JSON Exporter for Microsoft Excel

Giacomino Veltri

8/2/2010

Why Excel?

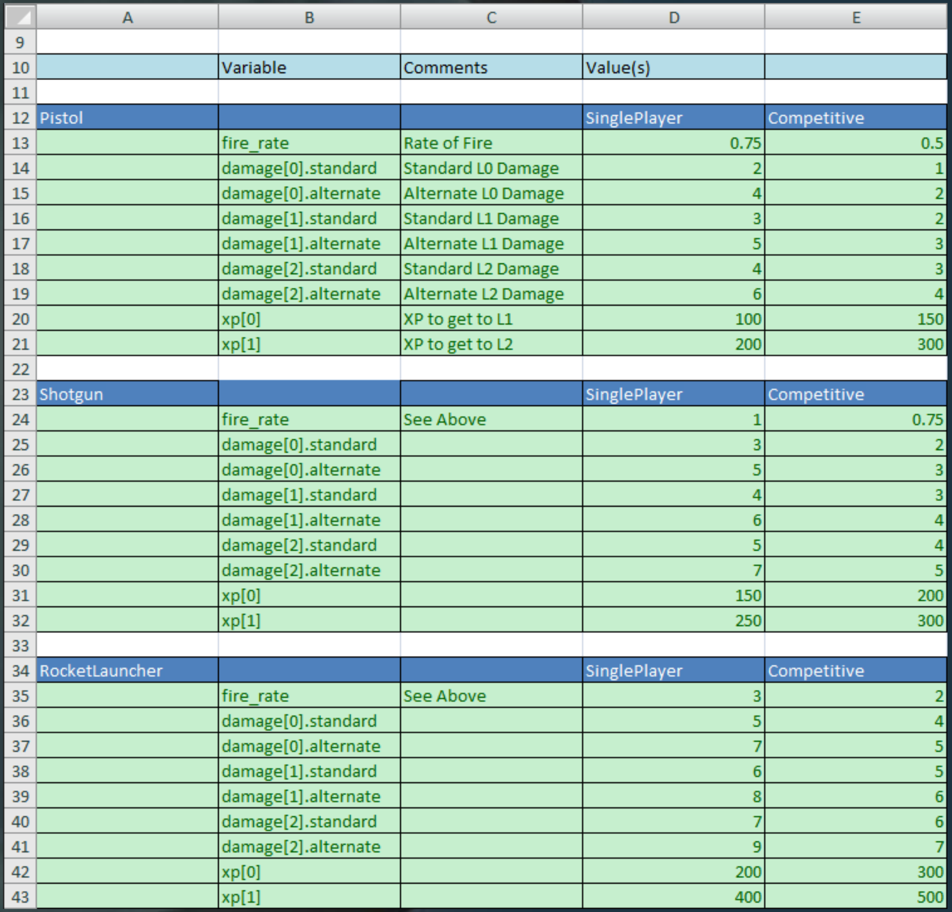
- Our designers like to edit configuration data in table form.
- People are familiar with Excel and its features (like inserting and removing rows).
- Creating a UI for configuration data would take more time than writing an exporter.
  - This is still an open possibility if we have time in the future.

Config File Excel Format

- Config data is grouped by class.
- Header rows define the class and possible value types.
- Each non-header row represents a member variable of that class and its value(s) for each value type.

Config File Excel Format

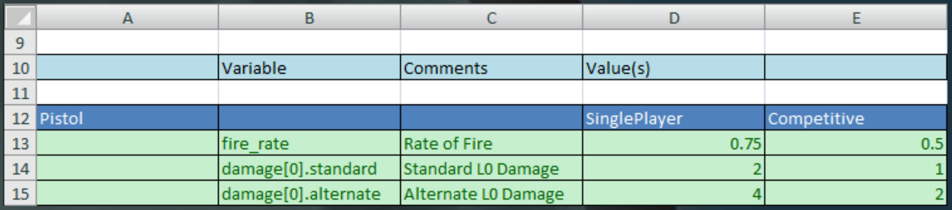
- Dark blue rows are header rows.
- Green rows are value rows.
- Column D is the SinglePlayer value.
- Column E is the Competitive value.



	A	B	C	D	E
9					
10		Variable	Comments	Value(s)	
11					
12	Pistol			SinglePlayer	Competitive
13		fire_rate	Rate of Fire	0.75	0.5
14		damage[0].standard	Standard L0 Damage	2	1
15		damage[0].alternate	Alternate L0 Damage	4	2
16		damage[1].standard	Standard L1 Damage	3	2
17		damage[1].alternate	Alternate L1 Damage	5	3
18		damage[2].standard	Standard L2 Damage	4	3
19		damage[2].alternate	Alternate L2 Damage	6	4
20		xp[0]	XP to get to L1	100	150
21		xp[1]	XP to get to L2	200	300
22					
23	Shotgun			SinglePlayer	Competitive
24		fire_rate	See Above	1	0.75
25		damage[0].standard		3	2
26		damage[0].alternate		5	3
27		damage[1].standard		4	3
28		damage[1].alternate		6	4
29		damage[2].standard		5	4
30		damage[2].alternate		7	5
31		xp[0]		150	200
32		xp[1]		250	300
33					
34	RocketLauncher			SinglePlayer	Competitive
35		fire_rate	See Above	3	2
36		damage[0].standard		5	4
37		damage[0].alternate		7	5
38		damage[1].standard		6	5
39		damage[1].alternate		8	6
40		damage[2].standard		7	6
41		damage[2].alternate		9	7
42		xp[0]		200	300
43		xp[1]		400	500

Config File Excel Format

- Row 15 generates the following lines of "C Code":



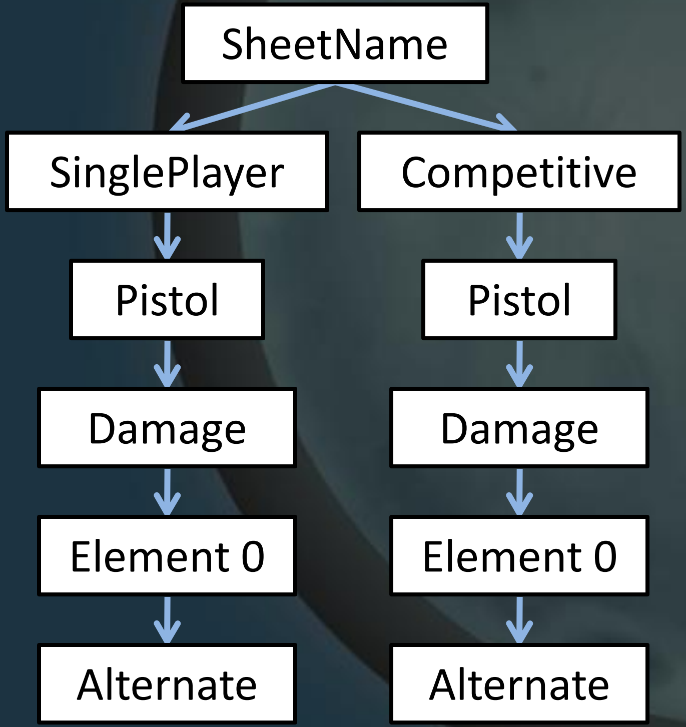
	A	B	C	D	E
9					
10		Variable	Comments	Value(s)	
11					
12	Pistol			SinglePlayer	Competitive
13		fire_rate	Rate of Fire	0.75	0.5
14		damage[0].standard	Standard L0 Damage	2	1
15		damage[0].alternate	Alternate L0 Damage	4	2

```
SheetName.SinglePlayer.Pistol.damage[0].alternate
SheetName.Competitive.Pistol.damage[0].alternate
```

- Note that the name of the sheet is present so multiple sheets can be exported.

Parsing the "C Code"

- Each member generates a new node in the tree that represents the config data hierarchy:



```
// Sample Output Json

{
  "SheetName":
  {
    "SinglePlayer":
    {
      "Pistol":
      {
        "Damage":
        [
          "Alternate": "Value"
        ]
      }
    },
    // Competitive is similar
  }
}
```

Parsing the “C Code”

- ProcessLine() will split the line into the first member variable and “everything else”.
- After its object is added to the hierarchy, ProcessArrayLine() is called to add child objects for each subscript. Missing elements will contain the value “”.
- After array objects have been handled (if necessary), ProcessLine() is called recursively on “everything else”.

Parsing the "C Code"

- "FindChildIndex(parent_index As Integer, child_name As String)" will return the child in the hierarchy (and add it if it does not exist already).

- "FindArrayChildIndex(parent_index As Integer, element_index As Integer)" will add a child to the *element_index* slot of the parent array object.

- Ideally, these two lower-level functions (and the functions they call) would be a separate Excel Add-In, since they are not specific to our config data structure in Excel.

Writing the File

- After all the rows in Excel have been processed, the hierarchy is output to a file using these functions:

```
PrintNodeArray(node_index As Integer, indent As String) As String

PrintNodeObject(node_index As Integer, indent As String) As String

PrintNode(node_index As Integer, indent As String) As String
```

  – PrintNode is just a helper that calls PrintNodeObject or PrintNodeArray.

Excel Add-In Issues

- Each node in the tree has an array of indices as children. These values are the index in the "all objects" array because I could not figure out how to make a dynamic array of pointers to nodes in Visual Basic.
  - Using an array of variants yielded "Only user-defined types defined in public object modules can be coerced to or from a variant or passed to late-bound functions".
  - Using an array of objects did not work because I could not figure out how to access my member variables on an Object, nor how to cast from an Object to my node type.

Excel Add-In Issues

- The “all objects” array is global.
  - I would have preferred to create the array locally and pass it in as a reference parameter; however, Visual Basic locks the array size of arrays passed to functions as reference parameters.
  - A fixed-size array is less desirable since guessing a maximum that is too small would require republishing the add-in.
- It seems like FindChildIndex() and FindArrayChildIndex() could be member functions on the node data type.
  - If I knew VBA better, I probably would have done it this way.
- It really should be two modules, or at least split up better.
  - Yes – the JSON hierarchy portion should be a Public Module consumed the code that parses our specific Excel spreadsheet format. But, I didn’t feel like stabbing myself in the eye that day.