

# Parsing the Object Table

The object table contains offsets to all of the object's various data segments and also determines the interrelation of those segments. When considering creating or visualizing a large, multi-piece object, it is important to know which blocks position each individual piece, which pieces are part of a larger structure, and which affect the model as a whole. All of this can be gleaned from the basic structure of the object table.

Each table entry consists of six words, five of which define file offsets.

| Type of data at second offset | Offset to parent entry | Offset to previous entry |
|-------------------------------|------------------------|--------------------------|
| 00 02 00 00                   | 05 00 19 64            | 05 00 00 D0              |
| Offset in file to data        | Offset to next entry   | Offset to child entry    |

The first two words are used to dictate what kind of data is expected and where in the file it is located. The other four offsets point to other entries in the object table and are used for ordering the table.

There are two ways to link entries. **Child** entries share all of the attributes of their parents. Display lists are almost always children of other commands, adopting their position from parent position commands, the distance they are rendered at from distance commands, etc. By creating a series of parent/child entries, many different attributes can be assigned to a part of the model.

The **next** entry is the next table entry that has the same parent as the current command. The next entry does not share the attributes of the current entry, but still shares those of its own parents. All next and previous entries have the attributes of their parents but don't share attributes with each other. If there is more than one model for an object, like a high-resolution model when nearby and a lower-resolution model when at a distance, the near model's command will have a **next** entry set to the far model.

To better illustrate the connections between each command, some examples use a simple format giving the type of data, the offset of the object table entry, and brackets that reflect which commands follow and are children of that entry.

| Offset | Type     | Data Offset | Parent Offset | Next Entry | Previous Entry | Child Offset |
|--------|----------|-------------|---------------|------------|----------------|--------------|
| 0x100  | 00020000 | 05000148    | 00000000      | 00000000   | 00000000       | 05000118     |
| 0x118  | 00180000 | 05000420    | 05000100      | 05000130   | 00000000       | 00000000     |
| 0x130  | 00180000 | 05000500    | 05000100      | 00000000   | 05000118       | 00000000     |

| Type   | Offset in Table | Parent | Child    | Parent | Children           | Next Entry |
|--------|-----------------|--------|----------|--------|--------------------|------------|
| 02:100 |                 | 02:100 | {18:118} | 02:100 | {18:118<br>18:130} |            |

02:100 denotes a table entry at offset 0x100 in the file. The command type is 02, which sets the position of a model part. It's first child entry is at 0x118, which is the next command. This is an 18 display list, and because it is a child of 02:100 it appears in brackets next to it. The parent offset is set to 0x100, it has no children of its own, and it is followed by another entry at 0x130. This is also a 18 display list, and it has the same parent as the previous entry at 0x118. It is also a child of 02:100 so it appears in brackets, but because it has no relation to 18:118 it is placed one line down.

Being children of 02:100, both 18:118 and 18:130 will adopt the position data set by 02:100. Technically the parts are rendered separately and moved to the position given in 02:100, but functionally it is as if every point in both of the display lists has had the position's x, y, and z values added to them. Neither 18 commands share any information with each other. If one of these had children, ie:

```
02:100  {0A:118  {18:130}
        18:148}
```

18:130 would be affected by 0A:118 and 02:100, since it is a child of both. However, 18:148 is only a child of 02:100, so it receives the position data but not the affects of either 0A:118 or 18:130. Both of those commands are outside of 18:148's scope.

Note though that parent's receive nothing from children. 02:100 in both examples receives no attributes from any of its children. In the second example, 0A:118 adopts the position of its parent, but the display list is beyond its own scope. Effects only compound from left to right.

Precedence becomes increasingly important in larger models. In guards and other large, multi-segmented models each piece of the model will be formed with one or more display lists and positioned relative to the rest of the model using one or more position commands. For instance, a hand is positioned to touch a forearm, the hand and forearm positioned to touch the triceps, and the whole arm is positioned on the chest, which is positioned above the legs to form the actual body.

```
01:B8      {02:D0      {02:3B8      {02:5B0      {02:5C8      {02:5E0      {0A:5F8      {18:610}}
                                                    0A:628      {08:640      {18:658}
                                                    08:670      {18:688}}}
                                                    0A:6A0      {08:6B8      {18:6D0}
                                                    08:6E8      {18:700}}
```

All of this code is used to render the left arm of Brosnan's tuxedo. 18:610 is the character's left hand. 18:658 and 18:688 render their left forearm, using the 08 commands to indicate at what distance each should be used. The 18:6D0 and 18:700 also trigger at specific distances and are used to render the upper arm.

Every point in 18:610 is moved a total of five times; 02:D0, 02:3B8, 02:5B0, 02:5C8, and 02:5E0 all affect it's final position. 18:658 and 18:688 are only moved four times since 02:5E0 is outside their scope. 18:6D0 and 18:700 are only moved three times total, 02:5C8 and 02:5E0 both being out of their scope.

To determine exactly what commands influence a certain table entry is as simple as gathering a list of all its parents. For instance, if a full object table isn't handy and you need to know all the position modifications needed to render the display list in command 18:700, you can retrieve its parent's offset, then that parent's offset, etc., until finally the first entry is reached that lacks one. All the commands that influence 18:700 are:

08:6E8, 0A:6A0, 02:5B0, 02:3B8, 02:D0, 01:B8.

### Simple Display Lists: Position Translation

Display lists render each of the parts of an object. All display list types consist of a table of points in the model as well as a list of display commands sent to the RDP. An example of a basic display list is the type 04 command, broken down here.

Offset to 1st Mapping      Reserved      Points in Table  
05004AF8 00000000 00000000 050017A8 0006 0300  
Offset to 2nd Mapping      Offset to Point Table

The mapping commands are numerous and complex, but for these purposes discussing them isn't important. Instead, a brief look at point tables is in order. A point table is constructed of many point structs, as seen here:

| # | X Position | Y Position | Z Position | Reserved | S Image Pin | T Image Pin | RGBA Color |
|---|------------|------------|------------|----------|-------------|-------------|------------|
| 0 | 0020       | FF96       | 0050       | 0000     | 0104        | 0099        | 676767FF   |
| 1 | FFE0       | FF96       | 0050       | 0000     | 00D5        | 0097        | 676767FF   |
| 2 | FFE0       | FF86       | 003B       | 0000     | 00D6        | 00A7        | 676767FF   |
| 3 | 0020       | FF86       | 003B       | 0000     | 0104        | 00A9        | 676767FF   |
| 4 | 0020       | 0000       | 0000       | 0000     | 0106        | 00D4        | 676767FF   |
| 5 | FFE0       | 0000       | 0000       | 0000     | 00D8        | 00D2        | 676767FF   |

Given above is a list of six points which would be used by the display list to render an object. The x, y, and z positions are used to produce the model itself. The s and t pins are used for alignment when image mapping, and a red/green/blue/alpha quad affects the resulting color of the generated point. The reserved short value is not used in basic display lists. For the rest of the document, only the three position values will be considered.

Each display list is rendered independently of each other but other commands in the object table can change the way in which rendering occurs. For instance, 08 commands only allow the display list to be rendered at certain distances, 12 commands can be used to select what kind of mapping is used, etc.

A part can be repositioned when one of the position commands is used. Each point within a subsequent child display list is offset by the given x, y, and z values, moving the whole part to a new location. The most common of these are type 02 commands.

X Position      Z Position      Piece Order #s      Unknown  
00000000 C1AA6670 C21484ED 000C 000C FFFF FFFF 00000000 430B7864  
Y Position      Command Order #      Offset to Child O2 Cmd.

For our purposes, the only parts of these position commands that will be considered are the three position values. The ordering and linkage values are used predominately by animation

routines to select particular parts and manipulate them. The position values are always important though, since it is these that will be used to offset the point information in the child display list.

To apply the change, the given offsets are added to every position in its child point tables. This requires first converting them from floating-point notation into their decimal equivalents, then adding the values to each point in the table.

|                |          |        |
|----------------|----------|--------|
| X modification | 00000000 | + 0.   |
| Y Modification | C1AA6670 | -21.30 |
| Z Modification | C21484ED | -37.13 |

| # | X Original | Y Original | Z Original | X Moved | Y Moved | Z Moved |
|---|------------|------------|------------|---------|---------|---------|
| 0 | 32         | -106       | 80         | 32      | -127.3  | 42.87   |
| 1 | -32        | -106       | 80         | -32     | -127.3  | 42.87   |
| 2 | -32        | -119       | 59         | -32     | -140.3  | 21.87   |
| 3 | 32         | -119       | 59         | 32      | -140.3  | 21.87   |
| 4 | 32         | 0          | 0          | 32      | -21.3   | -37.13  |
| 5 | -32        | 0          | 0          | -32     | -21.3   | -37.13  |

Multiple position commands can be applied to a point table. Below is the complete command list of the display list. Another position command influences the position of the part shown.

02:68            {02:290            {04:2A8}

02:68 positions the 04:2A8 display list in addition to the changes made by 02:290. The same process is used to move each of the points in the table. 02:68's position changes are shown in the short table below.

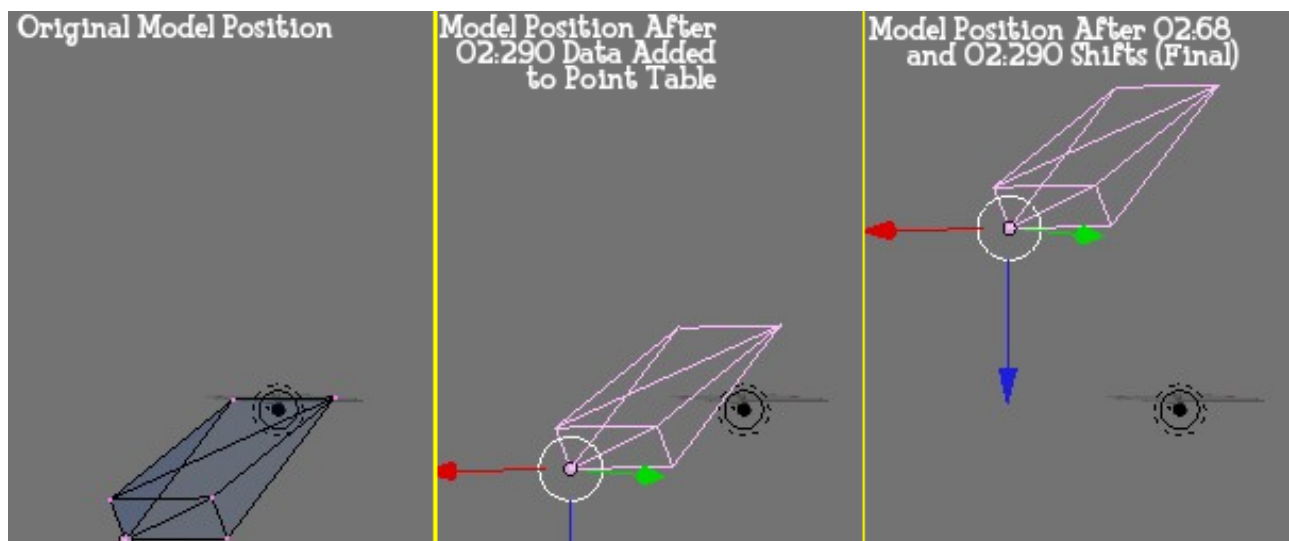
|                |          |            |
|----------------|----------|------------|
| X modification | 00000000 | + 0.       |
| Y Modification | 379F6230 | + 0.000019 |
| Z Modification | C2FFE3EF | -127.95    |

The larger table below shows how the changes are compounded to create the final position. The original position values are given on the left, converted to their decimal equivalents. The middle columns denote the position changes made by the 02:290 position command, and the right-most columns illustrate the 02:68 position change.

| # | X Org. | Y Org. | Z Org. | X First | Y First | Z First | X Final | Y Final | Z Final |
|---|--------|--------|--------|---------|---------|---------|---------|---------|---------|
| 0 | 32     | -106   | 80     | 32      | -127.3  | 42.87   | 32      | -127.3  | -85.08  |
| 1 | -32    | -106   | 80     | -32     | -127.3  | 42.87   | -32     | -127.3  | -85.08  |
| 2 | -32    | -119   | 59     | -32     | -140.3  | 21.87   | -32     | -140.3  | -106.08 |
| 3 | 32     | -119   | 59     | 32      | -140.3  | 21.87   | 32      | -140.3  | -106.08 |
| 4 | 32     | 0      | 0      | 32      | -21.3   | -37.13  | 32      | -21.3   | -165.08 |
| 5 | -32    | 0      | 0      | -32     | -21.3   | -37.13  | -32     | -21.3   | -165.08 |

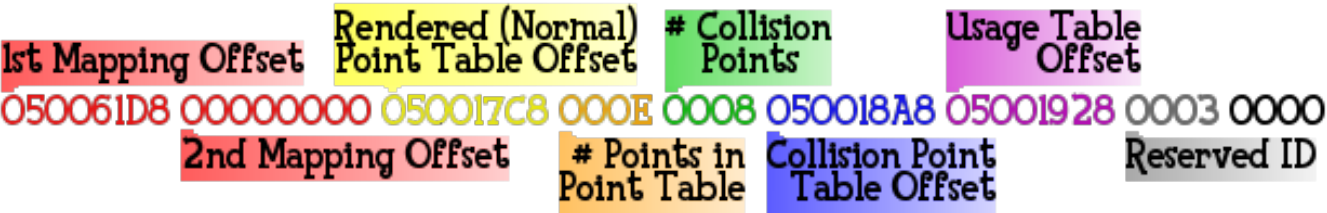
The right-most three columns are the final positions for each of the points in the point table. The total position change is equal to the position changes made by each of the parent position commands. Compounding position changes in this way moves the part to the final location in the rendered model.

|                | 02:290 | 02:68      | Total Change |
|----------------|--------|------------|--------------|
| X modification | + 0.   | + 0.       | + 0.         |
| Y Modification | -21.30 | + 0.000019 | + 21.300019  |
| Z Modification | -37.13 | -127.95    | -165.08      |



Complex Display Lists: Point Aliasing

Type 18 display list commands contain more than just basic points used for rendering. There are two other groups of data contained in the block used for registering hit detection and merging points with different parts of the model. A complete 18 command's data is shown below.



Collision tables are a separate index of the unique points found in the rendering point list. The format differs slightly from the ordinary point tables, retaining x, y, and z position data but including an index value and information used later to merge coordinates. The format of the point structs will be discussed in the next section.

Contrast the two different kinds of point tables. On the left is a normal point table, and on the right is the collision table assigned to the same display list. Notice the fourth column in the collision point table, the index value, matches the point numbers on the left.

| # | Rendering (Normal) Point Table |      |      |      |      |      |          | Collision Point Table |      |      |      |          |      |      |
|---|--------------------------------|------|------|------|------|------|----------|-----------------------|------|------|------|----------|------|------|
| 0 | FF9B                           | 001E | FFB2 | 0000 | 000C | 000A | 303030FF | FF9B                  | 001E | FFB2 | 0000 | 00000000 | FFFF | 0000 |
| 1 | FF9B                           | FFC8 | FFB8 | 0000 | 0037 | 000B | 303030FF | FF9B                  | FFC8 | FFB8 | 0001 | 00000000 | FFFF | 0000 |
| 2 | FF9B                           | 004B | 00A2 | 0000 | 063E | 002E | 303030FF | FF9B                  | 004B | 00A2 | 0002 | 00000000 | FFFF | 0000 |
| 3 | FF9B                           | FFCE | 00AB | 0000 | 067C | 0030 | 303030FF | FF9B                  | FFCE | 00AB | 0003 | 00000000 | FFFF | 0000 |
| 4 | FFBF                           | 0026 | 0094 | 0000 | 05F8 | 0103 | D4D4D4FF | FFBF                  | 0026 | 0094 | 0004 | 00000000 | FFFF | 0000 |
| 5 | FFF7                           | 0030 | FFFC | 0000 | 0236 | 0239 | 5E5E5EFF | FFF7                  | 0030 | FFFC | 0005 | 05000190 | 000D | 0000 |
| 6 | FFFC                           | FFDC | FFCC | 0000 | 00FB | 0256 | 5E5E5EFF | FFFC                  | FFDC | FFCC | 0006 | 05000190 | 0011 | 0000 |
| 7 | FFBD                           | FFEB | 00AF | 0000 | 06AA | 00FB | E2E2E2FF | FFBD                  | FFEB | 00AF | 0007 | 00000000 | FFFF | 0000 |
| 8 | 0002                           | 000C | FFBF | 0000 | 00A6 | 0272 | 5E5E5EFF | 0002                  | 000C | FFBF | 0008 | 05000190 | 000F | 0000 |
| 9 | 0003                           | FFE1 | FFEF | 0000 | 01E4 | 0281 | 5E5E5EFF | 0003                  | FFE1 | FFEF | 0009 | 05000190 | 0010 | 0000 |
| A | 0002                           | FFE7 | 0015 | 0000 | 02DF | 0284 | 5E5E5EFF | 0002                  | FFE7 | 0015 | 000A | 05000190 | 000E | 0000 |
| B | FFFB                           | 0020 | FFE1 | 0000 | 0187 | 0252 | 5E5E5EFF | FFFB                  | 0020 | FFE1 | 000B | 05000190 | 000C | 0000 |

The two tables shown above looked very similar because each and every point in the rendering display list is unique. Often, though, this isn't the case. Only 16 points can be utilized at a time from a rendering point table, so often a point will need to be repeated later in the model. In addition, sometimes the same point will be used several times but utilize different mipmapping coordinates or coloration. As was previously stated, collision point tables only list *unique* coordinates, so it is not uncommon to see a collision table with fewer entries than a render table.

#### Rendering (Normal) Point Table (24 entries)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 0B | 00 | 09 | FF | DA | 00 | 00 | 02 | B1 | 00 | DD | BC | BC | BC | FF |
| 00 | 0A | 00 | 21 | 00 | 05 | 00 | 00 | 01 | 88 | 00 | 3B | BC | BC | BC | FF |
| 00 | 60 | 00 | 17 | FF | DB | 00 | 00 | 02 | 87 | 03 | 2C | F0 | F0 | F0 | FF |
| 00 | 09 | 00 | 17 | 00 | 23 | 00 | 00 | 00 | BD | FF | CD | D4 | D4 | D4 | FF |
| 00 | 6B | 00 | 32 | 00 | 2E | 00 | 00 | 00 | 47 | 02 | 52 | F0 | F0 | F0 | FF |
| 00 | 60 | 00 | 17 | FF | DB | 00 | 00 | 00 | A1 | 02 | 31 | F0 | F0 | F0 | FF |
| 00 | 77 | FF | E7 | FF | CB | 00 | 00 | 03 | 25 | 03 | A9 | F0 | F0 | F0 | FF |
| 00 | 0B | 00 | 09 | FF | DA | 00 | 00 | 01 | 5D | FF | E9 | BC | BC | BC | FF |
| 00 | 0E | FF | E5 | FF | DF | 00 | 00 | 03 | 36 | 00 | 94 | D4 | D4 | D4 | FF |
| 00 | 0E | FF | DF | 00 | 20 | 00 | 00 | 02 | A3 | 03 | 8B | D4 | D4 | D4 | FF |
| 00 | 82 | FF | E3 | 00 | 33 | 00 | 00 | 03 | D1 | 00 | 28 | F0 | F0 | F0 | FF |
| 00 | 09 | 00 | 17 | 00 | 23 | 00 | 00 | 00 | 26 | 03 | 40 | D4 | D4 | D4 | FF |
| 00 | 6B | 00 | 32 | 00 | 2E | 00 | 00 | 00 | 0D | 00 | 3D | F0 | F0 | F0 | FF |
| 00 | 0E | FF | E3 | FF | FE | 00 | 00 | 02 | 6D | 00 | 18 | D4 | D4 | D4 | FF |
| 00 | 0E | FF | E5 | FF | DF | 00 | 00 | 01 | 43 | 00 | 2B | D4 | D4 | D4 | FF |
| 00 | 77 | FF | E7 | FF | CB | 00 | 00 | FF | FF | 02 | 8E | F0 | F0 | F0 | FF |
| 00 | 82 | FF | E3 | 00 | 33 | 00 | 00 | 03 | DA | 02 | 8F | F0 | F0 | F0 | FF |
| 00 | 0E | FF | DF | 00 | 20 | 00 | 00 | 03 | AF | 00 | 02 | D4 | D4 | D4 | FF |
| 00 | 0E | FF | E3 | FF | FE | 00 | 00 | 02 | 6D | 00 | 18 | D4 | D4 | D4 | FF |
| 00 | 77 | FF | E7 | FF | CB | 00 | 00 | FF | FF | 02 | 8E | F0 | F0 | F0 | FF |
| 00 | 77 | FF | E7 | FF | CB | 00 | 00 | FF | E9 | 02 | 1E | F0 | F0 | F0 | FF |
| 00 | 60 | 00 | 17 | FF | DB | 00 | 00 | 00 | A5 | 03 | 48 | F0 | F0 | F0 | FF |
| 00 | 82 | FF | E3 | 00 | 33 | 00 | 00 | 03 | B1 | 02 | 3E | F0 | F0 | F0 | FF |
| 00 | 6B | 00 | 32 | 00 | 2E | 00 | 00 | 03 | B8 | 04 | 16 | F0 | F0 | F0 | FF |

#### Collision Point Table (9 entries)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 0B | 00 | 09 | FF | DA | 00 | 00 | 05 | 00 | 04 | F0 | 00 | 11 | 00 | 00 |
| 00 | 0A | 00 | 21 | 00 | 05 | 00 | 01 | 05 | 00 | 04 | F0 | 00 | 10 | 00 | 00 |
| 00 | 60 | 00 | 17 | FF | DB | 00 | 02 | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 |
| 00 | 09 | 00 | 17 | 00 | 23 | 00 | 03 | 05 | 00 | 04 | F0 | 00 | 0F | 00 | 00 |
| 00 | 6B | 00 | 32 | 00 | 2E | 00 | 04 | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 |
| 00 | 77 | FF | E7 | FF | CB | 00 | 06 | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 |
| 00 | 0E | FF | E5 | FF | DF | 00 | 08 | 05 | 00 | 04 | F0 | 00 | 0D | 00 | 00 |
| 00 | 0E | FF | DF | 00 | 20 | 00 | 09 | 05 | 00 | 04 | F0 | 00 | 0E | 00 | 00 |
| 00 | 82 | FF | E3 | 00 | 33 | 00 | 0A | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 |

Notice there is not a 1-1 correspondence between the two tables. Instead, you have to resort to the index numbers in each table to see the relationship. Again, the rendered points are shown on the left, and the corresponding collision points to the right. Each of the index values is highlighted in blue.

| #  | Rendering (Normal) Point Table |      |      |      |      |      |          |      | Corresponding Collision Point Table |      |      |          |      |      |  |
|----|--------------------------------|------|------|------|------|------|----------|------|-------------------------------------|------|------|----------|------|------|--|
| 0  | 000B                           | 0009 | FFDA | 0000 | 02B1 | 00DD | BCBCBCFF | 000B | 0009                                | FFDA | 0000 | 050004F0 | 0011 | 0000 |  |
| 1  | 000A                           | 0021 | 0005 | 0000 | 0188 | 003B | BCBCBCFF | 000A | 0021                                | 0005 | 0001 | 050004F0 | 0010 | 0000 |  |
| 2  | 0060                           | 0017 | FFDB | 0000 | 0287 | 032C | F0F0F0FF | 0060 | 0017                                | FFDB | 0002 | 00000000 | FFFF | 0000 |  |
| 3  | 0009                           | 0017 | 0023 | 0000 | 00BD | FFCD | D4D4D4FF | 0009 | 0017                                | 0023 | 0003 | 050004F0 | 000F | 0000 |  |
| 4  | 006B                           | 0032 | 002E | 0000 | 0047 | 0252 | F0F0F0FF | 006B | 0032                                | 002E | 0004 | 00000000 | FFFF | 0000 |  |
| 5  | 0060                           | 0017 | FFDB | 0000 | 00A1 | 0231 | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 6  | 0077                           | FFE7 | FFCB | 0000 | 0325 | 03A9 | F0F0F0FF | 0077 | FFE7                                | FFCB | 0006 | 00000000 | FFFF | 0000 |  |
| 7  | 000B                           | 0009 | FFDA | 0000 | 015D | FFE9 | BCBCBCFF |      |                                     |      |      |          |      |      |  |
| 8  | 000E                           | FFE5 | FFDF | 0000 | 0336 | 0094 | D4D4D4FF | 000E | FFE5                                | FFDF | 0008 | 050004F0 | 000D | 0000 |  |
| 9  | 000E                           | FFDF | 0020 | 0000 | 02A3 | 038B | D4D4D4FF | 000E | FFDF                                | 0020 | 0009 | 050004F0 | 000E | 0000 |  |
| A  | 0082                           | FFE3 | 0033 | 0000 | 03D1 | 0028 | F0F0F0FF | 0082 | FFE3                                | 0033 | 000A | 00000000 | FFFF | 0000 |  |
| B  | 0009                           | 0017 | 0023 | 0000 | 0026 | 0340 | D4D4D4FF |      |                                     |      |      |          |      |      |  |
| C  | 006B                           | 0032 | 002E | 0000 | 000D | 003D | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| D  | 000E                           | FFE3 | FFFE | 0000 | 026D | 0018 | D4D4D4FF | 000E | FFE3                                | FFFE | 000D | 050004F0 | 000C | 0000 |  |
| E  | 000E                           | FFE5 | FFDF | 0000 | 0143 | 002B | D4D4D4FF |      |                                     |      |      |          |      |      |  |
| F  | 0077                           | FFE7 | FFCB | 0000 | FFFF | 028E | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 10 | 0082                           | FFE3 | 0033 | 0000 | 03DA | 028F | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 11 | 000E                           | FFDF | 0020 | 0000 | 03AF | 0002 | D4D4D4FF |      |                                     |      |      |          |      |      |  |
| 12 | 000E                           | FFE3 | FFFE | 0000 | 026D | 0018 | D4D4D4FF |      |                                     |      |      |          |      |      |  |
| 13 | 0077                           | FFE7 | FFCB | 0000 | FFFF | 028E | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 14 | 0077                           | FFE7 | FFCB | 0000 | FFE9 | 021E | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 15 | 0060                           | 0017 | FFDB | 0000 | 00A5 | 0348 | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 16 | 0082                           | FFE3 | 0033 | 0000 | 03B1 | 023E | F0F0F0FF |      |                                     |      |      |          |      |      |  |
| 17 | 006B                           | 0032 | 002E | 0000 | 03B8 | 0416 | F0F0F0FF |      |                                     |      |      |          |      |      |  |



Only the first instance of a point will be listed in the table. Notice in the table above point five, which doesn't have a collision table entry, is the same as point two. Normally, determining which collision entries pertain to rendering points would require string comparisons. Point usage tables circumvent this problem. The point usage table is a simple list of short integer values that indicate if other points in the table share the same location. The usage table for the above struct is given below.

A point usage value is usually FFFF, meaning the point is not shared by another in the table. If a value is given, though, it indicates the next entry ID that will use the exact same point. For instance, if entry 0's value is 0007, entry 7 will use the same collision point. This is noted in the table below.

| #  | Usage Table Entry |                                   | Corresponding Collision Point Table |      |      |      |          |      |      |
|----|-------------------|-----------------------------------|-------------------------------------|------|------|------|----------|------|------|
| 0  | 0007              | Entry 7 is identical to entry 0   | 000B                                | 0009 | FFDA | 0000 | 050004F0 | 0011 | 0000 |
| 1  | FFFF              | No matching entries               | 000A                                | 0021 | 0005 | 0001 | 050004F0 | 0010 | 0000 |
| 2  | 0005              | Entry 5 is identical to entry 2   | 0060                                | 0017 | FFDB | 0002 | 00000000 | FFFF | 0000 |
| 3  | 000B              | Entry B is identical to entry 3   | 0009                                | 0017 | 0023 | 0003 | 050004F0 | 000F | 0000 |
| 4  | 000C              | Entry C is identical to entry 4   | 006B                                | 0032 | 002E | 0004 | 00000000 | FFFF | 0000 |
| 5  | 0015              | Entry 15 is identical to entry 5  | 0060                                | 0017 | FFDB | 0002 | 00000000 | FFFF | 0000 |
| 6  | 000F              | Entry F is identical to entry 6   | 0077                                | FFE7 | FFCB | 0006 | 00000000 | FFFF | 0000 |
| 7  | FFFF              | No matching entries               | 000B                                | 0009 | FFDA | 0000 | 050004F0 | 0011 | 0000 |
| 8  | 000E              | Entry E is identical to entry 8   | 000E                                | FFE5 | FFDF | 0008 | 050004F0 | 000D | 0000 |
| 9  | 0011              | Entry 11 is identical to entry 9  | 000E                                | FFDF | 0020 | 0009 | 050004F0 | 000E | 0000 |
| A  | 0010              | Entry 10 is identical to entry A  | 0082                                | FFE3 | 0033 | 000A | 00000000 | FFFF | 0000 |
| B  | FFFF              | No matching entries               | 0009                                | 0017 | 0023 | 0003 | 050004F0 | 000F | 0000 |
| C  | 0017              | Entry 17 is identical to entry C  | 006B                                | 0032 | 002E | 0004 | 00000000 | FFFF | 0000 |
| D  | 0012              | Entry 12 is identical to entry D  | 000E                                | FFE3 | FFFE | 000D | 050004F0 | 000C | 0000 |
| E  | FFFF              | No matching entries               | 000E                                | FFE5 | FFDF | 0008 | 050004F0 | 000D | 0000 |
| F  | 0013              | Entry 13 is identical to entry F  | 0077                                | FFE7 | FFCB | 0006 | 00000000 | FFFF | 0000 |
| 10 | 0016              | Entry 16 is identical to entry 10 | 0082                                | FFE3 | 0033 | 000A | 00000000 | FFFF | 0000 |

|    |      |                                   |                      |                    |
|----|------|-----------------------------------|----------------------|--------------------|
| 11 | FFFF | No matching entries               | 000E FFD F 0020 0009 | 050004F0 000E 0000 |
| 12 | FFFF | No matching entries               | 000E FFE3 FFFE 000D  | 050004F0 000C 0000 |
| 13 | 0014 | Entry 14 is identical to entry 13 | 0077 FFE7 FFCB 0006  | 00000000 FFFF 0000 |
| 14 | FFFF | No matching entries               | 0077 FFE7 FFCB 0006  | 00000000 FFFF 0000 |
| 15 | FFFF | No matching entries               | 0060 0017 FFDB 0002  | 00000000 FFFF 0000 |
| 16 | FFFF | No matching entries               | 0082 FFE3 0033 000A  | 00000000 FFFF 0000 |
| 17 | FFFF | No matching entries               | 006B 0032 002E 0004  | 00000000 FFFF 0000 |

When a point is adopted, it is as if that information is copied from the current location to the one given. Data copied in this way is highlighted in yellow above. This makes it possible to sequentially change multiple entries.

Entry 6 is copied four times above. Entry 6 is copied to entry F, the new entry F is copied to entry 13, and this new entry 13 is copied to entry 14. So long as the list is always processed from beginning to end, the table integrity will remain intact.

A single entry's collision entry can be recovered by examining the table. If a matching ID isn't found in the collision table, check for the value in the usage table. Then, check that the new entry isn't also aliased. Here is an example of the method, recovering the collision entry corresponding to entry 0x13.

**Look for entry 0013 in first 0x12 slots...**

0007 FFFF 0005 000B 000C 0015 000F FFFF 000E 0011 0010 FFFF 0017 0012 FFFF 0013 0016 FFFF FFFF 0014

**ID found at entry 0xF.**

**Now, search for 000F in the first 0xE entries...**

0007 FFFF 0005 000B 000C 0015 000F FFFF 000E 0011 0010 FFFF 0017 0012 FFFF 0013

**ID found at entry 0x6.**

**Search again for 0006 in the first 5 entries...**

0007 FFFF 0005 000B 000C 0015 000F

**ID 0006 isn't found, so copy it's data from the collision table.**

**Copy the data in entry 0006 from the collision table!**

0077 FFE7 FFCB 0006 00000000 FFFF 0000

**Done!**

Obtaining matching data from the collision table may not seem useful at the moment. However, it is essential to merging points between different pieces in models correctly.

*Complex Display Lists: merging points*

Point merging involves borrowing points from a different display list. Some models, most notably guards, are designed to be animated during play. To mask the gaps between each section of their body when moving point merging is used, connecting points in one model part to another.

This is a complicated process, involving four steps. Firstly, merging points should be done when translating positions. The reason for this is because the point taken from one part of a model may not be positioned the same as another part of the model. Just placing the point at its final position saves effort later when its location needs to be known.

The first step is determining which point is being replaced, and what point it is being replaced with. To understand how this is done, it is necessary to revisit the collision point table.

X Position

000A

Z Position

0021

Offset to Merge Point Table

0005

Y Position

0001

Index Number

050004F0

Merge Point Number

0010

0000

There are two values in the collision table used to look up and merge points. The *offset to merge table*, when set, indicates the display list that contains the point being taken. The *merge point number* states which point it is in the table. If the merge table pointer is NULL and the merge point number is -1, then no merging occurs. The point is as given.

When an offset appears in the *merge table pointer*, then the point at the specified index is connected physically to another part of the object. The offset will specify a display list command in the *Object Table*. Jump to the offset in the file and retrieve the command.

| Offset | Type     | Data     | Offset | Parent Offset | Next Entry | Previous Entry | Child Offset |
|--------|----------|----------|--------|---------------|------------|----------------|--------------|
| 0x4F0  | 00180000 | 0500356C |        | 050004D8      | 00000000   | 00000000       | 00000000     |

The 18 command shown at the offset is the display list that contains the point being copied. The next step requires getting either the *point table offset* or the *collision point table offset* in the display list at offset 0x356C. It is easier to retrieve positions from the point table than parsing the collision points. Here is the display list data.

| Offset | Mapping  | Offsets  | Point Table | Points / Collision | Col. Table | Usage Table                 |
|--------|----------|----------|-------------|--------------------|------------|-----------------------------|
| 0x356C | 050066F8 | 00000000 | 05003140    | 0026               | 0018       | 050033A0 05003520 0003 0000 |

The idea is to look up a particular point from this table, then copy the data over the original position given in the example above in red. The point number being copied is the *merge point number* highlighted above in purple. In this case, you want the point with ID 0x10. Point 0x10 will be copied from the point table at 0x3140. Only the x, y, and z values are required; copying the s/t values or RGBA can actually cause undesirable effects.

| Offset | Point ID | X Position | Y Position | Z Position |
|--------|----------|------------|------------|------------|
| 0x3240 | 0x10     | 000E       | FFDF       | 0020       |

## Merging Points pt.2: point translation

However, points usually can not be copied as is. In most cases, different parts are positioned separately using the 02 position commands, as mentioned in the previous section. Usually every point in an object model is translated using the object's own position information as outlined in the *point translation* section above. However, the goal of merging points is to borrow a single point from another part of the model, and this part could be – and probably is – positioned differently. To correctly import a point, then, involves also determining its final location.

To properly import points, first the effects of any position commands have to be accounted for. To do this requires taking one more look at the object table command from before.

| Offset | Type     | Data Offset | Parent Offset | Next Entry | Previous Entry | Child Offset |
|--------|----------|-------------|---------------|------------|----------------|--------------|
| 0x4F0  | 00180000 | 0500356C    | 050004D8      | 00000000   | 00000000       | 00000000     |

This time the parent offset is necessary. By gathering a list of all the parent commands, any position commands that influence the display list can be noted. This was outlined in the last section on *parsing the object table*. A breakdown of these is shown below.

01:B8 {02:D0 {02:3B8 {02:448 {02:460 {0A:4C0 {08:4D8 {18:4F0}}}}}}}}

There are four different position commands at work: 02:D0, 02:3B8, 02:448, and 02:460. Point translation is done normally on the point, as outlined in the *point translation* section. For convenience sake, the actual data is listed here

|                | 02:D0 | 02:3B8  | 02:448   | 02:460   | Total Change |
|----------------|-------|---------|----------|----------|--------------|
| X modification | + 0.  | + 1.23  | + 211.71 | + 218.11 | + 431.05     |
| Y Modification | + 0.  | + 62.04 | + 272.09 | + 66.79  | + 400.92     |
| Z Modification | + 0.  | + 0.    | + 0.     | + 8.44   | + 8.44       |

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 000E       | FFDF       | 0020       |
| Original Decimal | 14         | -33        | 32         |
| New Positions    | 445.05     | 367.92     | 40.44      |

These new positions are copied over the original X, Y, and Z values from the point in the original example. Translation will become clearer in the final section, **Example of Object Output**, when an entire point table is translated and an object displayed at its final position. For now, let's focus on the last necessary part of point merging, *replacing point data*.

### Merging Points pt. 3: point replacement

The final step, after translating the point, is to copy the point into the normal point table so the part can be displayed properly. As stated before, merging points is best done when the point table has been or is already being translated, and so this section assumes that every point has already been translated to its final position, ready for rendering. For this reason, all the position data shown below will be decimal values.

Now that the point has been translated, it is time to copy the point from the collision point table into the final point table. The point at the given index number and any other points that share the same location will be replaced with the new data. This has already been touched upon in the section on *point aliasing*, but now it will be practically applied.

The first step is to refer to the *point usage table* to determine what points will be replaced. Match the ID value for the collision point to the same entry in the point usage table. Below is a table illustrating this; note the points given in the table have already been translated.

| #   | Usage Table Entry                    | Corresponding Collision Point Table         |
|-----|--------------------------------------|---|
| 0   | 0007 Entry 7 is identical to entry 0 | 692.85 410.18 -22. 0000 050004F0 0011 0000  |
| 1   | FFFF No matching entries             | 445.05 367.92 40.44 0001 050004F0 0010 0000 |
| ... | Points 3-16                          |   |
| 17  | FFFF No matching entries             | 788.85 451.18 62. 0004 00000000 FFFF 0000   |

Point #1 is highlighted in yellow above. The table shows no other points share the same position, so only point 0001 will be replaced in the rendering point table. If other entries shared the same position, those would be replaced as well.

Now, to replace the matching entry in the point table. Since other entries share the position, only the ID given is changed, point ID #1. Below, highlighted in yellow and already translated to their new positioned, are the original rendering point and the matching collision point.

| #   | Rendering Point Table, Translated          | Collision Point Table, Translated      |
|-----|--|--|
| 0   | 692.85 410.18 -22. 0000 02B1 00DD BCBCBCFF | 692.85 410.18 -22. 0000 050004F0 0011  |
| 1   | 691.85 434.18 21. 0000 0188 003B BCBCBCFF  | 445.05 367.92 40.44 0001 050004F0 0010 |
| ... | Points 2-16                                |  |
| 17  | 788.85 451.18 62. 0000 03B8 0416 F0F0F0FF  |  |

Copy and replace the original point table value with the new, converted point...

| # | New Rendering Point Table, Translated       |
|---|---|
| 1 | 445.05 367.92 40.44 0000 0188 003B BCBCBCFF |

*Example of Object Output:*      *CdjbondZ Right Hand*

This isn't so much an example as a complete conversion of the right hand from Bond's dinner jacket model. The point positions are translated from their current positions to their final ones, all shared points merged, and the final model rendered. This example focuses on just one display list command, found at offset 0x4A8.

| Offset | Type     | Data Offset | Parent Offset | Next Entry | Previous Entry | Child Offset |
|--------|----------|-------------|---------------|------------|----------------|--------------|
| 0x100  | 00180000 | 050030F0    | 05000490      | 00000000   | 00000000       | 00000000     |

First, before anything else can be done, all the position commands affecting the part have to be determined. This is as simple as finding all the parent commands affecting the display list command. Retrieve the parent offset from the display list, then that command's parent and so on until the first command with no parent is found. This was described in the section on *parsing the object table*. Listed below are all the commands that affect the display list.

01:B8 {02:D0 {02:3B8 {02:448 {02:460 {02:478 {0A:490 {18:4A8}}}}}}}}

As mentioned before, the value before the colon is the command type, and the value after the colon is the offset to the command. Five of the commands are 02 position types. Read each of these command's data pointers, then copy the position data found at that offset. A table of the values, converted to their decimal equivalents, is listed below.

|                | 02:D0 | 02:3B8  | 02:448   | 02:460   | 02:478  | Total Change |
|----------------|-------|---------|----------|----------|---------|--------------|
| X modification | + 0.  | + 1.23  | + 211.71 | + 218.11 | + 250.8 | + 681.85     |
| Y Modification | + 0.  | + 62.04 | + 272.09 | + 66.79  | + 0.26  | + 401.18     |
| Z Modification | + 0.  | + 0.    | + 0.     | + 8.44   | + 7.56  | + 16.0       |

The total change done to each point in the point table is equal to the sum of each of the position commands affecting it. The total change is given in the blue column to the very right.

The next step involves looking at the 18 command's data. Jump to the offset given in the command, 0x30F0. It contains the offsets to the point tables, usage table, and rendering commands. Right now, you need the *point table offset*. There are 24 (0x18) points in the table total.

| Offset | Mapping Offsets   | Point Table | Points / Collision | Col. Table        | Usage Table |
|--------|-------------------|-------------|--------------------|-------------------|-------------|
| 0x356C | 05006650 00000000 | 05002EA0    | 0018 000A          | 05003020 050030C0 | 0003 0000   |

Now, every point in the rendering point table is moved to the final position using the **total change** value. Simply add the change in x value to each point's x value, the change in y to the y values, and change in z to z values. This sets each point to their final positions. Notice in this table, only the x, y, & z values are shown, and each is converted to its decimal equivalent.

| #  | Org. X | Org. Y | Org. Z | New X  | New Y  | New Z |
|----|--------|--------|--------|--------|--------|-------|
| 0  | 11     | 9      | -38    | 692.85 | 410.85 | -22   |
| 1  | 10     | 33     | 5      | 691.85 | 434.18 | 21    |
| 2  | 96     | 23     | -37    | 777.85 | 424.18 | -21   |
| 3  | 9      | 23     | 35     | 690.85 | 424.18 | 51    |
| 4  | 107    | 50     | 46     | 788.85 | 451.18 | 62    |
| 5  | 96     | 23     | -37    | 777.85 | 424.18 | -21   |
| 6  | 119    | -25    | -53    | 800.85 | 376.18 | -37   |
| 7  | 11     | 9      | -38    | 692.85 | 410.85 | -22   |
| 8  | 14     | -27    | -33    | 695.85 | 374.18 | -17   |
| 9  | 14     | -33    | 32     | 695.85 | 368.18 | 48    |
| A  | 130    | -29    | 51     | 811.85 | 372.18 | 67    |
| B  | 9      | 23     | 35     | 690.85 | 424.18 | 51    |
| C  | 107    | 50     | 46     | 788.85 | 451.18 | 62    |
| D  | 14     | -29    | -2     | 695.85 | 372.18 | 14    |
| E  | 14     | -27    | -33    | 695.85 | 374.18 | -17   |
| F  | 119    | -25    | -53    | 800.85 | 376.18 | -37   |
| 10 | 130    | -29    | 51     | 811.85 | 372.18 | 67    |
| 11 | 14     | -33    | 32     | 695.85 | 368.18 | 48    |
| 12 | 14     | -29    | -2     | 695.85 | 372.18 | 14    |
| 13 | 119    | -25    | -53    | 800.85 | 376.18 | -37   |
| 14 | 119    | -25    | -53    | 800.85 | 376.18 | -37   |
| 15 | 96     | 23     | -37    | 777.85 | 424.18 | -21   |
| 16 | 130    | -29    | 51     | 811.85 | 372.18 | 67    |
| 17 | 107    | 50     | 46     | 788.85 | 451.18 | 62    |

The columns on the right show the final, transcribed positions. When the model is rendered at these locations, it will appear at the correct position relative to the rest of the body.

Now that all the rendering points are transcribed, points can be merged. To do this requires a look at the collision point table. There are only 10 point entries in this table. Any of those with a NULL merge offset can be ignored. Only those highlighted in yellow below are going to be used; the rest are not merged positions. Each one will be done individually.

| X Position | Y Position | Z Position | Index | Merge Offset | Merge # |      |
|------------|------------|------------|-------|--------------|---------|------|
| 000B       | 0009       | FFDA       | 0000  | 050004F0     | 0011    | 0000 |
| 000A       | 0021       | 0005       | 0001  | 050004F0     | 0010    | 0000 |
| 0060       | 0017       | FFDB       | 0002  | 00000000     | FFFF    | 0000 |
| 0009       | 0017       | 0023       | 0003  | 050004F0     | 000F    | 0000 |
| 006B       | 0032       | 002E       | 0004  | 00000000     | FFFF    | 0000 |
| 0077       | FFE7       | FFCB       | 0006  | 00000000     | FFFF    | 0000 |
| 000E       | FFE5       | FFDF       | 0008  | 050004F0     | 000D    | 0000 |
| 000E       | FFDF       | 0020       | 0009  | 050004F0     | 000E    | 0000 |
| 0082       | FFE3       | 0033       | 000A  | 00000000     | FFFF    | 0000 |
| 000E       | FFE3       | FFFE       | 000D  | 050004F0     | 000C    | 0000 |

Notice all of the selected merged points have the same merge offset. This is rather handy, since they'll all draw from the same point table. Right now lets take the first entry with a merge offset, the one with index 0000. You need point #11 from the table in 0x4F0. First, we'll extract the point from the proper display list entry. Jump to the command at offset 0x4F0. This is the same block the original *point merging* example referred to.

| Offset | Type     | Data Offset | Parent Offset | Next Entry | Previous Entry | Child Offset |
|--------|----------|-------------|---------------|------------|----------------|--------------|
| 0x4F0  | 00180000 | 0500356C    | 050004D8      | 00000000   | 00000000       | 00000000     |

Highlighted in blue is the offset to the display list data. In yellow is the parent offset, in order to find all the position entries that affect the point. This will be done first. You could translate every point table's entries before attempting to merge points, but in this example it is assumed only the current block has been translated. As described before, retrieve a list of all the parent commands, then filter out the ones that set part positions.

01:B8 {02:D0 {02:3B8 {02:448 {02:460 {0A:4C0 {08:4D8 {18:4F0}}}}}}}}

There are four different position commands at work: 02:D0, 02:3B8, 02:448, and 02:460. Point translation is done normally on the point, as outlined in the *point translation* section. For convenience sake, the actual data is listed here

|                | 02:D0 | 02:3B8  | 02:448   | 02:460   | Total Change |
|----------------|-------|---------|----------|----------|--------------|
| X modification | + 0.  | + 1.23  | + 211.71 | + 218.11 | + 431.05     |
| Y Modification | + 0.  | + 62.04 | + 272.09 | + 66.79  | + 400.92     |
| Z Modification | + 0.  | + 0.    | + 0.     | + 8.44   | + 8.44       |



Once the point in question has been extracted, the values in the *total change* column are added to the point, converting it to its final position. To extract the point, jump to the display list data offset at 0x356C – the one highlighted in blue in the table above.

| Offset | Mapping Offsets |          | Point Table | Points / Collision |      | Col. Table | Usage Table |           |
|--------|-----------------|----------|-------------|--------------------|------|------------|-------------|-----------|
| 0x356C | 05006650        | 00000000 | 05002EA0    | 0018               | 000A | 05003020   | 050030C0    | 0003 0000 |

You need to get the 11<sup>th</sup> point from the point table. Every point in the table is 16 (0x10) bytes long, so the address of the point you need is at 0x110 + 0x2EA0, or 0x2FB0. Retrieve the x, y, and z values of this point...

000E FFDF 0020

–and add the total position modifier from the *total change* column for this part.

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 000E       | FFDF       | 0020       |
| Original Decimal | 14         | -33        | 32         |
| New Positions    | 445.05     | 367.92     | 40.44      |

The new position will replace the x, y, and z of any points set to the original location. In order to do replacement, we need to look at the point usage table for the original point. Any entries that share the same point as index 0 will also be moved. The point usage table is at 0x30C0. Each entry is two bytes, so each entry can be looked up by its ID.

$2 * ID + 0x30C0 = \text{offset to matching entry}$

**Look up entry 0 at offset 0x30C0...**

0x30C0 0007 ...

**This one equals 0007, so point #7 is at the same location**

**Now, check entry 0007 at 0x30CE...**

0x30C0 0007 FFFF 0005 000B 000C 0015 000F FFFF ...

**If the entry equals FFFF, then no other points share that location.**

**Points 0 and 7 will both be replaced since both are at the same location.**

Both points 0 and 7 in the original point table will be replaced with the new values. Now, a quick look at the point table so far. Each entries coordinates have been translated to the new position. Entries 0 and 7, highlighted in purple, have both been replaced with their new, merged positions.

| #  | New X  | New Y  | New Z | -    | S img | T img | RGBA     |
|----|--------|--------|-------|------|-------|-------|----------|
| 0  | 445.05 | 367.92 | 40.44 | 0000 | 02B1  | 00DD  | BCBCBCFF |
| 1  | 691.85 | 434.18 | 21    | 0000 | 0188  | 003B  | BCBCBCFF |
| 2  | 777.85 | 424.18 | -21   | 0000 | 0287  | 032C  | F0F0F0FF |
| 3  | 690.85 | 424.18 | 51    | 0000 | 00BD  | FFCD  | D4D4D4FF |
| 4  | 788.85 | 451.18 | 62    | 0000 | 0047  | 0252  | F0F0F0FF |
| 5  | 777.85 | 424.18 | -21   | 0000 | 00A1  | 0231  | F0F0F0FF |
| 6  | 800.85 | 376.18 | -37   | 0000 | 0325  | 03A9  | F0F0F0FF |
| 7  | 445.05 | 367.92 | 40.44 | 0000 | 015D  | FFE9  | BCBCBCFF |
| 8  | 695.85 | 374.18 | -17   | 0000 | 0336  | 0094  | D4D4D4FF |
| 9  | 695.85 | 368.18 | 48    | 0000 | 02A3  | 038B  | D4D4D4FF |
| A  | 811.85 | 372.18 | 67    | 0000 | 03D1  | 0028  | F0F0F0FF |
| B  | 690.85 | 424.18 | 51    | 0000 | 0026  | 0340  | D4D4D4FF |
| C  | 788.85 | 451.18 | 62    | 0000 | 000D  | 003D  | F0F0F0FF |
| D  | 695.85 | 372.18 | 14    | 0000 | 026D  | 0018  | D4D4D4FF |
| E  | 695.85 | 374.18 | -17   | 0000 | 0143  | 002B  | D4D4D4FF |
| F  | 800.85 | 376.18 | -37   | 0000 | FFFF  | 028E  | F0F0F0FF |
| 10 | 811.85 | 372.18 | 67    | 0000 | 03DA  | 028F  | F0F0F0FF |
| 11 | 695.85 | 368.18 | 48    | 0000 | 03AF  | 0002  | D4D4D4FF |
| 12 | 695.85 | 372.18 | 14    | 0000 | 026D  | 0018  | D4D4D4FF |
| 13 | 800.85 | 376.18 | -37   | 0000 | FFFF  | 028E  | F0F0F0FF |
| 14 | 800.85 | 376.18 | -37   | 0000 | FFE9  | 021E  | F0F0F0FF |
| 15 | 777.85 | 424.18 | -21   | 0000 | 00A5  | 0348  | F0F0F0FF |
| 16 | 811.85 | 372.18 | 67    | 0000 | 03B1  | 023E  | F0F0F0FF |
| 17 | 788.85 | 451.18 | 62    | 0000 | 03B8  | 0416  | F0F0F0FF |

The next point to do is index 0001, calling for point #10 from the table in 0x4F0. This is the same table as before, so it isn't necessary to recompute the position modifiers and get the offsets – it has already been done before. Read the x, y, and z values from the 10<sup>th</sup> point in 0x2EA0: 0x100 + 0x2EA0, or 0x2FA0. After retrieving the point, apply the position modifier.

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 0082       | FFE3       | 0033       |
| Original Decimal | 130        | -29        | 51         |
| Modifier         | +431.05    | +400.92    | +8.44      |
| New Positions    | 561.05     | 371.92     | 59.44      |

Now, to check the point usage table. Again, the table is found at 0x30C0. This time, index 0001 is going to be checked. You can determine the address by multiplying the index by 2 and adding it to the table offset.

0x30C2    FFFF

No other points share this position, so only point #1 will be replaced. The newly changed value is highlighted in purple in the table below, and those changed previously in blue. Again, the point list as it stands now. Only four more points to go...

| #  | New X  | New Y  | New Z | -    | S img | T img | RGBA     |
|----|--------|--------|-------|------|-------|-------|----------|
| 0  | 445.05 | 367.92 | 40.44 | 0000 | 02B1  | 00DD  | BCBCBCFF |
| 1  | 561.05 | 371.92 | 59.44 | 0000 | 0188  | 003B  | BCBCBCFF |
| 2  | 777.85 | 424.18 | -21   | 0000 | 0287  | 032C  | F0F0F0FF |
| 3  | 690.85 | 424.18 | 51    | 0000 | 00BD  | FFCD  | D4D4D4FF |
| 4  | 788.85 | 451.18 | 62    | 0000 | 0047  | 0252  | F0F0F0FF |
| 5  | 777.85 | 424.18 | -21   | 0000 | 00A1  | 0231  | F0F0F0FF |
| 6  | 800.85 | 376.18 | -37   | 0000 | 0325  | 03A9  | F0F0F0FF |
| 7  | 445.05 | 367.92 | 40.44 | 0000 | 015D  | FFE9  | BCBCBCFF |
| 8  | 695.85 | 374.18 | -17   | 0000 | 0336  | 0094  | D4D4D4FF |
| 9  | 695.85 | 368.18 | 48    | 0000 | 02A3  | 038B  | D4D4D4FF |
| A  | 811.85 | 372.18 | 67    | 0000 | 03D1  | 0028  | F0F0F0FF |
| B  | 690.85 | 424.18 | 51    | 0000 | 0026  | 0340  | D4D4D4FF |
| C  | 788.85 | 451.18 | 62    | 0000 | 000D  | 003D  | F0F0F0FF |
| D  | 695.85 | 372.18 | 14    | 0000 | 026D  | 0018  | D4D4D4FF |
| E  | 695.85 | 374.18 | -17   | 0000 | 0143  | 002B  | D4D4D4FF |
| F  | 800.85 | 376.18 | -37   | 0000 | FFFF  | 028E  | F0F0F0FF |
| 10 | 811.85 | 372.18 | 67    | 0000 | 03DA  | 028F  | F0F0F0FF |
| 11 | 695.85 | 368.18 | 48    | 0000 | 03AF  | 0002  | D4D4D4FF |
| 12 | 695.85 | 372.18 | 14    | 0000 | 026D  | 0018  | D4D4D4FF |
| 13 | 800.85 | 376.18 | -37   | 0000 | FFFF  | 028E  | F0F0F0FF |
| 14 | 800.85 | 376.18 | -37   | 0000 | FFE9  | 021E  | F0F0F0FF |
| 15 | 777.85 | 424.18 | -21   | 0000 | 00A5  | 0348  | F0F0F0FF |
| 16 | 811.85 | 372.18 | 67    | 0000 | 03B1  | 023E  | F0F0F0FF |
| 17 | 788.85 | 451.18 | 62    | 0000 | 03B8  | 0416  | F0F0F0FF |

The next entry being merged is index 0003, requesting point 0xF from display list command 0x4F0. Do this point just like the last two, computing position and retrieving the point from the point table. Since the position is the same as the last two, compute the offset to the point.  $0xF0 + 0x2EA0 = 0x2F90$ . The point data is shown below, and the position modification is applied.

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 0077       | FFE7       | FFCB       |
| Original Decimal | 119        | -25        | -53        |
| Modifier         | +431.05    | +400.92    | +8.44      |
| New Positions    | 550.05     | 375.92     | -44.56     |

Now, to check the point usage table again. You want entry 3, which is found at  $2*3 + 0x30C0 = 0x30C6$ .

**Read the third entry at 0x30C6...**

**0x30C0 0007 FFFF 0005 000B .**

**Entry 000B shares the same position.**

**Read entry 000B to see if any more entries share the location**

**0x30C0 0007 FFFF 0005 000B 000C 0015 000F FFFF 000E 0011 0010 FFFF .**

**The entry reads -1, so no other points share the position.**

**Points 3 and B will be replaced since they are at the same location.**

Now, replace both entries 3 and B in the point table with the new entries. The current list is below, and again, the new entries are in purple and other replaced entries in blue.

| #  | New X  | New Y  | New Z  | -    | S img | T img | RGBA     |
|----|--------|--------|--------|------|-------|-------|----------|
| 0  | 445.05 | 367.92 | 40.44  | 0000 | 02B1  | 00DD  | BCBCBCFF |
| 1  | 561.05 | 371.92 | 59.44  | 0000 | 0188  | 003B  | BCBCBCFF |
| 2  | 777.85 | 424.18 | -21    | 0000 | 0287  | 032C  | F0F0F0FF |
| 3  | 550.05 | 375.92 | -44.56 | 0000 | 00BD  | FFCD  | D4D4D4FF |
| 4  | 788.85 | 451.18 | 62     | 0000 | 0047  | 0252  | F0F0F0FF |
| 5  | 777.85 | 424.18 | -21    | 0000 | 00A1  | 0231  | F0F0F0FF |
| 6  | 800.85 | 376.18 | -37    | 0000 | 0325  | 03A9  | F0F0F0FF |
| 7  | 445.05 | 367.92 | 40.44  | 0000 | 015D  | FFE9  | BCBCBCFF |
| 8  | 695.85 | 374.18 | -17    | 0000 | 0336  | 0094  | D4D4D4FF |
| 9  | 695.85 | 368.18 | 48     | 0000 | 02A3  | 038B  | D4D4D4FF |
| A  | 811.85 | 372.18 | 67     | 0000 | 03D1  | 0028  | F0F0F0FF |
| B  | 550.05 | 375.92 | -44.56 | 0000 | 0026  | 0340  | D4D4D4FF |
| C  | 788.85 | 451.18 | 62     | 0000 | 000D  | 003D  | F0F0F0FF |
| D  | 695.85 | 372.18 | 14     | 0000 | 026D  | 0018  | D4D4D4FF |
| E  | 695.85 | 374.18 | -17    | 0000 | 0143  | 002B  | D4D4D4FF |
| F  | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 10 | 811.85 | 372.18 | 67     | 0000 | 03DA  | 028F  | F0F0F0FF |
| 11 | 695.85 | 368.18 | 48     | 0000 | 03AF  | 0002  | D4D4D4FF |
| 12 | 695.85 | 372.18 | 14     | 0000 | 026D  | 0018  | D4D4D4FF |
| 13 | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 14 | 800.85 | 376.18 | -37    | 0000 | FFE9  | 021E  | F0F0F0FF |
| 15 | 777.85 | 424.18 | -21    | 0000 | 00A5  | 0348  | F0F0F0FF |
| 16 | 811.85 | 372.18 | 67     | 0000 | 03B1  | 023E  | F0F0F0FF |
| 17 | 788.85 | 451.18 | 62     | 0000 | 03B8  | 0416  | F0F0F0FF |

The next point being merged is index 0008, retrieving point 0xD from the command 0x4F0. Again, since the same table is being used, the position and offsets are the same as before. Calculate the offset to the data, copy it, then apply the position change. The point is found at 0xD0 + 0x2EA0 = 0x2F70.

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 000E       | FFE3       | FFFE       |
| Original Decimal | 14         | -29        | -2         |
| Modifier         | +431.05    | +400.92    | +8.44      |
| New Positions    | 445.05     | 371.92     | 6.44       |

Now, check the point usage table to see if any other points in the table share the same location.

Read entry 8 from the point usage table ...  
 0x30C0 0007 FFFF 0005 000B 000C 0015 000F FFFF 000E .

Entry 000E shares the same position.  
 Read entry 000E to see if any other entries are at that location  
 The previous entries (0-7) aren't shown.  
 0x30D0 . 000E 0011 0010 FFFF 0017 0012 FFFF .

The entry reads -1, so not other points share the position.  
 Entries 8 and E will be replaced.

Now simply replace both points 8 and E with the new position computed above. The new table is listed below with the new points highlighted in purple.

| #  | New X  | New Y  | New Z  | -    | S img | T img | RGBA     |
|----|--------|--------|--------|------|-------|-------|----------|
| 0  | 445.05 | 367.92 | 40.44  | 0000 | 02B1  | 00DD  | BCBCBCFF |
| 1  | 561.05 | 371.92 | 59.44  | 0000 | 0188  | 003B  | BCBCBCFF |
| 2  | 777.85 | 424.18 | -21    | 0000 | 0287  | 032C  | F0F0F0FF |
| 3  | 550.05 | 375.92 | -44.56 | 0000 | 00BD  | FFCD  | D4D4D4FF |
| 4  | 788.85 | 451.18 | 62     | 0000 | 0047  | 0252  | F0F0F0FF |
| 5  | 777.85 | 424.18 | -21    | 0000 | 00A1  | 0231  | F0F0F0FF |
| 6  | 800.85 | 376.18 | -37    | 0000 | 0325  | 03A9  | F0F0F0FF |
| 7  | 445.05 | 367.92 | 40.44  | 0000 | 015D  | FFE9  | BCBCBCFF |
| 8  | 445.85 | 371.92 | 6.44   | 0000 | 0336  | 0094  | D4D4D4FF |
| 9  | 695.85 | 368.18 | 48     | 0000 | 02A3  | 038B  | D4D4D4FF |
| A  | 811.85 | 372.18 | 67     | 0000 | 03D1  | 0028  | F0F0F0FF |
| B  | 550.05 | 375.92 | -44.56 | 0000 | 0026  | 0340  | D4D4D4FF |
| C  | 788.85 | 451.18 | 62     | 0000 | 000D  | 003D  | F0F0F0FF |
| D  | 695.85 | 372.18 | 14     | 0000 | 026D  | 0018  | D4D4D4FF |
| E  | 445.85 | 371.92 | 6.44   | 0000 | 0143  | 002B  | D4D4D4FF |
| F  | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 10 | 811.85 | 372.18 | 67     | 0000 | 03DA  | 028F  | F0F0F0FF |
| 11 | 695.85 | 368.18 | 48     | 0000 | 03AF  | 0002  | D4D4D4FF |
| 12 | 695.85 | 372.18 | 14     | 0000 | 026D  | 0018  | D4D4D4FF |
| 13 | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 14 | 800.85 | 376.18 | -37    | 0000 | FFE9  | 021E  | F0F0F0FF |
| 15 | 777.85 | 424.18 | -21    | 0000 | 00A5  | 0348  | F0F0F0FF |
| 16 | 811.85 | 372.18 | 67     | 0000 | 03B1  | 023E  | F0F0F0FF |
| 17 | 788.85 | 451.18 | 62     | 0000 | 03B8  | 0416  | F0F0F0FF |

Only two points remain. The first is index 9, drawing point 0xE from the 0x4F0 command. Compute the offset to the point, copy it, and apply the position modifier. The point is located at 0xE0 + 0x2EA0, or 0x2F80. The data is shown in the table below.

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 000E       | FFE5       | FFDF       |
| Original Decimal | 14         | -27        | -33        |
| Modifier         | +431.05    | +400.92    | +8.44      |
| New Positions    | 445.05     | 373.92     | -24.56     |

Now check entry 9 in the *point usage table*. The offset is  $2 \times 9 + 0x30C0$ , or 0x30D2.

**Retrieve entry 9 from offset 0x30D2** \_\_\_\_\_  
 0x30C0 0007 FFFF 0005 000B 000C 0015 000F FFFF 000E 0011 .

**Entry 11 also shares the location.**  
**Retrieve entry 11's value to see if other points are identical.**  
**The previous 9 entries are not shown.** |

0x30D2 . 0011 0010 FFFF 0017 0012 FFFF 0013 0016 FFFF .

**The value equals -1, so no other entries need testing.**

**Points 9 and 11 will be replaced with the new data.**

Before showing the updated table the final merged point will be handled. This uses index 000D and calls for point C from the 0x4F0 display list. Again, since this is the same binary as the rest of the points, all the same positions and offsets are in effect. Draw the point from the calculated offset:  $0xC0 + 0x2EA0 = 0x2F60$

|                  | X Position | Y Position | Z Position |
|------------------|------------|------------|------------|
| Original Hex     | 006B       | 0032       | 002E       |
| Original Decimal | 107        | 50         | 46         |
| Modifier         | +431.05    | +400.92    | +8.44      |
| New Positions    | 538.05     | 450.92     | 54.44      |

Look up entry D in the point usage table. Replace all the points that share this new position.

Retrieve entry D from 0x30DA

0x30C0 0007 FFFF 0005 000B 000C 0015 000F FFFF 000E 0011 0010 FFFF 0017 0012 .

The point is shared by entry 12.

Look up entry 12 to see if any other points are at the same position...

The first entries are not listed below

0x30DA . 0012 FFFF 0013 0016 FFFF FFFF .

-1 means the point isn't shared by another in the table.

Entries D and 12 are replaced with the new position data.

Now, just replace the points in the rendering table with the new ones. Below is the final point table. You are now ready to render the object, and with the updated point data this part will render at exactly the correct position relative to the rest of the model.

In conclusion, point merging is slightly complicated but by no means difficult. You apply the same technique to every part of the model. Scan through the point usage table for a merge pointer. Use that pointer to retrieve a list of its parent commands to determine the position mod to apply to the point. Use its data offset to retrieve the correct merge point number, then copy the result to every identical location in the original part.

-Zoinkity

| #  | New X  | New Y  | New Z  | -    | S img | T img | RGBA     |
|----|--------|--------|--------|------|-------|-------|----------|
| 0  | 445.05 | 367.92 | 40.44  | 0000 | 02B1  | 00DD  | BCBCBCFF |
| 1  | 561.05 | 371.92 | 59.44  | 0000 | 0188  | 003B  | BCBCBCFF |
| 2  | 777.85 | 424.18 | -21    | 0000 | 0287  | 032C  | F0F0F0FF |
| 3  | 550.05 | 375.92 | -44.56 | 0000 | 00BD  | FFCD  | D4D4D4FF |
| 4  | 788.85 | 451.18 | 62     | 0000 | 0047  | 0252  | F0F0F0FF |
| 5  | 777.85 | 424.18 | -21    | 0000 | 00A1  | 0231  | F0F0F0FF |
| 6  | 800.85 | 376.18 | -37    | 0000 | 0325  | 03A9  | F0F0F0FF |
| 7  | 445.05 | 367.92 | 40.44  | 0000 | 015D  | FFE9  | BCBCBCFF |
| 8  | 445.85 | 371.92 | 6.44   | 0000 | 0336  | 0094  | D4D4D4FF |
| 9  | 445.05 | 373.92 | -24.56 | 0000 | 02A3  | 038B  | D4D4D4FF |
| A  | 811.85 | 372.18 | 67     | 0000 | 03D1  | 0028  | F0F0F0FF |
| B  | 550.05 | 375.92 | -44.56 | 0000 | 0026  | 0340  | D4D4D4FF |
| C  | 788.85 | 451.18 | 62     | 0000 | 000D  | 003D  | F0F0F0FF |
| D  | 538.05 | 450.92 | 54.44  | 0000 | 026D  | 0018  | D4D4D4FF |
| E  | 445.85 | 371.92 | 6.44   | 0000 | 0143  | 002B  | D4D4D4FF |
| F  | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 10 | 811.85 | 372.18 | 67     | 0000 | 03DA  | 028F  | F0F0F0FF |
| 11 | 445.05 | 373.92 | -24.56 | 0000 | 03AF  | 0002  | D4D4D4FF |
| 12 | 538.05 | 450.92 | 54.44  | 0000 | 026D  | 0018  | D4D4D4FF |
| 13 | 800.85 | 376.18 | -37    | 0000 | FFFF  | 028E  | F0F0F0FF |
| 14 | 800.85 | 376.18 | -37    | 0000 | FFE9  | 021E  | F0F0F0FF |
| 15 | 777.85 | 424.18 | -21    | 0000 | 00A5  | 0348  | F0F0F0FF |
| 16 | 811.85 | 372.18 | 67     | 0000 | 03B1  | 023E  | F0F0F0FF |
| 17 | 788.85 | 451.18 | 62     | 0000 | 03B8  | 0416  | F0F0F0FF |