uocte / uocte / wiki

Because no specification of this file format was available for the development of uocte, the file format was reverse engineered for interoperability. The information on this page therefore is incomplete and may be incorrect. It only serves to document which parts of the data are interpreted by uocte and which assumptions it makes concerning interpretation.

Heidelberg data is stored in a single binary, little endian file with extension `e2e` or `E2E`. It contains a header, a directory that is split in chunks of entries in a single-linked list, and data chunks. The high-level structure is this:

```
// header
char[12]            magic1
u32                 version ("0x64")
u16[9]              9x "0xffff"
u16                 "0x0"

// main directory
char[12]            magic2
u32                 version ("0x64")
u16[9]              9x "0xffff"
u16                 "0x0"
u32                 num_entries
u32                 current
u32                 "0x0"
u32                 ?

// traverse list of directory chunks
do
  seek(current)
  char[12]            magic3
  u32                 version ("0x64")
  u16[9]              9x "0xffff"
  u16                 "0x0"
  u32                 num_entries
  u32                 ?
  u32                 prev
  u32                 ?

  for i in num_entries..1
    u32               pos
    u32               start
    u32               size
    u32               "0x0"
    u32               patient_id
    u32               study_id
    u32               series_id
    u32               slice_id
    u16               ?
    u16               "0x0"
    u32               type
    u32               ?
    if start > pos
      push (start,size) to stack S

  current = prev
while current != 0
```

This gives a stack of data chunks to process. uocte uses this first pass through the file to determine the number of slices as the maximum of `slice_id/2` for each combination of `study_id` and `slice_id`. uocte assumes only one `patient_id` to occur in a file.

In a second pass, uocte seeks to each the position from the stack `S` and reads one data chunk. Each chunk has a header with the following structure:

```
u8[12]                magic4
u32                   ?
u32                   ?
u32                   pos
u32                   size
u32                   "0x0"
u32                   patient_id
u32                   study_id
u32                   series_id
u32                   slice_id
u16                   ind
u16                   ?
u32                   type
u32                   ?
```

`slice_id` is `0xffffffff` for data not specific to a slice. There are numerous data chunks of differnt type, most of which I could not determine their semantics. Only the tags processed by uocte are listed here in detail. All strings are treated as ISO8859-1 encoded.

```
// type == 0x00000009 - patient info
u8[31]                given name
u8[66]                surname
u32                   birthdate
u8                    sex ("M" or "F")
```

The date of birth as Julian date is: `JD = (birthdate/64) - 14558805`.

```
// type == 0x0000000B - laterality
u8[14]                ?
u8                    laterality ('L' or 'R')
u8[?]                 ?

// type == 0x40000000 - image data
u32                   size
u32                   type ("0x02010201" for fundus, "0x02200201" for tomogram)
u32                   ?
u32                   width
u32                   height
if ind == 0
  u8[height][width]   raw fundus image
else
  uf16[height][width] raw tomogram slice image
```

`uf16` is a floating point type with no sign, 6-bit exponent, and 10-bit mantissa. uocte assumes that the tomogram spans [1/6,5/6]x[1/4,3/4] of the fundus image and has x- and z-dimensions of 6 and 4.5 millimeters, respectively, an a y-resolution of 3.9µm. These values were inferred from exported XML files. The images are not yet correctly registered. The official viewer shears the images in y-direction. The amount of shear has not yet been found in the data.

```
// type == 0x00002723 - contour data
u32                   ?
u32                   id
u32                   ?
u32                   width
f32[width]            contour depth in tomogram pixels for current slice
```

Note that the z-direction for tomogram and contours needs to be flipped.

Not yet found in the data is the aquisition date.

Updated 2015-06-04