it and you can edit it as a new file).

Load from CWAD - This will load the iscript.bin from inside a specific CWAD file. Note that it must be in scripts\iscript.bin in your CWAD.

Save File - Self explanatory.

Save File As - Self explanatory.

Save File to CWAD - This will import the current iscript.bin file into a specific CWAD with the path scripts\iscript.bin (the appropriate one for use by SC). This will save you the trouble of opening up the CWAD Manager all the time.

Create CWAD from File - This will create a new CWAD with your iscript.bin inside (with the path scripts\iscript.bin).

Main Headers

This tab on top will allow you to select which animation header you want to edit (or create a new one or delete one). Recall that an animation header is the set of line offsets which determine which individual animation for that animation set is located. ICE gives "names" for the headers indicating which sprite the header was originally associated with. Don't let the names confuse you though; they really don't mean a thing. (You can associate any animation header with any images.dat sprite)

The drop-down menu will allow you to select one of the current headers, and thus load the header's data (pointers and animation routines into the editing tabs on the bottom).

Create New Header - This will allow you to add a new header (set of animation routines) to the iscript.bin.

Script Name: This will make ICE associate a name with this animation header. Its just for helping you remember what this header is for. It doesn't effect anything.

Offset: This will tell ICE where to put this animation header in the iscript.bin ("line," or offset). By default ICE will add the header to the end so it won't mess up any pointers. YOU SHOULD ONLY PLACE THE HEADER IN ANOTHER LOCATION IF YOU KNOW WHAT YOU'RE DOING.

ID Tag: This is the Iscript ID number that is associated with this set of animations. For example, if you want a certain images.dat entry to associate with this animation routine, you would make this ID Tag and the Iscript ID variable in the images.dat the same number. If the ID number you choose is already associated with an images.dat entry (in the original images.dat in the MPQ), then ICE will allow you to "Preview" the GRP file which is associated with that images.dat entry.

Type: Each set of animations in the iscript is of a certain "type." The type of header determines how many different animations there are in the set. Most "normal" terran and protoss man sprite (non-building) have type 12 or 13. Most zerg man sprites have type 26 (extra animations routine needed for burrowing). Most buildings have 20 (zerg) or 26 or 27 (terran, protoss; need more animation for lift off and "building is working" animations). Most "overlay" type sprites (sprites that aren't used for units but just played in conjunction on top of or below to other sprites have type 0 or 1 (they only need an initial initial and death animations). If you're not sure what type to use, just go with the largest number of animations. Which animation is which doesn't change between types (e.g., the initial animation will always be the first animation listed in type 0, 13, or 27), but you don't always need all of them.

Delete Header - Self explanatory.

Rename Header - Self explanatory.

GRP Preview and WAV Preview

These are peripheral tabs that are designed to help you edit the iscript by allowing you to preview unit graphics and sounds, but have absolutely nothing to do with the actual iscript you are editing. In other words, you could ignore them if you wanted and there wouldn't be any difference, and messing with them doesn't effect your iscript. The GRP Preview tab will allow you to preview the GRP which is originally associated with this animation set (in the Main GRP window). How it does this is ICE looks up what original images.dat entry is currently associated with this animation header's ID tag and then looks up what GRP is associated with that images.dat entry. Keep in mind that no GRP is actually directly associated with the animation set, it is only indirectly connected by association with the images.dat entry which contains a pointer to this iscript routine. The Secondary GRP window allows you to preview another GRP (associated with a certain images.dat entry, or a certain sprites.dat entry -- which is really just a sprites.dat entry pointing to an images.dat entry which then points to a GRP). The purpose of this is to allow you to preview overlay or underlay graphics that are played by the iscript on the current main sprite. This is done by double clicking on certain opcodes which will be explained further later. The buttons between the two windows allow you to choose a specific GRP to preview in the Secondary GRP window. You just enter the number of the images.dat or sprites.dat entry, check the appropriate radio button, and click preview. ICE will show the GRP originally associated with that entry.

The WAV Preview is similar. While there are no "default" sounds associated with sprites, iscript animation routines are allowed to play sounds (i.e., the clash of psi blades when the zealot sprite attacks). When you double click on certain opcodes which play sounds (by association with an sound entry in sfxdata.dat), ICE will put the associated sound in this window. You can double click on it or press "Play WAV" to preview the actual sound. You can add more WAVs to the preview list by inputing an entry number from sfxdata.dat (the sounds data file, each entry points to a certain sound, listed in sfxdata.tbl -- we'll get to this later, but its not that complicated). ICE will add the associated wav file to the list for previewing. Similarly you can remove wavs from this list with the "Remove WAV" button.

Reference: ICE Part II

Now that we've learned how to navigate around the basics of ICE, its time to start modifying some headers and actual animation routines. First, you need to select or create a new header to edit.

Header Info

There's not much to edit here. This is just basic info that associates the header with an ID number (so images.dat can use it).

ID Tag: This is the Iscript ID tag associated with this header. If you didn't read the description I gave with the "Create New Header" section, here it is again: This is the Iscript ID number that is associated with this set of animations. For example, if you want a certain images.dat entry to associate with this animation routine, you would make this ID Tag and the Iscript ID variable in the images.dat the same number.

Offset: This determines the line offset where the header is actually located in the iscript (the header is the list of pointers to the actual animation routines). You should not mess with it unless you know what you're doing.

Entry Header

This is the actual header of the animation set. If you recall, the header is basically a list of pointers to the location of each individual animation routine in this set. In general, I advise against editing any of these pointers unless you are very sure of what you are doing. The reason you don't have to edit locations to routines manually through this section is because ICE will do it automatically for you when you create/modify the actual animation routines (next section). You don't want to accidentally make a pointer to the middle of nowhere (that will crash SC :), so its best to let ICE handle it.

You can change the "Type" of header this is. If you didn't read the description of set types before, here it is again: The type of header determines how many different animations there are in the set.

Most "normal" terran and protoss man sprite (non-building) have type 12 or 13. Most zerg man sprites have type 26 (extra animations routine needed for burrowing). Most buildings have 20 (zerg) or 26 or 27 (terran, protoss; need more animation for lift off and "building is working" animations). Most "overlay" type sprites (sprites that aren't used for units but just played in conjunction on top of or below to other sprites have type 0 or 1 (they only need an initial initial and death animations). If you're not sure what type to use, just go with the largest number of animations. Which animation is which doesn't change between types (e.g., the initial animation will always be the first animation listed in type 0, 13, or 27), but you don't always need all of them.

Animation Scripts

Here's where the heart of the iscript lies. Each entry in the drop-down menu will bring you to the actual opcodes of an animation routine, and allow you to edit that animation (for the current header set).

Animation [drop-down menu]: Select the animation routine of this set that you wish to edit. The opcodes will appear in the black box below. Here is a brief description of what each animation is for:

Initial animation - This is the animation that plays when the sprite is spawned or created. Normally, this animation also spawns any active overlay/underlay graphics for the main sprite (e.g., a shadow sprite or a turret). Also, in many cases, the code for this animation contains the code for the normal "idle animation" of the sprite (if it's a unit). This is because right after a unit is spawned, it goes into its idle state. However, do not mistake this entire animation for the idle stance, because it is not. What usually happens is that at the end of this routine, there will be a goto opcode that points to a line in the middle of this routine (the beginning part will usually spawn a shadow, so it will simply loop back to the part after the opcode which spawned the shadow, since it doesn't need to do that more than once).

Death Animation - This is the animation that is played when the sprite is destroyed (e.g., when the unit using this sprite is killed). Usually this will end with a "End animation, remove graphic" opcode, signaling that the sprite should be removed (otherwise it will linger on the battle field forever, though just as a graphic).

Initial Ground Attack Animation - This is the animation to play for this sprite when the unit it is associated with begins to attack a ground unit.

Initial Air Attack Animation - This is the animation to play for this sprite when the unit it is associated with begins to attack an air unit.

Special Ability Animation 1? - Unknown.

Repeated Ground Attack Animation - This is the animation that plays for a sprite when it continues to attack a ground unit. (I.E., when it begins to attack it will play the initial attack animation, but if the target is not dead yet, then it will continue attacking with this animation)

Repeated Air Attack Animation - This is the animation that plays for a sprite when it continues to attack an air unit. (I.E., when it begins to attack it will play the initial attack animation, but if the

target is not dead yet, then it will continue attacking with this animation)

Special Ability Animation 2? - This is the normal "cast spell" animation. E.G., the one to play when a High Templar casts Psi Storm or a Ghost casts Lock Down. Some spells do not require an animation and some "special" spells use a different one (but most use this one).

Return to Idle from Ground Attack - This is the animation that plays when a sprite is finished attacking its ground target (usually just a pointer back to the middle of the initial animation, where the idle stance starts).

Return to Idle from Air Attack - This is the animation that plays when a sprite is finished attacking its air target (usually just a pointer back to the middle of the initial animation, where the idle stance starts).

Special Ability Animation 3? - Unknown.

Walking Animation - This is the animation to play when the unit this sprite is associated with moves from one point to another. Note that if you have the flingy.dat Move Control setting set to Flingy Control (as opposed to iscript control) then any "move graphic XX units forward" opcodes in this animation will be ignored, though the rest of this animation will still play as the unit moves (but speed will be controlled by flingy.dat). If you have the Move Control set to iscript control then it will ignore speed settings in flingy.dat and use the "move graphic XX units forward" opcodes in this animation.

Return to Idle from Walking - This is the animation that plays when a sprite is finished walking (flying, gliding, or whatever) to its destination (usually just a pointer back to the middle of the initial animation, where the idle stance starts).

Other Animation? or Generic Animation? - Sort of Unknown. This is usually the "special" animation that is associated with very specific unit actions (it varies depending on what unit this sprite is associated with). For example, this is used for "suicide" attacks (like for the infested terran and scourge sprites).

The remaining animation routines are kind of unknown, but here's what I've picked up about some of them:

Unknown 1 - Is burrowed/in "other" state animation. (e.g., initialize burrowed units' animations if placed as burrowed in staredit since they never got to start burrowing in the first place)

Unknown 2 - Construction Animation (buildings, during construction); Mining Animation (Peons, when harvesting minerals).

Unknown 3 - "Building is working" (training unit, researching upgrade, etc.) Animation.

Unknown 4 - Landing Animation (for terran intransit buildings).

Unknown 5 - Lift off Animation (for terran intransit buildings).

Unknown 12 - Burrowing Animation.

Unknown 13 - Unburrowing Animation.

(Note that this tutorial may be older than the current version of ICE and some of these Unknowns may already be labeled, or shifted around)

Animation Opcodes [the black box =]: When you select an animation routine to edit, its opcodes will show up in this box. Recall that an animation routine is just a list of commands or "opcodes" that tell a sprite what to do (what frames to play, when to play a sound, when to wait, etc.).

You MUST end all animation routines with either a "Go to Offset XXXX" opcode or a "End animation, remove graphic" opcode (the latter usually only for death animations, but can be used elsewhere). ICE will warn you if you try to mess up otherwise. Some routines will say "No Animation Data" meaning there is no routine there yet. You can create a routine there by adding new opcodes using the commands at the bottom.

The trickiest part about writing animation routines has to be getting the "Go to Offset XXXX" pointing to the correct locations. Since the XXXX is just an offset number with no other indication of where it relates, it can be pretty confusing. Usually, you want this to lead to some other location in the current header set (e.g., after an attack animation, you probably want it to loop back to the begging so it can attack again, or after the initial animation animation finishes, you probably want it to loop back to the start of the "idle" stance part). The easiest way to do this is to go to the actual opcode you want it to loop to first (find it in the animation routines and select that line). ICE will display the offset to that opcode in the lower right hand corner as "Op Offset." Remember that number and then go back to your original Go to Offset opcode you were editing and paste that number there. If you're unsure of where to go to, a normally safe bet (if you are using the original iscript) is to go to offset 31162. At that location is a script that does nothing but loops itself, so you're sprite will just halt.

To change the values of an opcode (the numbers), just highlight them (they should turn green). Drag your cursor over them and input your new value and press enter. Here's a brief description of what all the buttons on the bottom do:

New - Insert a new opcode on top of the currently selected one. It will bring up a list of opcodes to choose from. Remember that you don't have to worry about throwing off pointers by inserting opcodes in ICE because when you add an opcode, ICE will automatically move your script to the end of the iscript file. (You can verify this by looking at the Script Offset number in the bottom right hand corner and see that it changes when you add an opcode)

Delete - Delete the currently selected opcode.

Copy - Copy the currently selected opcode.

Cut - "Cut out" the currently selected opcode. (Same as Copy and then Delete)

Paste - Paste the opcode last copied or cut on top of the currently selected one.

Hex - View a hex dump of the current animation routine. Ignore this unless you're researching opcodes like me. :)

Up - Move the currently selected opcode up.

Down - Move the currently selected opcode down.

Jump - If the currently selected opcode is a "Go to Offset XXXX" opcode, then jump to offset XXXX (and display those opcodes in the black box for editing).

Back - Move back to an animation routine if you jumped to XXXX (with the previous option). You can jump and go back multiple steps.

Goto Offset - Goto an arbitrary offset you specify and display the opcodes at that location for editing. Usually only useful if you know where a certain animation routine starts so you can enter it. (You don't want to go to the middle of no where)

And that's basically it. Now you know how to edit animation routines. Though you're probably still wondering specifically what some of those opcodes do. :) Here's some brief descriptions and examples: <u>Iscript Opcodes</u>.

Now wasn't that fun? We're almost done with all the animation stuff. One more section to go

Basic ICE Tutorial written by ChauSara

ICE is what makes up attack animations. If you downloaded the Doom Dragoon mod at the top, you will see that the Dragoon shoots multiple cannons. This is done with ICE, and you will learn how to do this in this tutorial. (NOTE: This was made by BSTRhino)

You will need: Arsenal III or Datedit, ICE, WinMPQ, MPQDraft (the MPQ Compactor is optional.) You can download all these programs from www.starcraft.org

So, first thing to do is open up ICE. Now, you want to start a new file, so go to 'File > Load Default.' This will load StarCraft's real iscript.bin - the one that hasn't been modified yet.

In the drop down menu in the top half, select Protoss Dragoon (you can type it in to make the searching faster.)

ICE is split into two halves. The top half is where you select things and preview things. The bottom half is where you edit their code.

Now, just for fun, go to the top half and click on the second tab (I can't remember what it's called.) Now you can see your Dragoon in action. You can even pull on the scrollbar under it to see all the frames it has.

Okay, back to business. Select the third tab down the bottom (I can't remember what that was called either.) Now, in the bottom half there will be a drop-down list thing. Since you've just opened the Dragoon, it will say 'Initial Animation.' Go down that list and select 'Initial Ground Attack.'

Just below that drop-down list containing animations will be a list of all the code. It's quite self-explanatory if you ask me. Somewhere there, will be a 'attack with 1' line. Copy that. Then paste that same line one on top of the other so you get a five lines saying 'attack with 1.' ICE has a bug, when you click paste for the first time, it will do nothing, so keep clicking paste until you get five of those lines.

Now, that's not all. You've changed the ground animation, but because of the

way ICE works, the air animation remains the same. (That doesn't happen in IceCC.) What you have to do is go to the second tab in the bottom half. Somewhere there it will say 'Initial Ground Attack Animation' with a box with some number (Thirty thousand something) next to it. Copy that. Paste that in the 'Initial Air Attack Animation' box.

Next, do the same thing to repeated ground and air attacks so it does it always.

That's the first part. Now the dragoon will shoot five bullets every time it attacks, no matter ground or air.

Save your iscript file to scripts\iscript.bin

The next bit you should already know how to do. If you don't, then you shouldn't be reading this tutorial and go learn some basics about modding.

Go into the weapons editor and change the dragoon's weapon (phase disruptor) so its movement type says 'Halo (spin.)' Just like the valkyrie rockets, the dragoon's weapon will scatter.

Not quite finished yet. Finally, give the phase disruptor splash damage. In the 'explosion type' box, select 'line splash' and then in the inner, middle and outer boxes, type in 32, 32, and 32. If you want to know what those numbers mean, that's another tutorial.

So then all you need to do is save your arr\weapons.dat file, make the patch, and you're all set.

I will be updating with more tutorials soon. Tell me how you thought this was!

*Friends must have the same mod.

1. basic knowhow for before the beginning first , running Ice and start load custom isript.bin files or load Default file on File menus of Ice then u can start scripting now~!

select unit in Header taps or typing units id number what u want change thing and click to search by header btn then u can see unit images in GRP prieview Tap

u can see each offset numbers of headers in Entry Header tap that number is what header used that scripts contents so if u change offset number then that header action's scripts will change to that numbers script

2. way to change and goto offset ok, i will explain to an example is "Terran marines" in this section

goto "Animation Scripts" tap and are there many animaitons headers that animations exist each offset numbers u can see find that numbers on "script offset" line that position of layout is lower of right side

ok first , choose "Initial Animation" in Animations drop down list and then showed operation code sripts in center contents box each code line has own offset number too like animation headers can find that number on "op Offset:" line

let's see the "Terran marine's Initial Animation"

<u>CODE</u>

place active underlay [images.dat] 240 at vertical offset 0 play frame 68 wait either 63 or 75 ticks [random] using randomizer value[?] 25 , jump to offset 28964 using randomizer value[?] 128 , jump to offset 28942 Go to offset 28928

u can find three offset codes in here that offset codes deferent operation to eachother but both action will be goto other offset check offset codes in scripts contents

"jump to offset 28964" -> this will go to other animations at randomly , ramdom time is values number -> "value[?] 25 "

and if u want find that offset number where are going ?

typing that number "28964" in "Goto offset" input box and click "goto" btn then u can see changed srcipt contents and find changed number "28964" on "scripts offset" line

that mean is srcipt offset goto one after the other codes must exist where are going next offset code or finishing operation codes

let's see other offset codes at last line

"Go to offset 28928" -> this code mean is where are going next animation when this script action finished

and u can go that offset directly, select that code line and just double click then u can jump to that offset "Go to offset" operation always can jump by that ways

if u want change jump offset ? then select offset number area and click one time, then number color be changes other color and u can edit that number

Remember, offset number is linked animations contents so u may write number without error in animation (this mean is that unit must be can running all operation codes with changed offset)

see "Go to offset 28928" again, and double click that line and jump to that offset then scripts change showed that deleted upper codes some lines as compared with Initial Animation

that is "Other" header in IceCC (when unit has not doing anything action for a while)

if u want change that scripts codes then u edit codes ur self firsthand or otherway is goto "Entry Header" tap and change Initial Animation section's number as other unit's "Initial Animation" offset number such as zealots etc...

the way is click [...] like this button and popup "Set Animation windows" select units header as u want things and select that's animations too , click to [ok] button then "Initial Animation" offset number

be changed and u can see changed operations codes in srcipts contents box of Animation Scripts Taps

Remember again, u may write offset number without error , this is very important about crash game

if changed operation code be can not running for Terran Marines (such as "unable play frames number" or written wrong id numbers)

3. randomizing functions in scripts

this section is about how can add new operation codes

goto Animation Scripts tap and u can see a lot of buttons, input box and contents area form

1 . Animation [.....] (dropdown list) this is showed about unit what has animation headers

recorded animations is linked to situation of "Entry Header" setting if each Entry Header item be registered then showed that animations in Animation dropdown list

2. scripts contents form box this box is showed what header written a operation codes and u can edit the value each opcodes, copy paste and delete code too

3.command buttons there are many command buttons , this button will creat new code and command other operation

----buttons

[New] - > this is write new operation code in Scripts lines

click that and popup "New Opcode" windows and select opcodes as u wanna write things and click to [ok] then created line and new code in Scripts contents form

[copy] -> copy to selected line codes

[paste] -> paste codes (there are some bug in ice u may click twowise at first time)

[up] -> selected codes is move up one line

[jump] -> scripts header jump to selected offset codes

[Delete] -> Delete code line

[Cut] -> same to Del

[Hex] -> Hex edit

[Down] -> selected codes is move down one line

[Back] -> go to offset ago

4.Offset search area in lower of right side , u can see search offset area

Goto Offset [] [Goto] -> this is search offset written numbers Script Offset : -> this is showed current Animation header's offset number Op Offset : this is showed current operation code's offset number

Detail of [New] button jump to popup "New Opcode" windows

if u want write new operation code like "play frame 17" then u may click [New] btn

and choose that opcode in opcoed drop-down list and change frame number value

All operation codes are be writed like that ways

i can't explain now about each opcodes because u can find and search out that opcodes meaning with original scripts of other units

ok, this tutorial is not rub a person the right way and my english not good i know that plz understand me ~!

anyway i hope this tut help to beginner Ice

<u>Changing Unit Speeds In ICE written by Stwong from Staredit</u> <u>Network</u>

What this example/tutorial intends to do is make the Protoss Dragoon move four times as fast, teaching the subject how frames work, what is possible speed-wise in ICE, and how to do most of the animation editing with a little bit of experimentation.

1. Open your favorite MPQ viewer, and extract ISCRIPT.BIN from a copy of >patch_rt.mpq (can be found in your Starcraft directory). Do not load the default one from ICE, since it is outdated and does not contain all of the updates Blizzard has made over the years.

Notes:

* Never use originals when modding. Originals are bad. Make copies, then mod them.

2. Pop open the extracted ISCRIPT.BIN in ICE.

3. Since we are using the Protoss Dragoon as our example, click on the Main Headers tab on the top tablist, and find Protoss Dragoon in the combo box labeled Header.

4. The Protoss Dragoon is now open for editing... we will want to edit his animation script to change either the delay between taking steps or the scale of the steps he takes. On the bottom tablist, hit Animation Scripts to open the script editor.

5. Since speed is located in the walking animation, click Walking Animation under the combo box labeled Animation.

6. In the black box below the combo box, you should see something to the effect of: <insert picture of default dragoon walking animation> If this is not seen, turn back now and recheck your steps.

7. Since the dragoon moves every time his frame changes, it will be useless to remove the delays, except to remove the animation and make him appear to hover. (This will NOT, however, render the speed uneditable.) We instead will change the scale of the steps he takes, by changing the "Move graphic x units forward" instruction to an increased or decreased speed. To make the changes VERY obvious, we will make the dragoon move four times as fast. With that speed, he will clear the map in a few bounds.

* By frames, we mean graphic changes. Units whose graphics do not change, yet still move, use Animation ID 31162 for movement, with an introduction of listing a bit off of the floor. A frame change instruction is symbolized by a "Play frame x" instruction, and this instruction is directly linked to the frames in the GRP. You can use the GRP viewer to see what a frame is ahead of time. Aside from death frames, all frames have a full hemisphere of movement, and rotate all the way around on the right side, meaning that you only have to select the beginning of the set of frames, instead of writing instructions for each direction that the unit is facing.

8. To change the value of x in the instruction, simply click the text, delete the current value, enter your new value, and hit enter. If you wish to cancel, click somewhere else, deselecting it. Note that pressing enter is the only way to save changes to instruction values. Multiply each "Move graphic x units forward" instruction by 4.

9. Save your changes, by hitting File>Save File.

10. Create a a new MPQ called MOD.MPQ or a similar name and save the modified ISCRIPT.BIN into it, under the SCRIPTS\ path. Use MPQDraft to use this modified PATCH_RT.MPQ, build a dragoon in a lang map, and watch it run like... well... a really fast dragoon.

<u>Changing Unit Heights In ICE</u> written by Stwong from <u>Staredit</u> <u>Network</u>

What this example/tutorial intends to do is make the Protoss Probe move four times as fast, teaching the subject how frames work, what is possible height-wise in ICE, and how to do most of the animation editing with a little bit of experimentation.

1. Open your favorite MPQ viewer, and extract iscript.bin from a copy of patch_rt.mpq (copied from your Starcraft directory). Do not load the default one from ICE, since it is outdated and will remove replay functionality.

Notes:

* Never use originals when modding. Originals are bad. Make copies, then mod them. * A method I've used to overcome the fact that Arsenal III files are outdated is by patching the files in Arsenal III's directory manually. Beware, though, that if you fail to do this properly, you may need to reinstall Arsenal III to get it working again.

2. Pop open the extracted iscript.bin in ICE.

3. Since we are using the Protoss Probe as our example, click on the Main Headers tab on the top tablist, and find Protoss Probe in the combo box labeled Header.

4. The Protoss Probe is now open for editing... we will want to edit his animation script to change the height at which it animates. On the bottom tablist, hit Animation Scripts to open the script editor.

5. Since we can place height anywhere, click Initial Animation under the combo box labeled Animation.

6. In the black box below the combo box, you should see something to the effect of: <insert picture of default probe initial animation> If this is not seen, turn back now and recheck your steps.

7. Click the New button below the animation editor.

8. Select Opcode Shift graphics position %1 vertical units.

9. Select that opcode, and shift it down. Click the 0 and set it to -20. This will shift it *UP* 20 units. (It works in reverse, negative is up, positive is down.)

10. Save your changes, by hitting File>Save File.

11. Create a a new MPQ called MOD.MPQ or a similar name and save the modified iscript.bin into it, under the scripts\ path. Use MPQDraft to use this modified patch_rt.mpq, build a dragoon in a lang map, and watch it run like... well... a really fast dragoon.

If you have questions, visit us on the forums, in the chat rooms, or e-mail me. This tutorial has been tested with Starcraft: Brood War 1.09b.

Shot missiles sereval times like Terran Valkiry written by NJ

🐚 ICE v2.03	R									
<u>File</u> Options	<u>H</u> elp									
Main Headers	GRP Pre	view WA	V Preview	1						
Main GRP:			Frame:	0		Secondary GRP:	ID Tag:		Frame:	
100					Prev • Image • Sprite					
						•			•	
Animation: Play frame Mait 1 tick Play frame Wait 1 tick Play frame Wait 1 tick Play frame	:17 10 10 11 (1 = Gr 0 17 17 17 12	Initial G	round Attao	:k Animati	on					
New	Сору	Paste	Up	Jump		Goto Offset:		<u>G</u> oto		
<u>D</u> elete	Cu <u>t</u>	<u>H</u> ex	D <u>o</u> wn	<u>B</u> ack		Script Offset: 1	1153		1	
		S2 - 50				Op Offset: 1	1158			

first off, select "Initial Ground Attack Animation" in Animation at Animations Scripts "Initial Ground Attack Animation" is first Attack order To copy that line, select line and press [Copy] btn and press [Paste] btn 2wise at first time then now you will see pasted another line same as first one and set weapon behavior target (1 is ground, 2 is Air)

Paste again as you pleases times In the Same way but you may need to use "Wait ticks" and "Play frame" to Animated naturally

And write Script to other related to Attack Animations the writing target are these "Air Attack animation" "Repeat Ground Attack Animation" "Repeat Air"

And save iscript.bin and Enjoy~!

Cast Spell Sererval times

🐚 ICE v2.03	R							
<u>File</u> Options	<u>H</u> elp							
Main Headers	GRP Prev	view WA	V Preview					
Main GRP:			Frame:	0		Secondary GRP:	ID Tag:	Frame:
								_
p - 1	1				Prev			
1-21					Image C Covito			
					spine	•		
United at 1.1	·	Animat	ion Sorinte	1		الساليني		
Header Inro E	ntry Heade	er Anima	ion scripts	1				1
Animation:		Special	Ability Anin	nation 2?				-
Wait 1 tick	(S	1						
Cast Spell Wait 1 tick	s							
Cast Spell								
Cast Spell								
Wait 1 tick Go to offse	ks ∋t 31162							
New	Сору	Paste	Up	Jump		Goto Offset:		Goto
<u>D</u> elete	Cu <u>t</u>	<u>H</u> ex	D <u>o</u> wn	<u>B</u> ack		Script Offset: 38	886 —	
						Op Offset: 38	894	

For Example is BattleCruser

Select "Special Ability Animation" in Aniamiton Scripts first time for select there are 2 different orders from other lines delete their 2 weird lines, and add these blowed order

- " wait 1 ticks "
- " cast spell "

copy and paste 2 lines as you pleases times

that mean is that will be Cast BattleCruse's Speical

Ability(default:YamatoGun) by use mana at once

Append Graphics and Sprite to unit/weapns

Example : Add Defensive Matrix Sprite on Zealot

1.Select "Protoss Zealot" in Header of Main header

2.Go to GRP Preview Tab

3.Select "Initial Animation" in Animation Scripts (Tip:"Initial Animation" is Unit's Basic pose)

4.Press [New...] and appears New Opcode

5.Select "Place active **overlay** image"

6 and type "317" in images.dat value (No# 317 is Defensive Matrix image.dat number)

7.As same ways, add "Place active **underlay** image.." value is "374

🐚 ICE v2.03	8								
<u>File Options</u>	<u>H</u> elp								
Main Headers	GRP Preview W/	V Preview							
Main GRP:		Frame:	0		Secondary GRP:	ID Tag: 300	Frame:	0	
a de la compañía de l Referencia de la compañía de la compa			▲ ▼	371 <u>Prev</u> • Image • Sprite	S				
			F					•	
Header Info E	ntry Header Anima	tion Scripts	1						
Animation: Initial Animation									
Place activ Place activ Place activ Place activ Ray frame Go to offse	e underlay (images a e underlay (images a e overlay (images da 05 t 31162	dat) 152 at v dat) 374 at v at) 371 at ve	ertical off retical offs rtical offs	set 0 et 0					
<u>N</u> ew	<u>Copy</u> Paste	Up	Jump		Goto Offset:	Got	0		
<u>D</u> elete	Cu <u>t</u> Hex	D <u>o</u> wn	<u>B</u> ack		Script Offset: 420)89			
					Op Offset: 420	94			

Save iscript.bin file and Enjoy it~!