TOD TUTORIAL

This tutorial assumes that you know how to use 3D Studio Max. 2.0 or 3D Studio 4.0 and have at least achieved the basics on the net Yaroze such as displaying a 3D object.

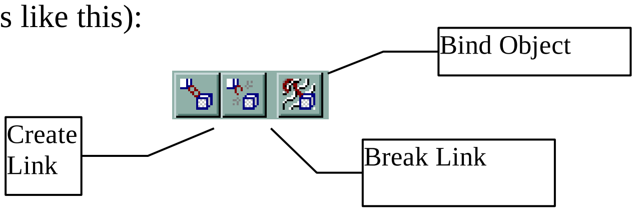
There are many ways to achieve animation on the Yaroze, but for detailed animation it is probably more practical to create these animation's on some form of 3D software package and then implement them straight into your program. This is possible if we use something called a TOD file which is a file format invented by Sony that just contains animation data.

Creating an animation on 3D Studio Max.

The following section describes a few things that you should consider when designing and creating animations to be used with the TOD files on the Yaroze.

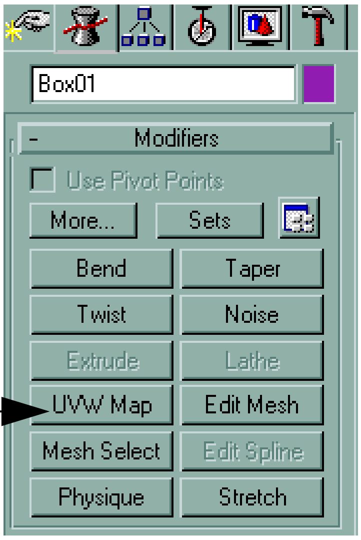
When creating your animation you must remember that you can only use forward kinematics, that is you can only link your object together using the link feature of 'Max'. You can not use inverse kinematics or free form deformations, although it is possible to use squash and stretch to some extend.

To create a linked model in 3D Studio Max you must connect the separate objects using the link option. The link button is located on the long toolbar at the top of the screen (,the button looks like this):



The create link button, will create a link between two objects. For example you may wish to create a link between a left foot object and a left leg object. To do this you would first click on the create link button. You would then move the cursor over the 'foot' object, at this point the cursor will change to indicate that you can select this object. Then click the left button and hold it down while you drag the cursor to the 'left leg' object where the cursor should again change to indicate this object is selectable, let go of the mouse button.
Now when you move the leg the foot should also move.

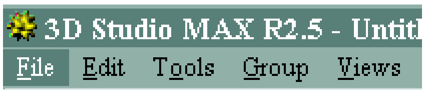
When texturing your models remember to apply a UVW mapping to any textured objects. Otherwise your texture will not appear on the model at all. To do this, select the 'UVW map' button from the modify menu. You may have to play with the options of this modify to get the required mapping effect as there is any number of options, such as Spherical wrap and box wrap to name a few.



Select this button from the Modify menu.

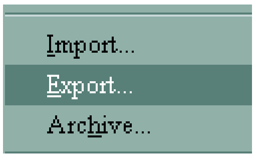
If you are not sure if something is going to convert to a TOD file, a good way of testing its compatibility is to export the animation to .3DS and then immediately import the 3DS file replacing the whole scene. Then if when you play back your animation everything looks fine, it will most likely convert to TOD safely. The following describes this procedure in detail:

1. Select File from the main pull down menu, located at the top of the screen.



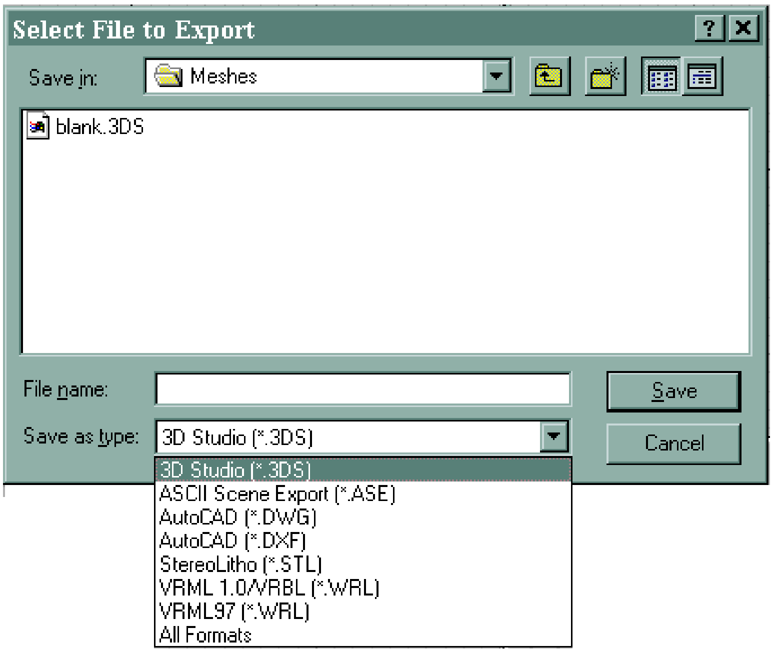
The main pull down menu.

2. From this menu choose ‘Export’.



Export

3. A dialog box is displayed that allows you to create a file name for your animation. Make sure you ‘save as type’ 3D Studio (*.3DS).



Select File to Export

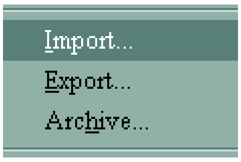
Save in: Meshes

blank.3DS

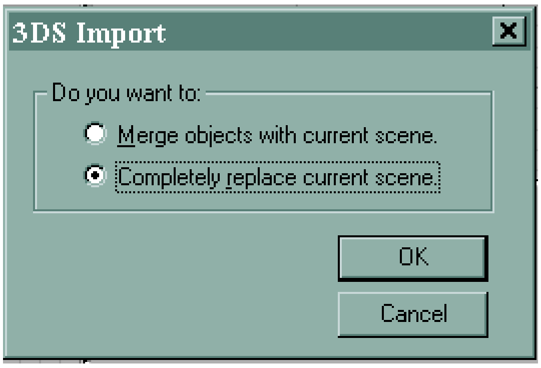
File name:

Save as type: 3D Studio (*.3DS)

4. Now select File from the main pull down menu again. Then choose Import



5. After selecting the file you wish to open, the following dialog box is presented. Choose to completely replace current scene.



3DS Import

Do you want to:

○ Merge objects with current scene.

● Completely replace current scene.

6. Play your animation. If everything is fine then you shouldn't have any problems converting this animation to a TOD.

It is possible to have more then one TOD file associated with a model. When using more than on TOD you must use completely different animation files making sure that the model you use is exactly the same in each. Remember this means you only have to create one TMD for all the TOD files. See example programs to find out more about using more than one TOD with a single model, as this tutorial will not explain how to do this in detail. Although this isn't anymore difficult to achieve than using a single TOD file.

Converting the MAX file to a TOD file

Converting your animation is rather a long winded process, but luckily enough it isn’t too difficult.

Because at the present you can only convert 3D studio 4.0 animation’s to the TOD format you must convert everything to 3D studio 4.0. So the first step is to export (explained earlier) your animation from 3D Studio Max. to a .3DS file which is the native format of 3D Studio 4.0. Then open this file in Studio 4.0 and press F4 followed by F12. Select TOD from the popup window.

Making sure the RST and OFF is selected click on save TOD to convert your animation to a TOD file. To create a compatible TMD you must now select save RSD. When asked if you would like the RSD files to be automatically named select yes.

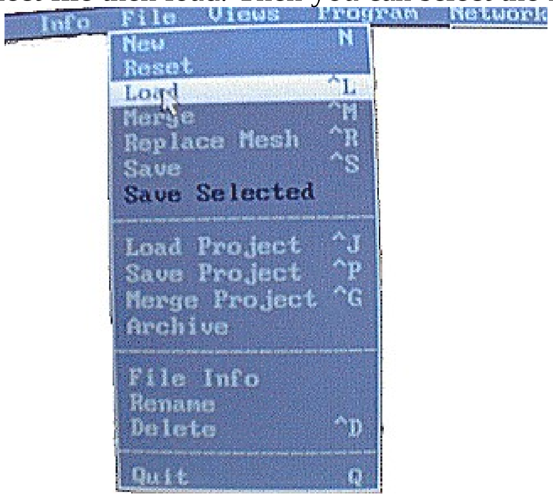
After exiting 3D studio 4.0, at the DOS prompt change to the directory that the RSD files are in usually ‘c:\psx\rsd\2 and then type ‘edit 000.rsd’. You should now be in the DOS edit program with the file 000.rsd open. Delete all the comment lines, they start with a hash ‘#’, so that you are left with just a list of RSD file names. Save as model.arg and exit. Now at the prompt type ‘rsdlink -v -o model.tmd model.arg’. If the objects are textures you must have a copy of the TIM files in the same directory as the RSD files.

Summary of steps

1. From within 3D Studio Max. use the File pull down, select Export then choose .3DS.

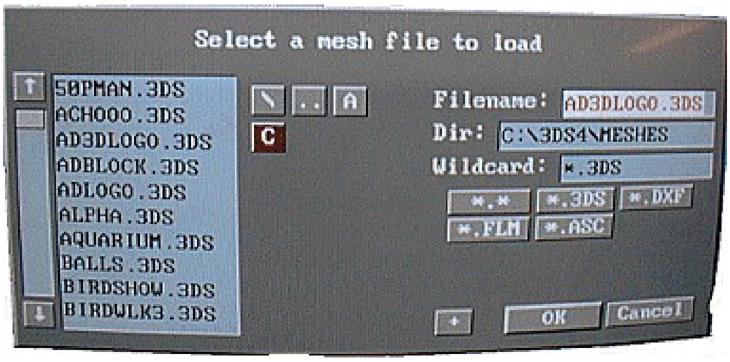
2. Copy your 3DS file into the 3D Studio 4.0 scenes directory (usually C:\3ds4\scenes\ ).

3. Open 3D Studio 4.0 and load your 3DS file. From the pull down menu at the top of the screen select file then load. Then you can select the file to load.



3D Studio 4.0 selecting to load a file.

If your file is in a different directory to that shown in the box marker ‘Dir:’ Then You can click on this box and type in the directory of your file. Once you have your file selected click on the OK button.



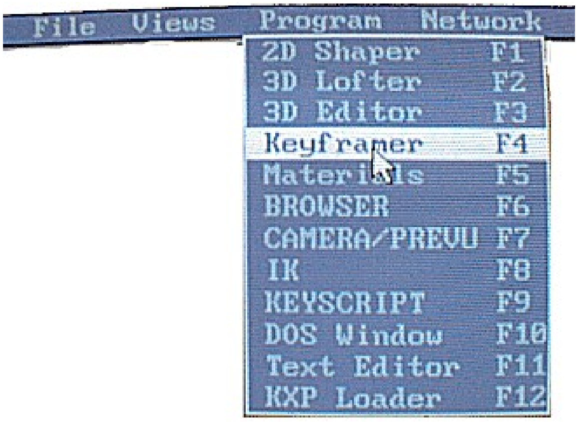
Select a mesh file to load

58PMAN.3DS
ACHOOO.3DS
AD3DLOGO.3DS
ADBLOCK.3DS
ADLOGO.3DS
ALPHA.3DS
AQUARIUM.3DS
BALLS.3DS
BIRDSHOW.3DS
BIRDWLK3.3DS

Filename: AD3DLOGO.3DS
Dir: C:\3DS4\MESHES
Wildcard: *.3DS

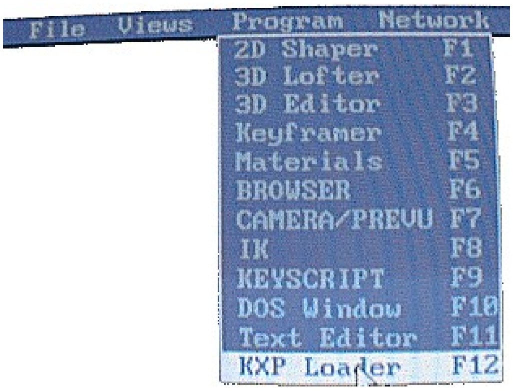
Selecting a file to load.

4. Press F4 to get to the Key Framer Screen. You can also get to the key Framer screen by selecting Program from the pulldown menu at the top of the screen followed by choosing keyFramer from the menu. [Note. you can preview your animation at this screen].



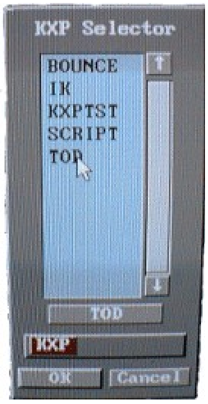
Selecting ‘keyFramer’ from the ‘Program’ menu.

5. Press F12 to open the KXP plugin window and select TOD, then click OK. Again you can also open the plugin window by selecting the ‘KXP Loader’ form the ‘Program’ pull down menu.



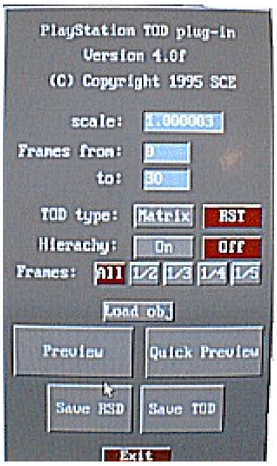
Selecting the KXP Loader.

You are then presented with the KXP Selector menu. Now you must select TOD and click the OK button.



KXP Selector

6. You are then presented with the TOD conversion screen.



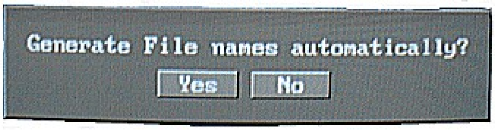
PlayStation TOD plug-in
Version 4.0f
(C) Copyright 1995 SCE

scale: 1.000003
Frames from: 0
to: 30
TOD type: Matrix [✓ RST]
Hierachy: On [✓ Off]
Frames: [✓ All] 1/2 1/3 1/4 1/5

the PlayStation TOD plug-in.

7. You must make sure that RST and OFF are selected and that the correct scale and frames range has been set. As shown in fig ????

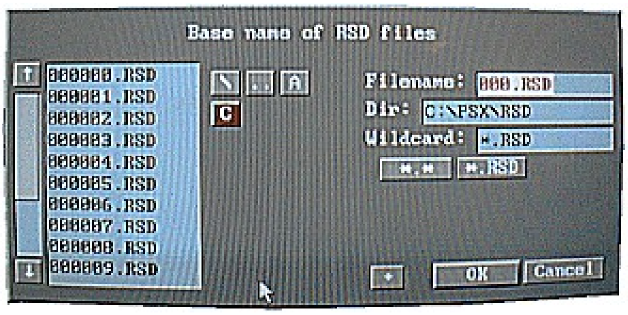
8. Now click on the ‘save RSD’ button. This saves each object in your animation as separate RSD files. Select OK when asked if you would like the files automatically named.



Generate File names automatically?

Automatic file name generation prompt.

When you are presented with the RSD save file dialog box enter the relevant details. For the purpose of this tutorial leave the details to the default and just select the OK button.



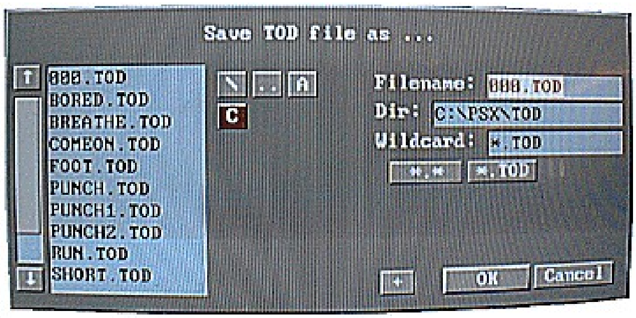
Base name of RSD files

000000.RSD
000001.RSD
000002.RSD
000003.RSD
000004.RSD
000005.RSD
000006.RSD
000007.RSD
000008.RSD
000009.RSD

Filename: 000.RSD
Dir: C:\PSX\RSD
Wildcard: *.RSD

RSD Save dialog.

9. Next click on the ‘save TOD’ button. This saves your animation into a TOD file. This dialog is looks the same as the dialog in step 8. You may wish to change the name of the TOD file.



Save TOD file as ...

000.TOD
BORED.TOD
BREATHE.TOD
COMEON.TOD
FOOT.TOD
PUNCH.TOD
PUNCH1.TOD
PUNCH2.TOD
RUN.TOD
SHORT.TOD

Filename: 000.TOD
Dir: C:\PSX\TOD
Wildcard: *.TOD

TOD file save dialog.

10. You must now exit 3D studio 4.0. Press ‘Q’ and reply to the resulting dialog with a yes.

11. In the directory ‘C:\PSX\RSD\’ edit the file ‘000.RSD’, this contains a list of all the RSD files in your animated model. Delete all the commented lines (lines that begin with ‘#’), to leave a list of the RSD file names. Save this file as model.arg.

12. At the DOS prompt type:-

```
rsdlink -v -o  model.tmd model.arg
```

You now have your TMD file and your animated TOD file ready to include in your program.

Explanation of Functions

This section will explain what parameters are required by the functions that are needed to use TOD files. This will help you understand what data you need at hand for writing your own program. Remember an example program is at the end of this tutorial.

```
void InitialiseModelLink(pointer to the object, Maximum RSD, Number of TOD,
                         X, Y, Z, Models Address);
```

The parameters for this function are a pointer to the structure of itself, the amount of objects in the TMD file then its initial position (X,Y,Z). Also the physical memory address of the TMD is needed as well as the number of animations that will be associated with this object.

void InitTod (pointer to the object, TOD number, ID list, TOD Address);

The first parameter is a pointer to the object structure. The next two parameters are the TOD number and its physical memory address. The TOD number is what will associate which animation is being referred to and can be any number from 0 to the Number of TOD files value set when initializing this model. The ID list can be ignored as this is usually set to a null value.

void ResetTod(pointer to the object , TOD number);

reset to the first frame of the TOD.

void DrawObjectLink(the object table);

The only parameter required by this function is the object's object table itself.

u_long NextFrame(pointer to the object, loop);

This function needs to return a value indicating whether the animation has reached the end or not. Again a point to the object needs to be passed to the function. The function also needs to know if you want the animation to loop when it completes the animation or if you want it to stop until it is reset.

Using the TOD file in your program

The first thing to do is change the OBJS line in your 'makefile.' So that it includes objctl.o todanim.o and graphics.o. Your resulting make file should look like this.

```
CFLAGS  = -Wall
LINKER  = -Xlinker -Ttext -Xlinker 80140000

PROG =        main
OBJS    =    main.o loadTims.o loadTexs.o pad.o objctl.o todanim.o graphics.o

all: $(PROG)

main    :$(OBJS)
        $(CC) $(LINKER) $(OBJS) -o $@
        strip $@
.c.o:
        $(CC) $(CFLAGS) -funsigned-char -c $<
```

Next move all the relevant files to the data directory of your program, this includes the TMD, TOD and TIM files. Enter the file names into you 'memtool.dat' file, and then run memtool. As memtool doesn't support TOD files correctly I suggest you list these last.

Once memtool has run you need to write a line in your ‘addrs.h’ file to define the TOD files address. To do this simply copy the address from the ‘auto.’ file adding ‘0x’ to the beginning. The following shows an example of how to do this with the file named ‘anim.tod’.

Auto.	Addrs.h
local dload data\ANIM.TOD 800A31D0	#define ANIM_TOD 0x800A31D0

Next comes the fun part.

In your program you must declare the model to be of type LinkObjectModel. Then at the beginning of your main function you must call the initializing functions for the model and one for each TOD file that is associated with that model. You must then reset the TOD or the model will not be visible and your program may even crash..

Then each screen refresh you call the DrawLinkModel function to draw the model to the screen. To advance a frame of the animation simply call the NextFrame function. A skeleton example program. The functions and their parameters are explained immediately after this example:-

```
//includes
#include <libps.h>
#include “graphics.h”
…
//globals
LinkObjectModel     theModel;     //Instance of linked object
…
// Function Main
void main()
{
        …
        InitialiseLModelLink(&theModel,8,1,0,0,0,MODEL_TMD);
        InitTod(&theModel,0,NULL,ANIM_TOD);
        ResetTod(&theModel,0);
        …
        For(;;)
        {
                RenderPrepare();
                …
                DrawModelLink(&theModel.objs);
                …
                RenderFinish();
                NextFrame(&theModel,0);
        }
        …
}
```