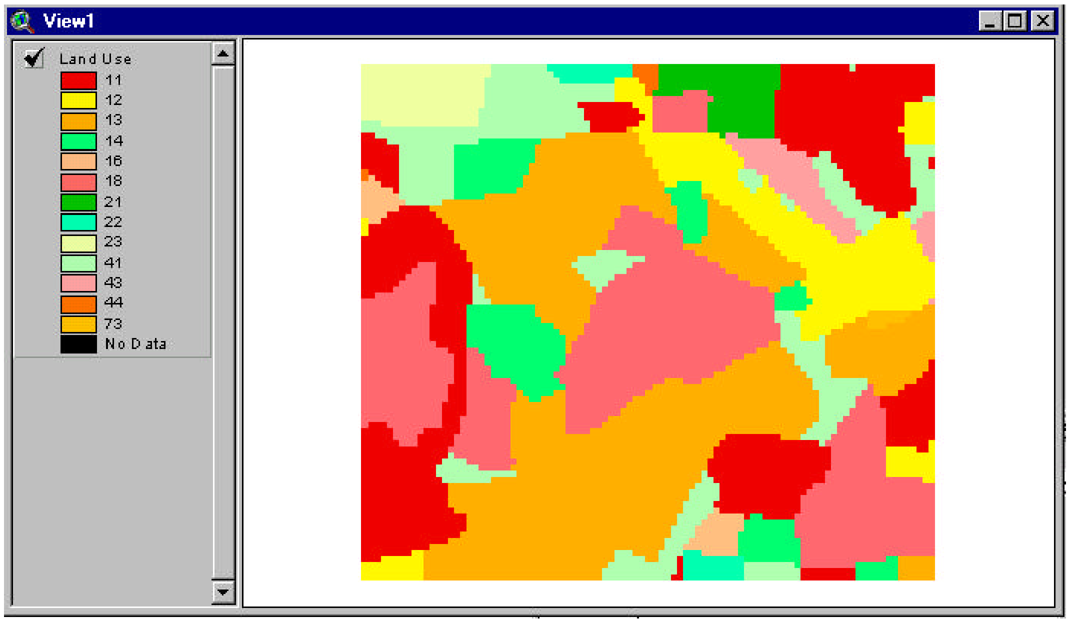
Working With Tables in ArcView

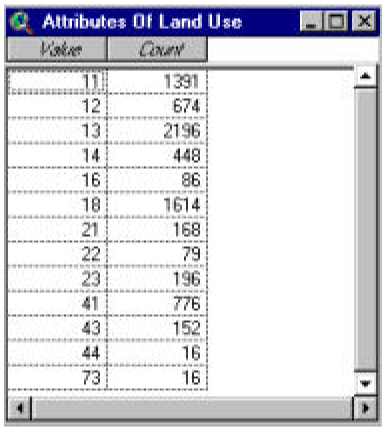
As with Views and Scripts, Tables represent another document type in ArcView. The concept of a table is what really separates a GIS from just a plain paper map. Imagine that you are looking for a particular street on a roadmap. The extent of the roadmap is such that it covers all the roads within a particular county. If you are not familiar with that county, you could wind up spending a long time looking for that road. You're probably thinking, "Well, I could just use the map index to narrow down the location of that road." But you're assuming the roadmap has an index. By making such an assumption, you've made that road map more than just a plain paper map. The index of the roadmap functions much like a table does is ArcView. The table contains additional (non-graphic) information about a theme. In the case of our roadmap, the corresponding roadmap table might contain information like road location, road length, road category (e.g. residential, primary, secondary, etc.), and road name. The table acts as a database for the theme that appears in the view.

To really take advantage of the power of GIS, you need to understand how tables work and what you can do with them. This handout will illustrate some applications of tables especially as they apply to grid themes.

Let's say you had a grid showing land use over an area such as the one shown below:



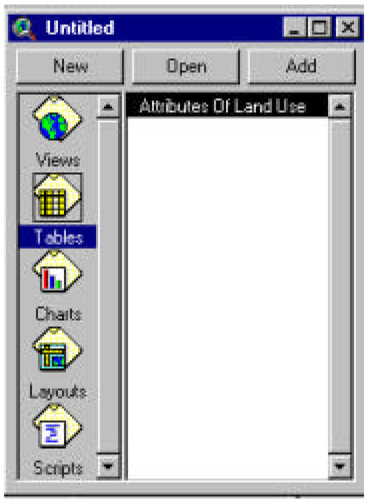
If you click on the "Open Theme Table" button, or select the menu choice "Theme: Table…" a new window will appear that looks like the one at top left on the next page:

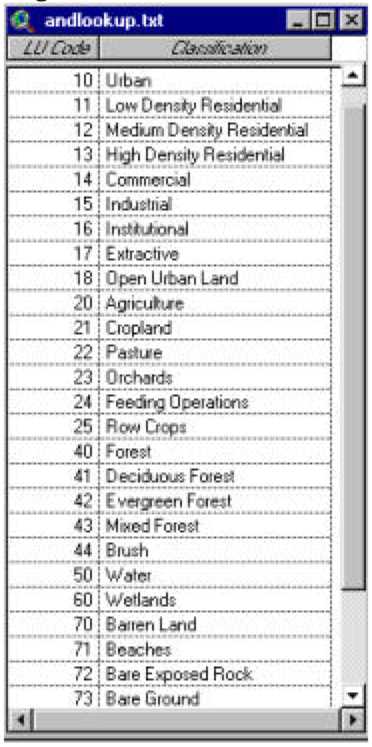


Value	Count
11	1391
12	674
13	2196
14	448
16	86
18	1614
21	168
22	79
23	196
41	776
43	152
44	16
73	16

This table is typical of what you would expect to see for a grid describing categorical data. You can tell it is associated with a grid because grid tables typically have tables consisting of two columns (called "Fields"). The first field has the name "Value" which corresponds to the values that appear in the grid's legend as displayed in the view. The second field has the name "Count" is simply a tally of the number of pixels of that value (category) within the grid. Each row in the table is called a "Record". Looking at the first record in the table at right we say a "Value" of 11 appears 1391 times in this particular grid. If we sum the "Counts" over all records, that sum should equal the total number of pixels in the grid that contain data other than "No Data" pixels.

By now, you are probably wondering what the different categories correspond to. The land use values above come from a standard numbering scheme (see Anderson et al….). As luck would have it, we have an external text file in tabular format (i.e. items in a given line are tab-delimited with a carriage return at the end of each line). We can go to the project window (shown at left) and load this table into the ArcView environment by first either double-clicking on the "Tables" icon of the Project window, or by clicking on the "Add" button of the project window with the "Tables" icon selected. In either case, a file browser dialog box will appear and you will need to specify the name of the external file. In our case, this external file has the name "**`andlookup.txt`**". The result of this load operation produces a new window showing the contents of the external table as shown at right. Quick inspection indicates that the field called "LU Code" in the external table has categories with numerical values that match the "Value" field of our "Attributes of Land Use" table accompanying the "Land Use" grid theme. This is all very nice, but wouldn't it be better if all the information was located in a single table? ArcView makes this desire an easy one to accomplish.

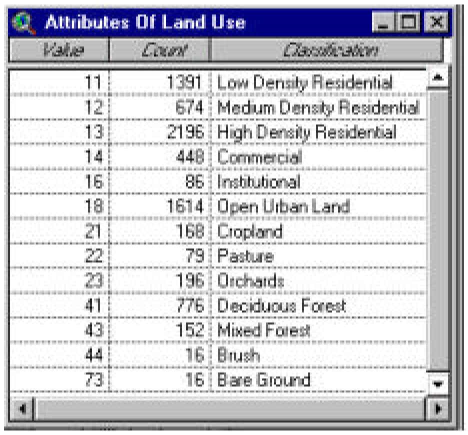




LU Code	Classification
10	Urban
11	Low Density Residential
12	Medium Density Residential
13	High Density Residential
14	Commercial
15	Industrial
16	Institutional
17	Extractive
18	Open Urban Land
20	Agriculture
21	Cropland
22	Pasture
23	Orchards
24	Feeding Operations
25	Row Crops
40	Forest
41	Deciduous Forest
42	Evergreen Forest
43	Mixed Forest
44	Brush
50	Water
60	Wetlands
70	Barren Land
71	Beaches
72	Bare Exposed Rock
73	Bare Ground

<u>To join two tables together:</u>

1. You must first have two separate tables, each with a common field that you wish to join on. ("Value" and "LU Code" are the common fields in this example.)
2. Click on the common field name in the table you want to appear as the right hand portion of the new overall table. ("LU Code" in this example.)
3. Click on the common field name of the other table. This table will appear as the left had portion of the new overall table. ("Value" in this example.)
4. Choose "Table: Join" from the menu.



Value	Count	Classification
11	1391	Low Density Residential
12	674	Medium Density Residential
13	2196	High Density Residential
14	448	Commercial
16	86	Institutional
18	1614	Open Urban Land
21	168	Cropland
22	79	Pasture
23	196	Orchards
41	776	Deciduous Forest
43	152	Mixed Forest
44	16	Brush
73	16	Bare Ground

In the given example, you will note that the "Attributes of Land Use" table has changed to reflect the join that has just taken place. The new table is shown at right. You will note that the "LU Code" field has disappeared while the "Value" field remains. The record that previously had a "LU Code" of 11 (Low Density Residential), now has a "Value" of 11. Records with "LU Codes" not observed in the Land Use "Values" have disappeared (e.g. "LU Code" equal to 50, corresponding to a classification of "Water" does not appear in the updated "Attributes of Land Use" theme. You should note that as demonstrated here, table joining has a directionality associated with it. If we had joined the tables in the opposite order, the "Value" field would have disappeared, there would have been empty entries for the "Count" associated with categories such as "Water", and the order of the fields (from left to right) would have been "LU Code", "Classification", and "Count". Further, the updated table would have been the one corresponding to "**`andlookup.txt`**". Although we've illustrated a single join of two tables, it is possible to perform multiple joins across many tables (two at a time), provided there is always a common field being shared by the joined tables.

<u>Joining Tables in a Script</u>

Let's now look at this process from the perspective of scripting in Avenue. The following script performs the same join as we have just performed manually above.

```
theView = av.FindDoc("View1")

LUgrid = theView.FindTheme("Land Use").GetGrid
ClassFile = "andlookup.txt"

LookUpTable = av.FindDoc(ClassFile).GetVTab
LookUpField = LookUpTable.FindField("LU code")
LandUseTable = LUgrid.GetVTab
LandUseField = LandUseTable.FindField("Value")
LandUseTable.Join(LandUseField, LookUpTable, LookUpField)
```

There are three commands in the above script that should be new to you. The first command uses "**`GetVTab`**":

```
LookUpTable = av.FindDoc(ClassFile).GetVTab
```

or

```
LandUseTable = LUgrid.GetVTab
```

These lines assign the variables **`LookUpTable`** and **`LandUseTable`** with the contents of the tables associated with "**`andlookup.txt`**" and the Land Use grid, respectively. (It is encouraged that you check the Help files for more information on the syntax of this command. You might also want to look at the command **`GetFTab`** which gets the table associated with a Feature Theme, such as the familiar "**`pg_roads.shp`**".)

The second new command should look familiar to you, it is much like the command to find a particular theme in the view, but in this case we are finding a particular field in a table:

```
LookUpField = LookUpTable.FindField("LU code")
```

or

```
LandUseField = LandUseTable.FindField("Value")
```

These lines assign the variables **`LookUpField`** and **`LandUseField`** with the contents of the corresponding columns in **`LookUpTable`** and **`LandUseTable`**, respectively.

Finally, the two tables are joined using the command:

```
aVTab.Join (aToField, aFromVTab, aFromField)
```

Let us call the table gaining additional information the "To" table, and the table that is providing that additional information the "From" table. With this naming understanding, **`aVTab`** is the name of the "To" table, **`aToField`** is the name of the joining field in the "to" table, **`aFromVTab`** is the name of the "From" table, and **`aFromField`** is the name of the joining field in the "From" table. For the specific example discussed above, this

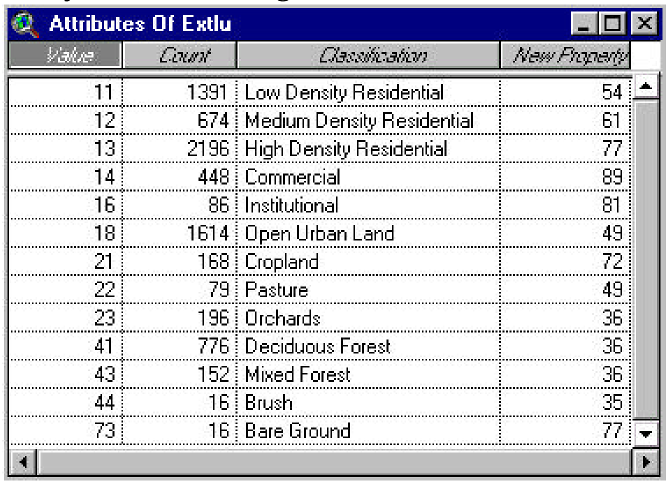
```
LandUseTable.Join(LandUseField, LookUpTable, LookUpField)
```

The directionality of this command should be evident to you. For instance, joining the tables in the opposite way:

```
LookUpTable.Join(LookUpField, LandUseTable, LandUseField)
```

would join the tables in the opposite direction, appending the contents of "**`andlookup.txt`**".

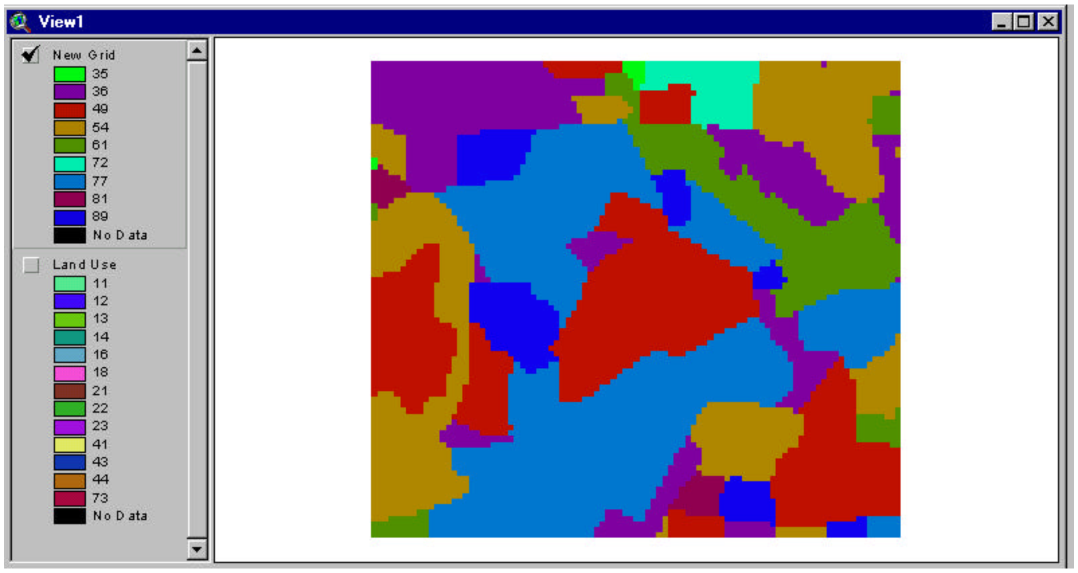
We can use joined tables in a powerful way to create a new grid based on the contents of the old grid and the joined table. For instance, say we had a field of additional information in "**andlookup.txt**" called "New Property". That field would also be appended when this table is joined to the "Attributes of Land Use" table. The new table might look like the one at left. Now for a given pixel value of say "11" which corresponds to a "Low Density Residential" classification, we also have a new property which has a value of "54". We can create a new grid that makes this assignment formally using the **Lookup** command as shown below:


Attributes Of Extlu

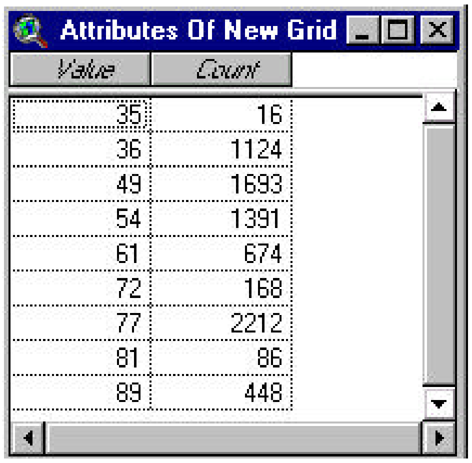
Value	Count	Classification	New Property
11	1391	Low Density Residential	54
12	674	Medium Density Residential	61
13	2196	High Density Residential	77
14	448	Commercial	89
16	86	Institutional	81
18	1614	Open Urban Land	49
21	168	Cropland	72
22	79	Pasture	49
23	196	Orchards	36
41	776	Deciduous Forest	36
43	152	Mixed Forest	36
44	16	Brush	35
73	16	Bare Ground	77

```
      New_Grid = LUgrid.Lookup("New Property")
```

This new grid is shown in the view below:

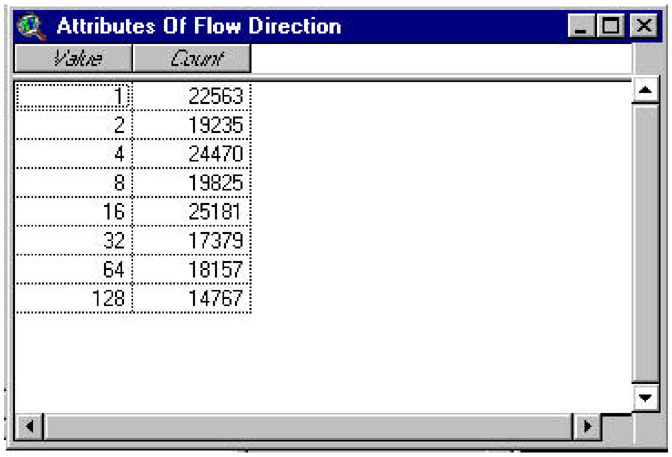


You will notice that this grid looks similar to the previous grid, but has fewer categories owing to the fact that several different land uses mapped to values of "New Property" of 36, 49, and 77. The table associated with "New Grid" is shown at left. You should verify that the "Counts" shown at left are consistent with the reassignments performed on the original "Land Use" grid.


Attributes Of New Grid

Value	Count
35	16
36	1124
49	1693
54	1391
61	674
72	168
77	2212
81	86
89	448

Accessing Entries in a Table



Attributes Of Flow Direction

Value	Count
1	22563
2	19235
4	24470
8	19825
16	25181
32	17379
64	18157
128	14767

The table at right shows generically the same information you would expect to be associated with any grid. It has an unknown number of records (8 records in this case) and two fields of predictable names: "Value" and "Count". What if you knew you wanted the count associated with "Value" equal to 4? How could you get that number? Here's how.

```
theView = av.GetActiveDoc
flowdir = theView.findtheme("Flow Direction").GetGrid
DirTab = flowdir.GetVTab
if (DirTab <> NIL) then
   DirVal = DirTab.FindField("Value")
   DirCnt = DirTab.FindField("Count")
   numrecords = DirTab.GetNumRecords
   for each i in 1..numrecords
      tempval = DirTab.ReturnValue(DirVal, i - 1)
      if (tempval = 4) then
         Dir4 = DirTab.ReturnValue(DirCnt, i - 1)
      end
   end
end
msgbox.info(Dir4.AsString, "Count for 4 is:")
```

The key lines in the above string use the **`ReturnValue`** request which has the syntax:

```
aVTab.ReturnValue (aField, aRecordNumber)
```

In the above example, **`aVTab`** is **`DirTab`** (the flow direction table), **`aField`** is the "Value" field in the first instance and the "Count" field in the second instance. The loop is iterating over all records until it finds a "Value" record equal to 4. Notice that the use of **"i – 1"** is because the initial record is record "0" while our loop is starting from "1".