Bulkuo

Bulkuo is for facilitating batch or bulk processing of Ultima Online data. It allows for extraction, creation, and merging of different UO data types from their individual native formats. These native formats are: image files (bmp, non compressed), wav files, and text files.
General usage is: **bulkuo [flags] type action path,. . . .**
Supported operating systems: *Windows*, *macOS*, and *Linux/BSD*. It will probably work on most **Unix** operating systems.
Bulkuo is a **Command Line** utility. For *Windows* users, that means one must open a command/terminal prompt (*Windows* start menu, run cmd.exe).
## Conventions
- *File naming*

Bulkuo uses the following filename convention: id-label.extension. Id is the id for the data type. Labels is optional, and appended if labels are provided on extraction (or user added). Extension is the data type for the file. **Bulkuo** can append labels automatically upon extraction if a user supplied label file is provided (–label=filepath). This file has the format of :
id = label
- *Ids*

Bulkuo uses the art image ids when extracting information (–info) data from **tiledata.mul**. It does not reset to 0 for **item** tiles. So the first **item** tile would start at id 16384 (0x4000) and go to 81,919 (0x13FFF)
## Flags
Bulkuo has several flags that format or influence the operations.
### –hex
If present, ids (in both filenames and data) that is produced by **bulkuo** will be in hex format. Flags, and other specific data will be in hex as well. This flag only influences output. Any data ingested by **bulkuo** is automatically interperted by the presence of "0x" for hex data.
### –bmp24
Bulkuo normally produces image data in UO natives format (which is 16 bit color). However, if ones tools can not process 16 bit color, this flag will result in 24 bit bmp image files being produced. This flag has no influence on input. The bmp file is automically interpreted and converted as needed.
Note: BMPS can not be compressed.
### –overwrite
Bulkuo normally will not overwrite data when producing files. The use of the –overwrite flag will allow files to be overwritten.
### –version
Results in **bulkuo** printing its version and returning.
### –help
Results in **bulkuo** printing the command line usage.
–id
Allows the user to restrict **bulkuo** operations on a constrained set of ids. The format is:

"id=f:filepath" or "id=l:list"
A list is a series of id or ranges separated by commas. A range is designed by: startid-endid, and is inclusive. As an example, the list: 3,6,10-14,20 would include all ids of: 3,6,10,11,12,13,14,20. Ids can be hex(appropriately prefixed) or decimal. The format of a file for this option is lines of id lists.
### –label=filepath
Label is an directing bulkuo to utilize a label file for extraction (only time it is utilized). Usage would be **–label=filepath**, where the file is a series of lines that are: **id = label** The names will be appended to the file name for those ids.
### –category=filepath
Category allows one to direct the location of data when extracted. It is a list of subdirectory locations that will be relative to the specified directory for extraction. The file contains a series of lines of the format **path = idlist** where id list is a list is a series of id or ranges separated by commas. Ids can be repeated, but the last entry in the file for an id is used. A category file for art data might be:
land = 0-0x3fff
item = 0x4000- 0x13fff
This would have all land artwork be in a subdirectory land, and all item artwork be in a subdirectory item.
## Types
Bulkuo can process various data types of uo data.
### –animation
This will extract animations from an idx/mul file combination. It takes two directories for output. The first directory is for entries that have actual data, so bmps and csv data will be placed there. The second directory is a list of files with the id that indicates for that entry, no data is available, but it is an valid id, that can be substituted by information in the client text files. These entires are used on creation/merging to endure the created idx/mul does not lose this information (and one can change it). The csv files created, indicates for each animation id, how many frames exist, an for each frame, if it has an image, and the centerX,centerY for the offset when drawing that image. Frames are labeled as: animationID.frameID[-label].bmp , where the [-label] is only present if the –label option is used.
### –art
This will extract or ingest bmp files for art images. IDs from 0 to 0x3FFF must be 44 x 44 pixels.
### –gump
This will extract or ingest bmp files for a gump.
### –hue =[width,height]
This will extract a bmp file for the color range, with a secondary text file for the text string. The optional width,height on the –hue is for extraction only. It specified the cell size to use for each color (so the total bmp will be (widthx32) x height pixels. For creation/merging, the bmp can be any size. The width of the bmp is divided by 32 to make a cellsize. The color is then taken from pixel(((cellsize x hueindex) + cellsize/2)),height/2).
### –info

This will extract or ingest the the information data about a tile normally contained in **tiledata.mul**. It will be a single csv file.
### –light
This will extract or ingest bmp files for a light entry. The last channel (blue) is used for the value , as this is a grayscale.
### –multi
This will extract or ingest commas separated file (csv) for a multi entry.
### –sound
This will extract or ingest wav files. There is a seconary file(txt), which is the an internal text string for an entry. If that secondary file is not present, an empty text string will be used for creation/merging.
### –texture
This will extract or ingest bmp files. They must be 64 x 64 or 128 x 128 pixels in size.
## Actions
### –create
Bulkuo will scan the directory and all subdirectoris for input.
This is of the form:
directory uoptocreate
or
directory idxtocreate multocreate
For info data it is:
inputcsvfile mulfiletocreate
For hue it is :
directory mulfiletocreate

Remember, the ids used will be restriced by any –id quailifers.
### –exist
Exist action will print out all ids that exist in thw data (as constrained by –id) to standard output (This can be piped to a file with: » **filename**).
### –extract
This is of the form:
uopsource outputdirectory
or
idxsource mulsource outputdirectory
For info data it is:
mulfilesource outputcsvfile
For hue it is :
mulfilesource outputdirectory
### –merge
Bulkuo will scan the directory and all subdirectoris for input.
This is of the form:
directory uopsource
or
directory idxsource mulsource
For info data it is:

inputcsvfile mulfilesource
For hue it is :
directory mulfilesource
It will will create new output files with the extension **.bulkuo** for the type of source data it had(uop,idx/mul). It will create new uop/mul files, with the contents of the source uop/mul and the directory data (constrained by –ids) either overwritting or adding to.
### –name
One of **bulkuo's** actions is the creation of name files (–name) which will generate name files
utilizing the UO data for sound,hues,info(info names can be used for art data). Multi names can be generated as well, but are hard coded into the program and not read from the data.

Bulding

Visual Studio

There is a *Visual Studio* solution. Open **bulkuo.sln** in Visual Studio.
### XCode
There is a *Xcode* project. Open **bulkuo.xcodeproj** in XCode
### Cmake
- Create a build directory in the top level **bulkuo** directory. (i.e. `mkdir build`)
- Change to that directory. (i.e. `cd build`)
- Generate the makefiles. `cmake .. -DCMAKE_BUILD_TYPE=Release`.
- For *Window* and *macOS* use the `-G"NMAKE Makefiles"` and `-G"Unix MakeFiles"` respectively. - Build it. `cmake --build .`
# Examples
## Creating labels automatically
- Prerequisite
- Directory with three subdirectories: category, label, uo
- Tiledata.mul and artLegacyMUL.uop in the uo subdirectory
- **bulkuo** executable is in the main directory.
- Goal
- Create labels and categories so artwork is binned in either an item or land subdirectory with a name appended to the id filename.
- Enter: `./bulkuo --info --name uo/tiledata.mul label/tilename.txt`
- This extracts names for each tile id that has a name in the mul file and puts it into "label" format in *label/tilename.txt*
- Edit the names, or add new entries if desired in *label/tilename.txt*
- Create a text file, *sort.txt* in the *category* subdirectory with the following two lines:
- **land = 0x0-0x3fff**
- **item = 0x4000-0x10000**
- Enter: `./bulkuo --extract --art --label=label/tilename.txt --category=category/sort.txt uo/artLegacyMUL.uop artwork`

- This will create artwork in the newly created *artwork* subdirectory. The images will be sorted into land/terrain subdirectories, and have the name append to the tileid.