Table of contents:

Disclaimer: This program was initially intended for private use by the developer, so the following program is not very user-friendly, and may crash if something unintended is done. However, this guide will teach you how to use the program so it (hopefully) never happens. That being said, you should always have a backup of your levels and save frequently.

This program was originally made to give a game I am developing a special file format that provides the game with vital information any other file format would not be able to give, however this program can be used as an alternative to regular 3d modelers like Blender, 3DSMax, etc. There are tools given in the installation that will allow you to convert these files to .obj files and vice versa so you can use these models (often called "levels" in this manual due to the modeler often being used to create levels) for your regular needs. If you are using this program as a replacement of normal 3d modeling programs, there is a lot of information in this manual you will not need to know. For example, the chapter on the .uwl file format. Actually, the only time you will ever need that chapter is if you want to create your own program that uses that format, or if you want to modify the 3d modeler or the format itself.

1. Files and folders

img
: Misc. images used in the program.

saves
: Where level files are located.

sources
: Where the source code of each program is located. (Python and used libraries must be installed in order for them to be run, and the files need to be placed in the main directory when running.)

textures
: Where texture plane textures are located. Texture 0000 must always be present, and textures must be a 4 digit hexadecimal number (2 bytes) as they correlate with texture id.

_modeler.exe:
: The program.

manual.pdf:

The thing you are reading right now.

objto.exe
: Convert a .uwl file to a .obj file.

objfrom.exe
: Convert a .obj file to a .uwl file. (The .uwl file will miss important information in programs that can only use this file format, so you will need to manually input this information later on.)

readtrace.exe:
: Reads memory dumps.

2. Basic tutorial

The length of this tutorial section may seem intimidating, but that's because the program is not user-friendly, and what to do and what not to do has to be described in many words. Once you start modeling passed this tutorial, everything will go a lot faster and smoothly.

This tutorial of the program teaches how to use it as a normal modeler, as a substitute for other modelers. If your purpose of using this program is to create the special .uwl file format, you will need to learn how my game's collision and physics work, which will be explained in the game's documentation rather than here.

First, run _modeler.exe. You will see a control panel on the left, and the 3d environment on the right with a white dot at the origin.

You can hold left click on your mouse while moving the mouse to pan the camera, and hold right click while moving to rotate the camera. You can zoom by holding left click while scrolling the scroll wheel up or down. You can use the R and P keys to reset the camera's rotation and position respectively.

For this tutorial, we will be making a simple cube. These are the following coordinates of the cube's vertexes. You don't need to know the level's vertex coordinates in order to model it, but it will make things easier.

Side 1: (-1,1,1) (1,1,1) (1,1,-1) (-1,1,-1)
Side 2: (-1,-1,1) (-1,1,1) (-1,1,-1) (-1,-1,-1)
Side 3: (1,-1,1) (-1,-1,1) (-1,-1,-1) (1,-1,-1)
Side 4: (1,1,1) (1,-1,1) (1,-1,-1) (1,1,-1)
Side 5: (-1,-1,1) (1,-1,1) (1,1,1) (-1,1,1)
Side 6: (-1,-1,-1) (1,-1,-1) (1,1,-1) (-1,1,-1)

Let's create our first side.

Below the + and EX buttons on the control panel is a switch. The dot on the switch shows what setting it's on. The switch I'm talking about is the switch with the settings TEX and COL. Make sure the dot is below the TEX setting. To switch switch settings, click the area on the switch below the text, or click the text. To the right of that switch should be another switch with the settings OBJ, PLN, and VER. Make sure that is set to PLN.

Now, click the + button to add your first polygon. You should see the white dot move to the center of the polygon.

The TEX setting tells the program that we want to create a polygon with a texture, and the PLN setting lets the program know that we want to edit the polygon as a whole.

The polygon for now should have a weird texture on it, assuming you didn't tamper with the textures folder already. This is a test texture. You are not required to have custom textures on this cube for this tutorial, but you will eventually need to know how to, so why not explain it in this tutorial instead of making you figure it out on your own.

You can add a custom texture by putting an image file in the textures folder and naming it a 4 digit hexadecimal number with an extension of .png. For example, 047C.png, F70A.png, and 6E61.png are all valid file names, although it's good practice to create them in order: 0000.png, 0001.png, 0002.png, etc.

To apply a texture, click on the button below the second switch labeled "TEXTURE". Now, look at the white text to the right of the control panel. Look at the TEXTSEL label. If the number next to it is not 3, the modeler failed to detect your button click, and you should try clicking again. If it is 3, type the texture's file name. (without the extension.) As you type, you should see what you type start to appear next to the 3. If you mess up, you can use backspace to delete a digit. When you are done typing the texture, press enter/return to apply the texture. If you typed it correctly, the texture should display on the polygon. If the inputted texture file does not exist, the program will crash. If the text you entered does not follow proper formatting, (for example, the text being anything but 4 digits long, or invalid digits) pressing enter/return will do nothing.

Once you have the texture displayed, switch the right switch to the VER setting. This tells the program that we want to edit the polygon's individual vertexes. You should now see the white dot on one of the vertexes.

For the first vertex, search the control panel for the section labeled "PLANE VERTEX". Recall the coordinates listed earlier. The first coordinate of the first side is (-1,1,1). Click on the field labeled "X", then TEXTSEL should change to a number other than 0. Then, type in "-1", then press enter/return. The vertex should now move, or rather in this specific case, it should look like it dissipated. Repeat this process for the second and third values in (-1,1,1), and apply them in the Y and Z fields similar to how you just did with the X field.

Click the right arrow button near the top of the control panel. This changes the vertex you are selecting. Now, apply the next vertex: (1,1,1). Then select the next vertex, then apply (1,1,-1), then select the next vertex, then apply (-1,1,-1). If done properly, you should have side 1 done.

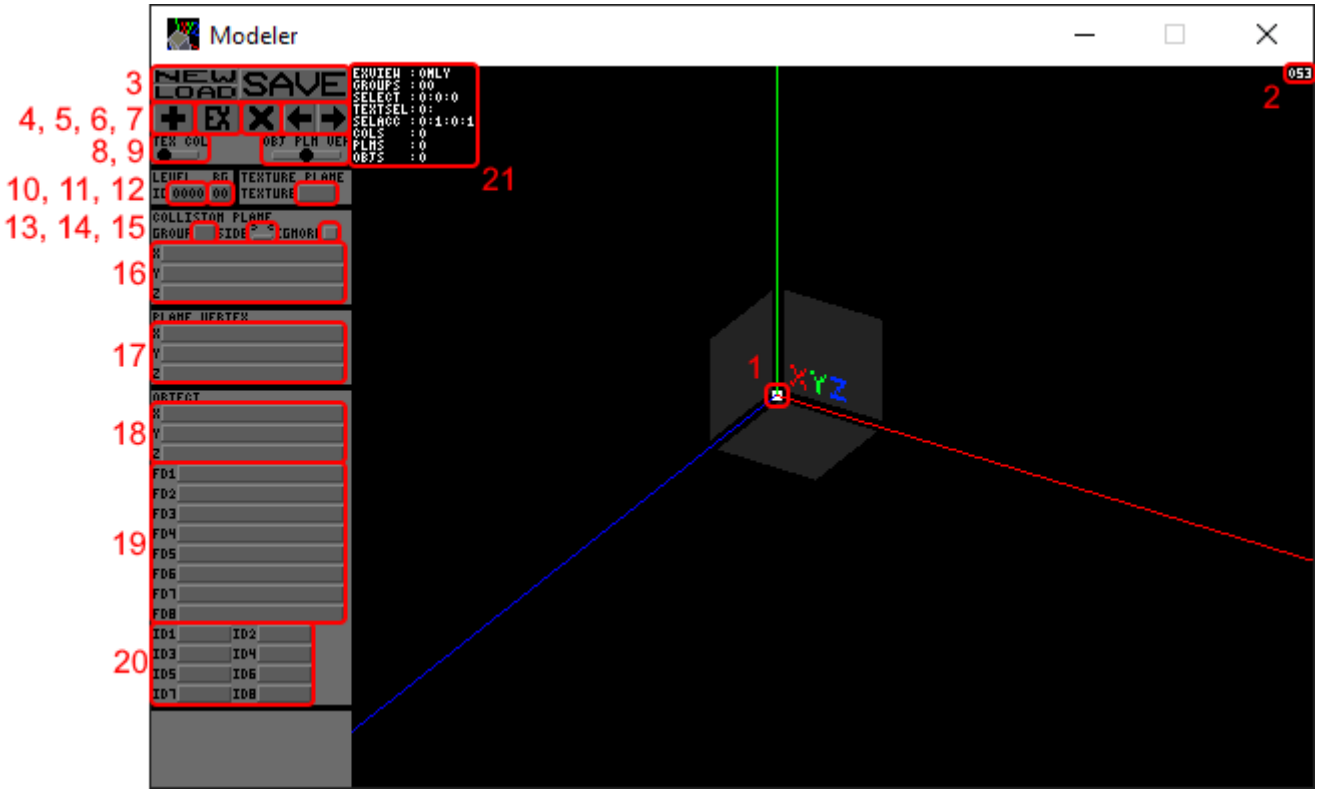
Add the last 5 sides and use the vertexes listed earlier to move each vertex. Make sure you switch back to the PLN setting before adding each polygon, then back to VER when moving the vertexes. If you add all 5 polygons at once before moving their vertexes, you can press the left and right buttons on the control panel while the PLN setting is enabled to switch to a different polygon.

If you did everything correctly, you should have a cube now. Look at the label "ID" below the left switch. You should see a 4 digit hexadecimal number. This is the name of the file that will be saved to the saves folder. If you already have a file with that name already, change the value before saving, or

the file will be overwritten. Press the save button on the top of the control panel to save your file. If you want to load your file, press the load button while ID is set to the file name.

The tutorial is now over. Assuming you learned how to use custom textures, you should now know everything you need to know in order to do basic modeling, however you should still go over the interface and controls chapters for extra information and precaution.

3. Interface



* = Can be ignored if the special .uwl file format is not needed for your purpose.

1. Axis, axis labels, and selection

The axis is labeled with three colored lines in the positive direction. The X axis is red, the Y axis is green, and the Z axis is blue, as shown in the axis label. (The letters "XYZ" shown.) The gray squares in between the axis gives you a better perception of the 3d space in terms of distance.

The white dot in the center of the screen serves multiple purposes depending on modes. When no element is selected, the dot will be white at the origin of the 3d space as shown above. When a plane or object is selected, the dot will show at the center of the element. When a vertex is selected, the dot will show at the vertex.

When a collision plane is selected, a second dot will show around the origin. This is the normal vector of the plane. The dot will show either red or green depending on whether or not the vector is on the side of the plane that results in the plane's inequality being true. Green means

true, red means false. The inequality symbol used in the equation is determined by the side setting.

2. FPS

Displays the number of frames drawn on screen in one second.
A general estimation of how fast your device can process the program.

3. New, load, and save

New: Erases all unsaved data.
Load: Load a file. The file loaded is determined by the value of level id.
Save: Save a file. The file saved is determined by the value of level id.

4. Add

Adds a texture plane if Mmode is set to TEX and Smode is not set to OBJ
Adds a collision plane if Mmode is set to COL and Smode is not set to OBJ
Adds an object if Smode is set to OBJ

5. *Exview

The setting of exview is shown in the misc. information section of the screen. Depending on the setting, the screen will display collision planes, texture planes, objects, or a combination of them, and the rest will appear invisible. This can allow you to work with only one or two types of elements without having the distraction of having other types present.

ALL: Displays all element types.

ONLY: Only display the element type that is being selected.

SUPPORT (COL/TEX):

If a collision or texture plane is selected, collision and texture planes are displayed.
If an object is selected, objects and texture planes are displayed.

ODD (COL/OBJ):

If a collision plane or object is selected, collision planes and objects are displayed.
If a texture plane is selected, texture planes and collision planes are displayed.

6. Delete

Deletes the selected object

7. Navigate selection

Changes the selected element by -1 if the left arrow is pressed, or by 1 if the right arrow is pressed. You can exponentially speed up selection change if you click multiple times quickly, so if there are 300 elements in your level, you don't need to click 300 times.

8. Mmode

Model mode.
Changes what type of element you are working with.
TEX = Texture plane, COL = Collision plane

9. Smode

Selection mode.
Changes what type of element you are selecting.
OBJ = Object, PLN = Plane, VER = Vertex

10. Level id
    The internal id of the level.
    This is what the level will be called when you save it to a file also.
    What id you give your level does not matter if the special .uwl file format is not needed for your purpose, however you can overwrite an existing file if they share the same id, so you have been warned.
    (Uses a 4 digit hexadecimal number (2 bytes) as input.)

11. *Level bg
    The background the level will use
    (Uses a 2 digit hexadecimal number (1 byte) as input.)

12. Texture id
    The texture the currently selected texture plane is using.
    (Uses a 4 digit hexadecimal number (2 bytes) as input.)

13. *Group
    The polygon group that the currently selected plane is in.
    (Uses a 2 digit hexadecimal number (1 byte) as input.)

14. *Side
    The inequality sign used in when calculating if the plane's normal vector is true in its plane's inequality equation.

15. *Ignore
    Whether or not this plane should be ignored during physics detection.
    (A blue quare means it is enabled.)

16. *Collision xyz
    The currently selected collision plane/vertex's coordinates.

17. Plane xyz
    The currently selected texture plane/vertex's coordinates.

18. *Object xyz
    The currently selected object's coordinates.

19. *Fdata
    Floating point data numbers 1-8.

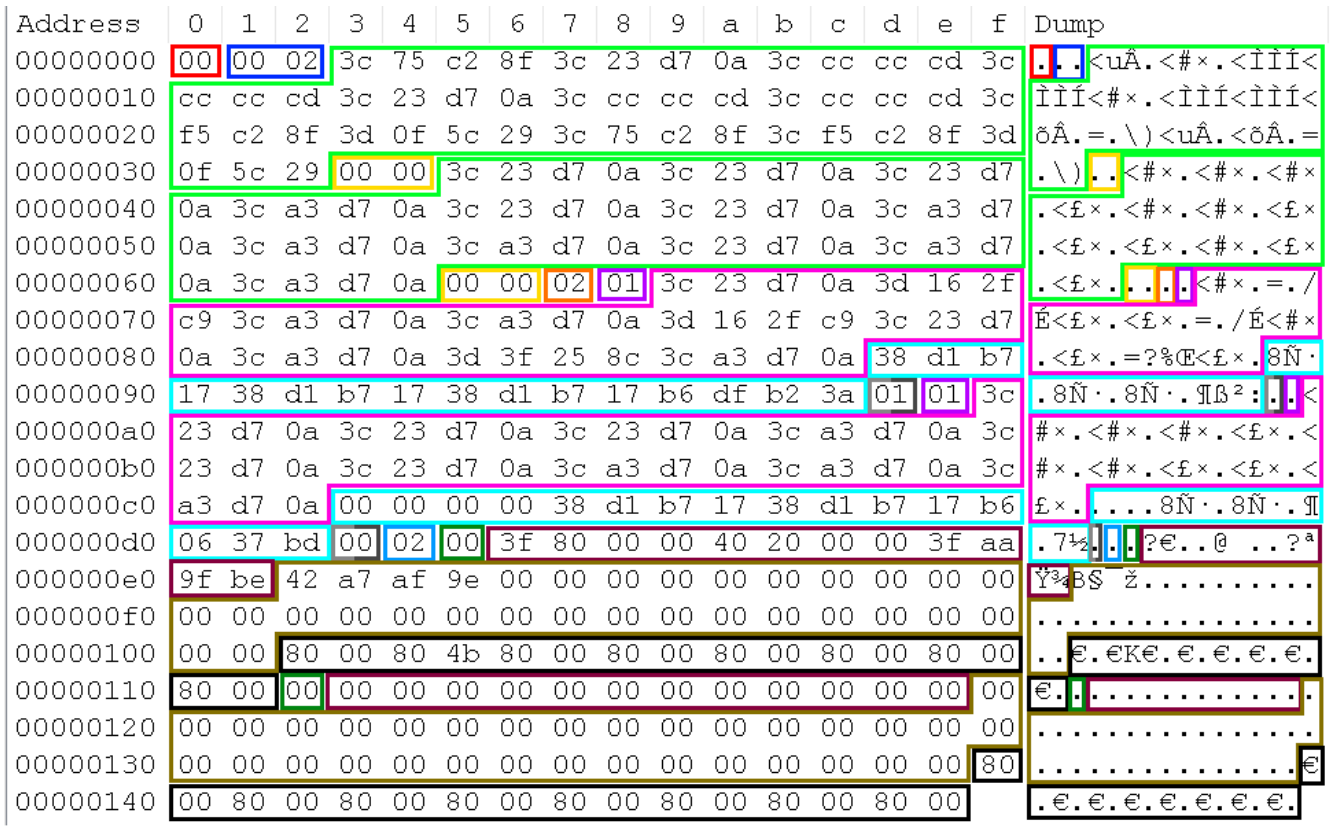
20. *Idata
    Integer data numbers 1-8.
    The first number indicates the object type.
    If the object type has sub-types, the second number indicates the sub-type.

21. Misc. info
    - EXVIEW:
        The current exview setting.
    - GROUPS:
        The number of groups in the level. (There is no simple way to calculate this, so it just says the highest group used, not the actual amount.)
    - SELECT:
        Which object, plane, and vertex is being selected.
    - TEXTSEL:
        Which text field is selected, and the input the user is currently typing.
    - SELACC:
        Frames until left/right navigation selection acceleration wares off, and how many elements will be navigated if clicked again.
    - COLS:
        The number of collision planes in the level.
    - PLNS:
        The number of texture planes in the level.
    - OBJS:
        The number of objects in the level.

4. Controls

Left mouse button: Click button.
Left mouse button (hold): Pan the camera in the X or Y axis.
Left mouse button (hold) + scroll wheel: Pan the camera in the Z axis.
Right mouse button (hold): Rotate the camera in the X or Y axis.
Right mouse button (hold) + scroll wheel: Rotate the camera in the Z axis.
0-9, A-F, -, .: Type numbers 0-9, letters A-F, a dash, or period if a text field is selected.
Enter/return: Confirm and apply a text field if one is selected.
R: Reset camera rotation.
P: Reset camera position.
Keypad 0-9: Create a memory dump.

5. The .uwl file format



Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	00	00	02	3c	75	c2	8f	3c	23	d7	0a	3c	cc	cc	cd	3c	...<uÂ.<#×.<ÌÌÍ<
00000010	cc	cc	cd	3c	23	d7	0a	3c	cc	cc	cd	3c	cc	cc	cd	3c	ÌÌÍ<#×.<ÌÌÍ<ÌÌÍ<
00000020	f5	c2	8f	3d	0f	5c	29	3c	75	c2	8f	3c	f5	c2	8f	3d	õÂ.=.\)<uÂ.<õÂ.=
00000030	0f	5c	29	00	00	3c	23	d7	0a	3c	23	d7	0a	3c	23	d7	.\)..<#×.<#×.<#×
00000040	0a	3c	a3	d7	0a	3c	23	d7	0a	3c	23	d7	0a	3c	a3	d7	.<£×.<#×.<#×.<£×
00000050	0a	3c	a3	d7	0a	3c	a3	d7	0a	3c	23	d7	0a	3c	a3	d7	.<£×.<£×.<#×.<£×
00000060	0a	3c	a3	d7	0a	00	00	02	01	3c	23	d7	0a	3d	16	2f	.<£×.....<#×.=./
00000070	c9	3c	a3	d7	0a	3c	a3	d7	0a	3d	16	2f	c9	3c	23	d7	É<£×.<£×.=./É<#×
00000080	0a	3c	a3	d7	0a	3d	3f	25	8c	3c	a3	d7	0a	38	d1	b7	.<£×.=?%Œ<£×.8Ñ·
00000090	17	38	d1	b7	17	38	d1	b7	17	b6	df	b2	3a	01	01	3c	.8Ñ·.8Ñ·.¶ß²:..<
000000a0	23	d7	0a	3c	23	d7	0a	3c	23	d7	0a	3c	a3	d7	0a	3c	#×.<#×.<#×.<£×.<
000000b0	23	d7	0a	3c	23	d7	0a	3c	a3	d7	0a	3c	a3	d7	0a	3c	#×.<#×.<£×.<£×.<
000000c0	a3	d7	0a	00	00	00	00	38	d1	b7	17	38	d1	b7	17	b6	£×.....8Ñ·.8Ñ·.¶
000000d0	06	37	bd	00	02	00	3f	80	00	00	40	20	00	00	3f	aa	.7½.....?€..@ ..?ª
000000e0	9f	be	42	a7	af	9e	00	00	00	00	00	00	00	00	00	00	Ÿ¾B§ ¯ž..........
000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000100	00	00	80	00	80	4b	80	00	80	00	80	00	80	00	80	00	..€.€K€.€.€.€.€.
00000110	80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	€..............
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	80	...............€
00000140	00	80	00	80	00	80	00	80	00	80	00	80	00	80	00		.€.€.€.€.€.€.€.

Background.
Number of texture planes.
Texture plane xyz1234.
Texture plane texture id.
Number of groups.
Number of collision planes.
Collision plane xyz123.
Collision plane slope.
Collision plane side.
Collision plane ignore.
Number of objects.
Object id.
Object xyz.
Object fdata.
Object idata.

In the above example is a level with 2 texture planes, 2 objects, and 2 groups, each with 1 collision plane. On the right is the file shown in extended ASCII, the letters you would probably see in a normal text editor, although special characters like NUL, SOH, and STX are displayed as a period in this example. On the left is the file displayed in hexadecimal.

This file format does not use delimiters. Instead, each section has a byte or two that tells how many entries are in the section. Since each entry in each section has the same length no matter what, a delimiter is not needed; the end of one section just leads into another, as shown in the example.

The collision plane side and ignore values are flags, and because of that they can be stored in the same byte as shown in the example. The least significant bit defined the ignore flag, and the second most least significant bit defines the side flag. If the bit is set, the inequality symbol used is ">", and vice versa is true.

Integer values, like idata, are signed short integers. They are 2 bytes big, and can range from -32768 to 32767 with the hexadecimal value $8000 representing 0.

Floating point numbers, used in coordinates, slopes, and fdata, are stored in 4 bytes. Their format is a bit complex; the modeler program uses a library to convert between floats and bytes instead of making my own conversion code. These float values are stored using the single precision IEEE 754 standard.