Joeri van Grimbergen (1244825) | Nursize Bilen (1260235) | Rick van Gorp (1328417)

Zuyd Hogeschool | Heerlen | GRoep 4

MICT1

eXercise week 8

Inhoud

[1. Voorbereiding van analyse 2](#_Toc447826473)

[Doel van de opdracht 2](#_Toc447826474)

[Tools 2](#_Toc447826475)

[2. Analyse van *ZIF-formaat* 3](#_Toc447826476)

[2.1 Analyseren van ZIF-header 4](#_Toc447826477)

[2.2 Analyseren van de COLR-chunk 5](#_Toc447826478)

[2.3 Analyseren van de DATA-chunk 8](#_Toc447826479)

[2.4 Hypotheses testen met Python 9](#_Toc447826480)

[2.5 ZIF-formaat 11](#_Toc447826481)

[3. Python scripts 13](#_Toc447826482)

# 1. Voorbereiding van analyse

Er is onderzoek uitgevoerd naar de in week 8 geleverde bestanden *“source.png”, “source.zif”* en *“modified.zif”*. Om deze bestanden te analyseren is gebruik gemaakt van 010 Editor. Dit is een hex-editor die in staat is om templates te gebruiken die de gebruiker programmeert. Op basis van structures kan door het binaire bestand genavigeerd worden. Naast 010 Editor is gebruik gemaakt van een Python script om de inhoud van het *ZIF*-bestand leesbaar te maken voor RGB-image viewers.

## Doel van de opdracht

Om de analyse uit te kunnen voeren moet het doel van de opdracht bekend zijn. Het doel van de opdracht is om te ontdekken hoe het nieuwe bestandsformaat “Zuyd Image Format” is opgebouwd en om uit een dergelijke afbeelding het ingebakken bericht te exporteren.

## Tools

De tools die zijn gebruikt voor dit onderzoek zijn onderstaand weergegeven:

* 010 Editor: <http://sweetscape.com/download/010editor/>;
* Irfanview: <http://www.irfanview.com/main_download_engl.htm>;
* StructAnalysis.py: Te vinden als bijlage in map week 8.

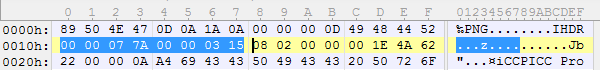
# 2. Analyse van *ZIF-formaat*

Er zijn enkele zaken bekend van het Zuyd Image Format (ZIF), namelijk:

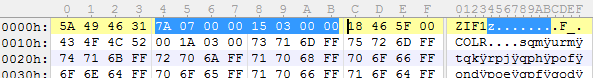
* Het is een afbeelding;
* De afbeelding is gelijk aan de afbeelding in *source.png*.

Omdat de afbeelding gelijk is aan de afbeelding in *source.png* wordt in 010 editor gezocht naar de hexadecimale waarden voor de grootte (*hoogte x breedte)* van de afbeelding in *source.png*. Vervolgens wordt gecontroleerd in bestand *image.zif* of dit bestand dezelfde waarden bevat en op welke locatie. Het is noodzaak dat de grootte van de afbeelding in pixels is gedefinieerd, omdat deze gegevens gebruikt worden bij het invullen van de pixels met kleuren. Bij het inlezen van RAW image data kan worden gespecificeerd wat het aantal bytes per pixel (BPP) is. Als de afbeelding bijvoorbeeld een 24 bit RGB afbeelding is, zijn er 3 bytes per pixel gedefinieerd. Als de grootte van de afbeelding in het ZIF-bestand verschilt van de originele afbeelding, moet bekend worden wat de grootte is van de afbeelding in het ZIF-bestand, alvorens deze afbeelding zoals oorspronkelijk (maar kleiner) kan worden weergegeven.

Met behulp van de specificaties van PNG wordt bepaald waar de grootte van de afbeelding *source.png* is gedefinieerd. Deze waarde wordt gebruikt om in *source.zif* te zoeken naar overeenkomende waarden die de grootte zouden kunnen definiëren.



Op hexadecimale offset 0x10 in bestand *source.png* is een *unsigned integer (32-bits)* gedefinieerd die de breedte van de afbeelding omschrijft. Een 32-bits *unsigned integer* is 4 bytes lang. Op hexadecimale offset 0x14 in bestand *source.png* is een *unsigned integer (32-bits)* gedefinieerd die de hoogte van de afbeelding omschrijft. Als deze lengtes worden omgezet naar integers is de breedte van de afbeelding gelijk aan 1914 pixels en de hoogte van de afbeelding gelijk aan 789 pixels. De hexadecimale waarden hiervan zijn respectievelijk gelijk aan 00 00 07 7A en 00 00 03 15. Zoeken naar deze waarden in een ZIF-bestand levert geen resultaten op. Na het converteren van de notatie van little endian naar big endian (00 00 07 7A wordt 7A 07 00 00 en 00 00 03 15 wordt 15 03 00 00) werden de volgende resultaten weergegeven in het ZIF-bestand:

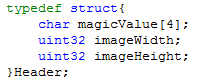


Deze waardes volgen direct de *file signature* (ZIF1). Het is zeer waarschijnlijk dat deze bytes representatief zijn voor de hoogte en breedte van de afbeelding. Big-endian waarde 7A 07 00 00 is gevonden in bestand *source.zif* op hexadecimale offset 0x4. Big-endian waarde 15 03 00 00 is gevonden in bestand *source.zif* op hexadecimale offset 0x8. Beide waardes zijn in dit bestand ook *Unsigned Integers (32-bits)*.

## 2.1 Analyseren van ZIF-header

Nu bekend is waar de header van het ZIF-bestand zich bevindt met algemene data, wordt een template in 010 editor opgesteld voor het gehele ZIF-bestand. Hierin wordt de structuur van de data gedefinieerd, waardoor later in 010 editor op basis van deze template door het bestand *source.zif* en *modified.zif* genavigeerd kan worden.

Enkele waardes, zoals de file signature en de grootte van de afbeelding zijn op dit moment bekend en staan in het begin van de afbeelding. Dit resulteert in *struct*:



Een struct is een bundeling van definities van data types. Onderstaand worden de waardes toegelicht:

* *magicValue*: Dit is data van het type *char* met een lengte van 4 bytes;
* *imageWidth*: Dit is data van het type *unsigned integer 32*. Dit is een getal met een lengte van 4 bytes;
* *imageHeight*: heeft dezelfde definitie van data als *imageWidth*.



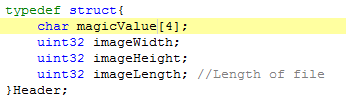
In bovenstaande afbeelding is de ZIF file header weergegeven. Deze wordt vervolgd door 8 bytes met data betreffende de grootte van de afbeelding. Er zijn nog 4 bytes over tot de mogelijk nieuwe *chunk* wordt gedefinieerd. Chunk COLR lijkt andere data te bevatten en een kleine interne header die mogelijk gegevens bevat over deze eigen chunk. De hypothese die wordt gesteld is dat de laatste 4 bytes die zijn overgebleven in de ZIF *file signature* de grootte van het hele bestand representeren.

Om dit vast te stellen wordt de decimale waarde bepaald van de hexadecimale waardes op hexadecimale offset 0x12 tot offset 0x16. *Dec(LITTLE-ENDIAN(18465F00))* = 6.243.884 bytes.

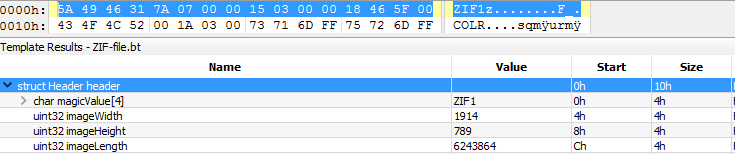
Dit komt overeen met de waarde in de eigenschappen van *source.zif*:



Dit resulteert in de volgende *ZIF-header* structuur:

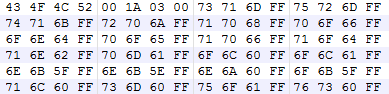


De header kan worden geparsed in 010 editor door de line: *Header header;* toe te voegen. De waardes die 010 Editor in het bestand vindt zijn de volgende:



## 2.2 Analyseren van de COLR-chunk

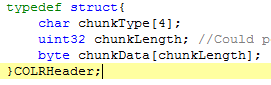
De COLR chunk bevat andere data dan de ZIF-header, waardoor het lijkt dat op dit punt een nieuwe chunk begint.



Bovenstaand zijn de hexadecimale waardes van een stukje van de COLR chunk weergegeven. Uit deze weergave is op te maken dat de eerste twee sets van 4 bytes chunk-gerelateerde gegevens bevatten en dat op byte 8 de data uit deze chunk is gedefinieerd. De hypothese voor de structuur van deze chunk is als volgt:

* *4 Bytes: Chunk Data Type (COLR);*
* *4 Bytes: Chunk Data Length;*
* *[Chunk Data Length] Bytes: Chunk Data.*

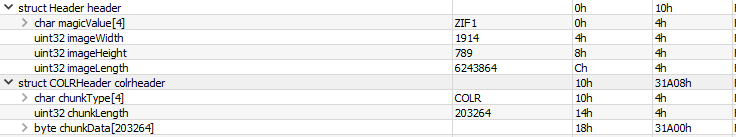
Deze structuur wordt gereproduceerd in 010 Editor en vervolgens wordt gecontroleerd of de hypothese klopt.



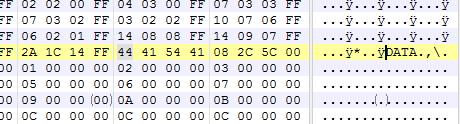
Onderstaand worden de waardes toegelicht:

* *chunkType*: Dit is data van het type char met een lengte van 4 bytes;
* *chunkLength*: Dit is data van het type *unsigned integer* (32-bits) dat representatief is voor de lengte van de chunkData;
* *chunkData:* Dit is data van het type byte, waarvan de lengte afhankelijk is van de waarde van *uint32 chunkLength*.

Om te bepalen of de hypothese van de betreffende structuur klopt, moet bepaald worden of het einde van de in 010 Editor geparsete sectie te vinden is bij het begin van een nieuwe chunk data type. Het template geparsed in 010 Editor ziet er als volgt uit:



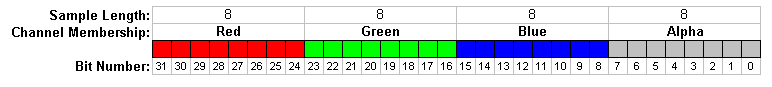
De chunkLength geeft aan dat de lengte van *chunkData* gelijk is aan 203264. Om dit te bevestigen wordt het einde van *chunkData* gecontroleerd. Het einde van *chunkData* is te vinden op hexadecimale offset 0x31A18. Op deze offset lijkt een nieuwe *chunkType* weergegeven, namelijk: DATA.



De hypothese die is gesteld voor het formaat van deze COLR chunk is bevestigd. De data die binnen *chunkData* is weergegeven in de COLR-chunk, heeft een repetitief patroon:

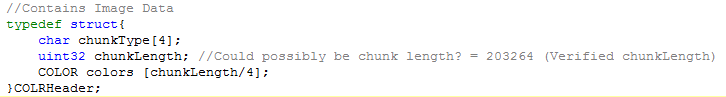


Er worden steeds 4 bytes weergegeven, waarvan de laatste byte gelijk is aan FF. Aangezien bekend is dat het ZIF-formaat een afbeelding betreft, wordt aan de hand van de RGB-specificatie bepaald of er sprake is van kleurendata.



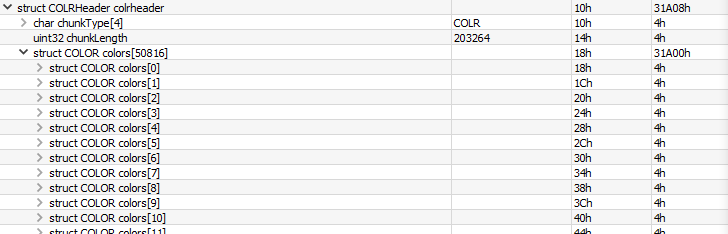
Bovenstaand is een afbeelding weergegeven afkomstig van <https://en.wikipedia.org/wiki/RGBA_color_space>. Deze afbeelding bevat informatie over het hexadecimaal formaat van RGB-data. In het totaal is er sprake van 32-bits. Dit is gelijk aan de 4 bytes die zijn te vinden in de COLR-chunk. Elke byte is representatief voor een onderdeel van het RGBA-formaat. Het ZIF-formaat staat echter in Little Endian formaat, waardoor de data wordt gelezen als FF 14 1C 2A. Dit houdt in dat de Alpha waarde vooraan staat. Deze is gelijk aan FF (dus er is geen opacity). Het is op dit moment echter onbekend wat de volgorde is van de rest van de kleuren, maar er is wel bekend dat de volgorde van het formaat met A van RGBA begint bij het inlezen.

De structuur van de COLR-chunk wordt naar aanleiding van deze bevinding gedetailleerder en krijgt een definitie per kleur in hexadecimaal toegevoegd. Dit resulteert in het volgende template onderdeel voor bestand *source.zif*.





Er worden voor de COLR chunk twee structures gedefinieerd, waarvan structure COLOR wordt gebruikt als definitie van één kleur in de chunk. Deze kleur wordt gespecificeerd als *unsigned integer 32* en heeft een lengte van 4 bytes. Daarnaast wordt deze in hexadecimaal formaat weergegeven. Dit template geeft het volgende resultaat in 010 Editor:



Het is onbekend voor welke doeleinden de data in deze COLR chunk wordt gebruikt. Er zijn twee hypotheses die bevestigd kunnen worden:

* De data in de COLR chunk is een color table. Dit is een kleurentabel dat een verzameling is van alle unieke kleuren binnen de afbeelding. Deze verzameling wordt door latere data gebruikt om de pixels in te kleuren.
* De data in de COLR chunk is de daadwerkelijke afbeelding met alle pixels reeds gedefinieerd.

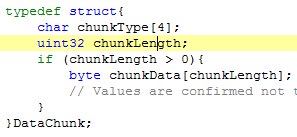
Na de COLR-chunk volgt de DATA-chunk, die groter is dan de COLR-chunk. Het is dus onwaarschijnlijk dat de tweede hypothese correct is. Met deze reden wordt de eerste hypothese als eerste getest.

## 2.3 Analyseren van de DATA-chunk

Dezelfde hypothese wordt gebruikt als bij de COLR-chunk. Het formaat van de DATA-chunk ziet er als volgt uit:

* *4 Bytes: Chunk Data Type (COLR);*
* *4 Bytes: Chunk Data Length;*
* *[Chunk Data Length] Bytes: Chunk Data.*

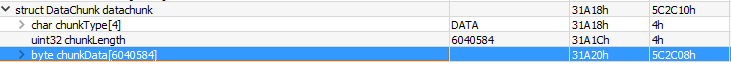
Deze structuur wordt gereproduceerd in 010 Editor en vervolgens wordt gecontroleerd of de hypothese klopt.



Onderstaand worden de waardes toegelicht:

* *chunkType*: Dit is data van het type char met een lengte van 4 bytes;
* *chunkLength*: Dit is data van het type *unsigned integer* (32-bits) dat representatief is voor de lengte van de chunkData;
* De onderstaande waarde komt alleen voor als de waarde in *chunkLength* groter is dan 0;
* *chunkData:* Dit is data van het type byte, waarvan de lengte afhankelijk is van de waarde van *uint32 chunkLength*.

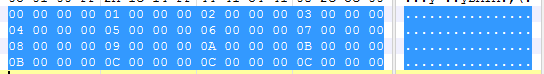
Om te bepalen of de hypothese van de betreffende structuur klopt, moet bepaald worden of het einde van de in 010 Editor geparsete sectie te vinden is bij het begin van een nieuwe chunk data type. Het template geparsed in 010 Editor ziet er als volgt uit:

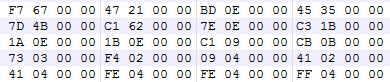


De chunkLength geeft aan dat de lengte van *chunkData* gelijk is aan 6040584. Om dit te bevestigen wordt het einde van *chunkData* gecontroleerd. Het einde van *chunkData* is te vinden op hexadecimale offset 0x5F4628. Vanaf deze offset bevat het bestand *sources.zif* geen data meer.

De data in de DATA-chunk begint met een opeenvolgende reeks aan aan hexadecimale waarden, waarbij steeds het getal 1 opgeteld wordt:

(Data vanaf offset: 0x31A20)





In bovenstaande afbeelding is een deel van de DATA-chunk op een latere offset te zien: 0x431DF0. De reeks als aan het begin van de DATA-chunk te zien is hier niet langer zichtbaar. Naar aanleiding van deze informatie zijn de volgende hypotheses opgesteld:

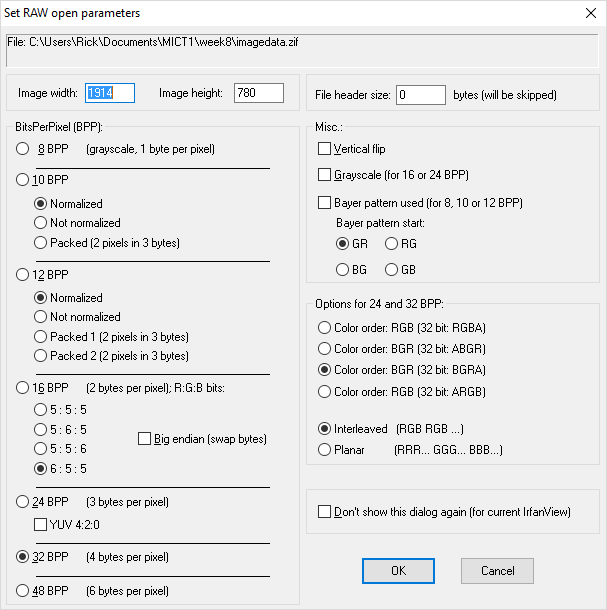
* Per 4 bytes wordt, als deze zijn geconverteerd naar een decimaal getal, een index aangegeven binnen de COLR-chunk. Dit houdt in dat een getal uit de DATA-chunk representatief is voor de index van een bepaalde kleur in de COLR-chunk;
* Per 4 bytes wordt, als deze zijn geconverteerd naar een decimaal getal, een offset aangegeven binnen de COLR-chunk. Dit houdt in dat een getal uit de DATA-chunk representatief is voor de offset van data in de COLR-chunk.

## 2.4 Hypotheses testen met Python

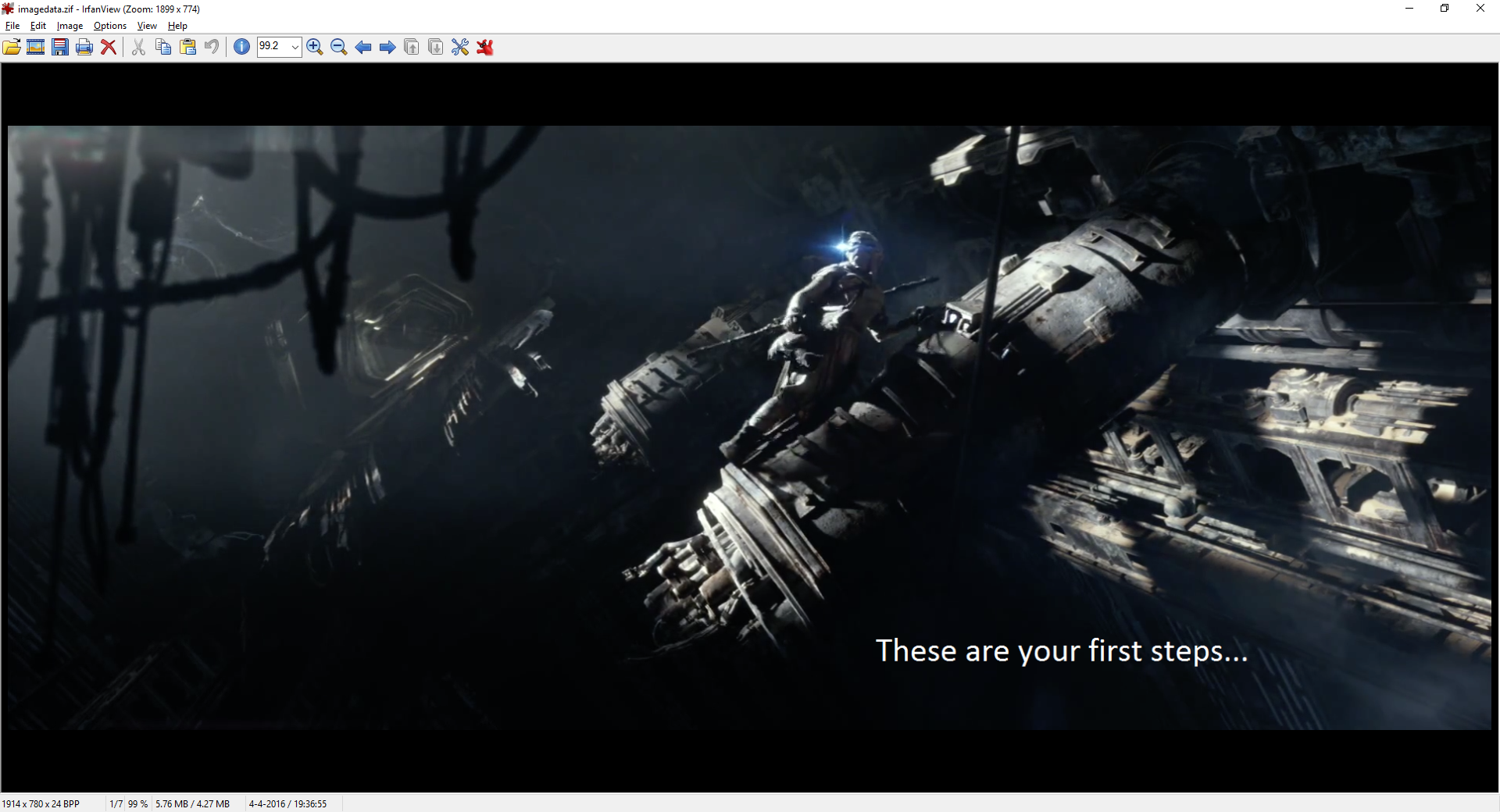
Omdat de DATA-chunk veel data bevat die vervangen moet worden met een kleur uit de kleurentabel COLR, wordt dit geautomatiseerd uitgevoerd. Hiervoor is het Python-script *StructAnalysis.py* gemaakt. Deze wordt nader toegelicht in hoofdstuk 3: Python scripts. Dit script is in staat om per set van 4 bytes in de DATA-chunk een *unsigned integer (32-bits)* te bepalen. Door deze waarde te nemen als index binnen de COLR-tabel kan worden bepaald welke kleur op die betreffende locatie moet staan. Het resultaat hiervan wordt gereconstrueerd in het bestand *imagedata.zif* die te vinden is als bijlage in de map week 8.

Het bestand *imagedata.zif* is het directe resultaat van *modified.zif*. Dit is een gemodificeerde afbeelding, maar heeft dezelfde structuur als *source.zif*. *imagedata.zif* is een bestand die alleen imagedata bevat op basis van het RGBA-formaat (mogelijk is ook ARGB en alle andere varianten). Met deze kleurendata worden de pixels ingekleurd.

Het bestand *imagedata.zif* wordt als RAW-image bestand ingelezen in Irfanview. De instellingen die worden gebruikt zijn reeds bekende instellingen zoals de hoogte en breedte van de afbeelding. Daarnaast wordt gekozen voor 4 Bytes Per Pixel (32 BPP), omdat de kleurendata in het bestand *imagedata.zif* in het RGBA formaat staat. Het RGBA-formaat gebruikt een notatie in 32-bits. De Color Order is gekozen op basis van interpretatie van de weergegeven afbeelding en de bron afbeelding *source.png*. Onderstaande instellingen zijn gebruikt om het bestand te laden in IrfanView: (*zie volgende pagina)*.



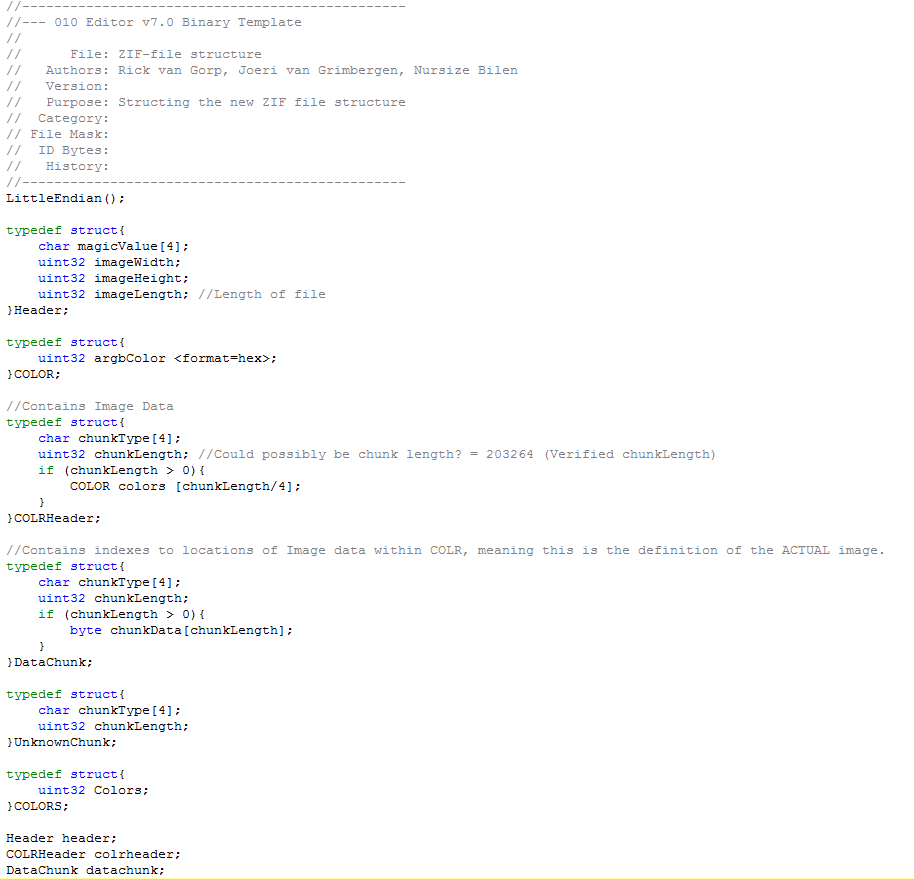
De afbeelding die wordt weergegeven is gelijk aan *source.png* met tekst toegevoegd:



Dit is het bericht dat beschreven werd in het doel van het onderzoek: “These are your first steps”. Hierbij wordt de hypothese “Per 4 bytes wordt, als deze zijn geconverteerd naar een decimaal getal, een index aangegeven binnen de COLR-chunk. Dit houdt in dat een getal uit de DATA-chunk representatief is voor de index van een bepaalde kleur in de COLR-chunk” bevestigd.

## 2.5 ZIF-formaat

In dit gedeelte wordt het ZIF-formaat in geheel beschreven. Het template dat is gebruikt om het ZIF-formaat te specificeren in 010 Editor is onderstaand weergegeven:



Deze structs zijn eerder reeds besproken. De structs zijn gevisualiseerd op de volgende pagina:

ZIF File format:

ZIF-header (16 Bytes)

COLR-chunk (8 + *n* Bytes)

DATA-chunk (8 + *n* Bytes)

|  |  |
| --- | --- |
| File Signature: ZIF1  4 Bytes (char[4]) | ImageWidth  4 Bytes (uint32) |
| ImageHeight  4 Bytes (uint32) | ImageLength  4 Bytes (uint32) |
| ChunkType  4 Bytes (char[4]) | ChunkLength  4 Bytes (uint32) |
| *n* \* Color  4 Bytes per Color (uint32)  *n =* (ChunkLength/4) | |
| ChunkType  4 Bytes (char[4]) | ChunkLength  4 Bytes (uint32) |
| *n \** ChunkData  *n Bytes* in ChunkData (bytes[ChunkLength])  *n = ChunkLength* | |

Uit bovenstaande informatie is op te maken dat het algemene formaat van een chunk binnen het ZIF file format gelijk is aan:

|  |  |
| --- | --- |
| ChunkType  4 Bytes (char[4]) | ChunkLength  4 Bytes (uint32) |
| [any type] ChunkData  Gebaseerd op ChunkLength | |

Daarnaast zijn onderstaande definities bekend:

* Big Endian notatie, maar moet uitgelezen worden als Little Endian;
* COLR-chunk is de chunk die een kleurentabel bevat. Deze kleuren zijn op basis van indexen te raadplegen;
* DATA-chunk is de chunk waarin wordt gedefinieerd wat de kleur is per pixel in de afbeelding. Dit wordt gedefinieerd door een index aan te geven van de kleur uit de COLR-chunk.

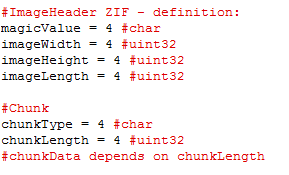
# 3. Python scripts

De data in de DATA-chunk is geconverteerd naar kleurendata met behulp van het Python script *StructAnalysis.py*.

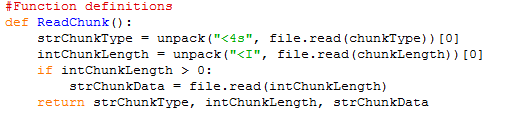


Allereerst import het script enkele modules die nodig zijn om het script uit te voeren:

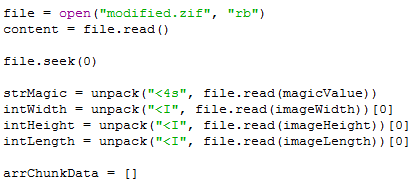
* Module re: Deze wordt gebruikt om op basis van reguliere expressies overeenkomsten te vinden in het geladen bestand;
* Module binascii: Deze wordt gebruikt om de hexadecimale data uit het bestand *allheaders* te converteren naar binaire data, leesbaar voor Python;
* Het script gebruikt de bibliotheek struct om binaire data van bestanden te converteren naar, bijvoorbeeld, *unsigned integers* en *unsigned chars*. *Signed* is ook mogelijk. De bibliotheek is gebaseerd op *struct* dat in programmeertaal C wordt toegepast.



In bovenstaande afbeelding zijn de algemene definities weergegeven van reeds bekende datalengtes in bytes. Daarnaast staat het type data in een comment (achter #) weergegeven.



De functie *ReadChunk* leest de chunks uit het bestand *sources.zif* of *modified.zif* op basis van de algemene chunk structuur die wordt toegepast binnen het ZIF-formaat. De uitgelezen chunktype, chunklength en data worden als return teruggestuurd.



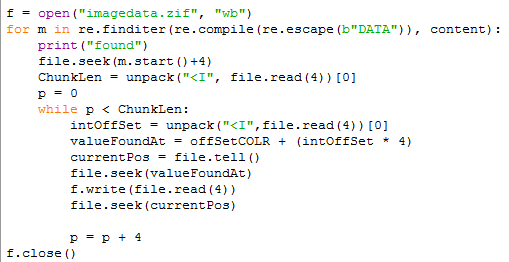
Het script exporteert de ZIF-header van het bestand volgens het bekende ZIF-header formaat. Deze worden in variabelen geplaatst. *<* geeft aan dat de waardes ingelezen worden als Little Endian.



In dit gedeelte worden alle chunks uitgelezen tot het einde van het bestand is bereikt.



Het script zoekt in het bestand naar de byte string *COLR*. Vervolgens wordt de locatie hiervan + 8 bytes opgeslagen in *offSetCOLR*. In *offSetCOLR* wordt de positie van het begin van de chunkData aangegeven.



Het doelbestand *imagedata.zif* wordt in binaire schrijfmodus geopend. Vervolgens wordt in het geopende bestand *sources.zif* of *modified.zif* gezocht naar de binaire string *DATA*. De cursor binnen het bestand wordt op deze locatie geplaatst + 4 bytes. Vervolgens wordt de chunklengte uitgelezen en worden de volgende acties uitgevoerd per 4 bytes tot het einde van de chunk is bereikt:

* Lezen waarde in bestand *modified.zif* of *sources.zif* (4 bytes) en converteren naar integer;
* Variabele *valueFoundAt* gelijkstellen aan de positie van de kleur in *COLR* aan de hand van de berekende index + bekende offSet van *COLR*. De cursor wordt verplaatst naar deze positie.
* De gelezen waarde uit de *COLR-chunk* wordt geschreven in het bestand *imagedata.zif*.

<https://en.wikipedia.org/wiki/RGBA_color_space>

<http://www.scantips.com/basics1d.html>