# MICT1

## EXERCISE WEEK 3

Joeri van Grimbergen (1244825) | Nursize Bilen (1260235) | Rick van Gorp (1328417)
ZUYD HOGESCHOOL | HEERLEN | GROEP 4

# 1. What we found and where we found it

Voor dit onderzoek is een script gebouwd in Python die in staat is om te bepalen of er sprake is van foutieve blocks en in staat is om toegevoegde random data blocks, zoals de nullen, te verwijderen uit het bestand. Alle handelingen die onderstaand zijn beschreven dienen in volgorde te worden uitgevoerd om te voorkomen dat de waardes van de grenzen niet langer kloppen. Het doel van dit onderzoek is om zoveel mogelijk data terug te halen uit het GIF-bestand.

Het resultaat bestaat uit het GIF-bestand *e_r.gif*, waarbij 11 frames hersteld zijn en het GIF-bestand *f_z.gif*, waarbij 7 frames hersteld zijn. Aan de hand van de gedefinieerde "magic value" GIF89a die als prefix is weergegeven in beide bestanden is vastgesteld dat het om twee GIF-bestanden gaat.

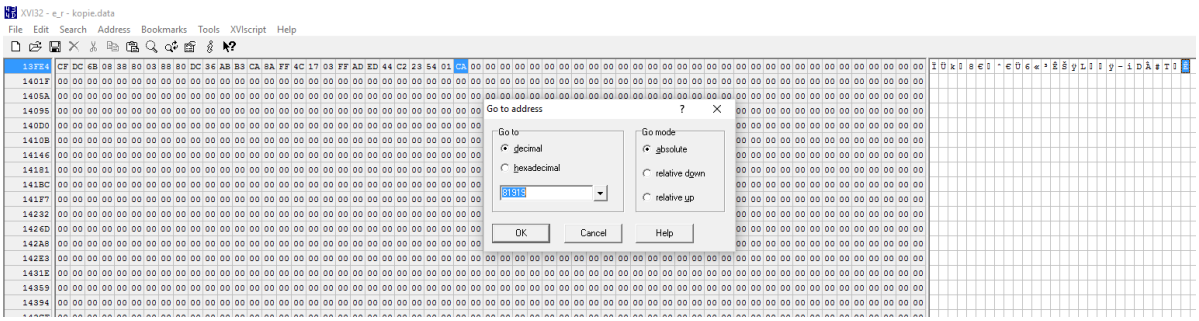
Onderstaande resultaten hebben betrekking op het bestand *e_r.data*.

```
Verified Values:
[['APPLIC', 781, 800], ['GRAPHIC', 800, 808], ['IMG', 808, 67132], ['GRAPHIC', 67132,
67140], ['IMG', 67140, 82000], ['GRAPHIC', 147254, 147262], ['IMG', 147262, 163914], [
'GRAPHIC', 390449, 390457], ['IMG', 390457, 409669], ['GRAPHIC', 466416, 466424], ['IM
G', 466424, 505449], ['GRAPHIC', 505449, 505457], ['IMG', 505457, 532605], ['GRAPHIC',
581464, 581472], ['IMG', 581472, 625730], ['GRAPHIC', 625730, 625738], ['IMG', 625738,
663565], ['GRAPHIC', 663565, 663573], ['IMG', 663573, 709300], ['GRAPHIC', 709300, 709
308], ['IMG', 709308, 747861], ['GRAPHIC', 747861, 747869], ['IMG', 747869, 796348], [
'GRAPHIC', 796348, 796356], ['IMG', 796356, 843366], ['GRAPHIC', 843366, 843374], ['IM
G', 843374, 849197]]

Unallocated spaces (Boundaries):
[[82000, 147254], [163914, 390449], [409669, 466416], [532605, 581464]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[]

More specific unallocated spaces: [[81919, 112640], [163839, 225280], [245759, 276480]
, [307199, 358400], [409599, 450560], [532479, 563200]]
Above spaces are filled by: b'\x00'

Start of block: [81919, 133119, 153599, 184319, 235519, 317439]

Press Enter to check again...
```

In bovenstaande afbeelding zijn de eerste resultaten van het Python script te zien. Onderstaand staan de betekenissen van de waardes beschreven:

- Verified values: De datablocks die zijn gevalideerd door de parser in het script;
- Unallocated spaces: De datablocks die zijn overgebleven en dus nog geen waarde toegewezen hebben gekregen;
- Broken Values: Alle datablocks die corrupte gegevens bevatten die tot nu toe zijn vastgesteld;
- Problems found in IMG Blocks: De grenzen van de corrupte datablocks in de image descriptor van een GIF-bestand;
- More specific unallocated spaces: Dit zijn de datablocks, waarbij is vastgesteld dat deze dezelfde data hebben. In "Above spaces are filled by" staat welke waarde dit is.
- Start of block: Dit zijn de beginpunten van de datablocks in het nieuwe bestand. Dit houdt in dat dit de beginpunten zijn geweest van de nulblokken. De waardes zijn aangepast naar het formaat van het nieuwe bestand.

In bovenstaande afbeelding worden offsets in decimalen weergegeven. Dit houdt dus in dat in het bestand op de genoemde locaties bij “More specific unallocated spaces” de grenzen te vinden zijn van de datablocks die gevuld zijn met byteobject *\x00*. Zie onderstaande afbeelding als voorbeeld:




Dit houdt dus specifiek in dat de volgende handelingen zijn uitgevoerd door het script:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset dec) | Rechts block (offset dec) | Lengte (dec) |
|---|---|---|---|
| \x00 | 81919 | 112640 | 30721 |
| \x00 | 163839 | 225280 | 61441 |
| \x00 | 245759 | 276480 | 30721 |
| \x00 | 307199 | 358400 | 51201 |
| \x00 | 409599 | 450560 | 40961 |
| \x00 | 532479 | 563200 | 30721 |

Aangezien de inhoud van deze datablocks gelijk was aan *\x00*, voldoet deze niet aan een structuur dat gedefinieerd is in de GIF-specificaties. Met deze reden zijn deze datablocks verwijderd.

Na de eerste bewerking om de nulblokken te verwijderen is het script nog eens uitgevoerd om te controleren of er nog fouten in het bestand staan:

```
Verified Values:
[['APPLIC', 781, 800], ['GRAPHIC', 800, 808], ['IMG', 808, 67132], ['GRAPHIC', 67132,
67140], ['IMG', 67140, 116534], ['GRAPHIC', 116534, 116542], ['IMG', 116542, 191276],
['GRAPHIC', 216369, 216377], ['IMG', 216377, 251376], ['GRAPHIC', 251376, 251384], ['I
MG', 251384, 290409], ['GRAPHIC', 290409, 290417], ['IMG', 290417, 335704], ['GRAPHIC'
, 335704, 335712], ['IMG', 335712, 379970], ['GRAPHIC', 379970, 379978], ['IMG', 37997
8, 417805], ['GRAPHIC', 417805, 417813], ['IMG', 417813, 463540], ['GRAPHIC', 463540,
463548], ['IMG', 463548, 502101], ['GRAPHIC', 502101, 502109], ['IMG', 502109, 550588]
, ['GRAPHIC', 550588, 550596], ['IMG', 550596, 597606], ['GRAPHIC', 597606, 597614], [
'IMG', 597614, 603437]]

Unallocated spaces (Boundaries):
[[191276, 216369]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[134930, 153618], [156980, 184372]]

Press Enter to check again...
```

Uit bovenstaande resultaten is op te maken dat er specifieke structuurproblemen zijn in een image descriptor block of meerdere image descriptor blocks nog structuurfouten zitten. Aan de hand van de offsets wordt bepaald waar de problematische datablokken zich bevinden.

De waardes die bovenstaand worden weergegeven zijn waardes die een indicatie aangeven waar het probleem zich ongeveer bevindt. Deze zijn gebaseerd de doorbroken Data Block Size reeks die zich voordoet.  Door te zoeken naar de grootste Data Block Size die in de buurt ligt (FF) kan middels de functie *Go To* (Jump width = 255 instellen of FF) worden bepaald waar de Data Blocks verbroken worden.

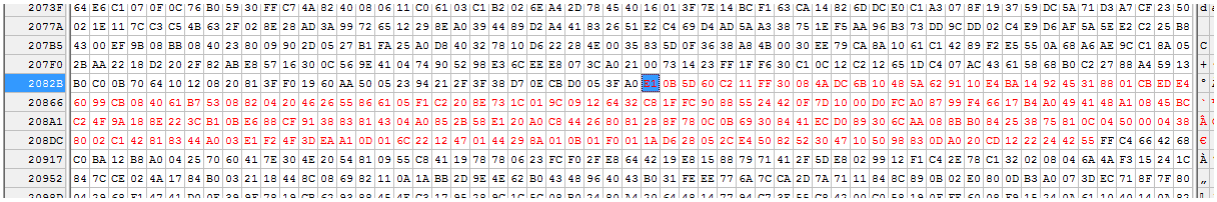
Op hex offset: 0x20812 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x20749 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

In dit onderdeel is het mogelijk dat óf de header van de image descriptor mist of er is data midden in het block overschreven. Uit de eerder weergegeven afbeeldingen kan worden opgemaakt dat de start van een nieuw block, namelijk bij decimale offset 133119 in de buurt ligt van dit foutieve block. Het is dus mogelijk dat er data mist of data overschreven is. Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 20849 | 20911 | 200 |



Om de nieuwe locaties te bepalen wordt het script nogmaals uitgevoerd:

```
Verified Values:
[['APPLIC', 781, 800], ['GRAPHIC', 800, 808], ['IMG', 808, 67132], ['GRAPHIC', 67132,
67140], ['IMG', 67140, 116534], ['GRAPHIC', 116534, 116542], ['IMG', 116542, 191075],
['GRAPHIC', 216168, 216176], ['IMG', 216176, 251175], ['GRAPHIC', 251175, 251183], ['I
MG', 251183, 290208], ['GRAPHIC', 290208, 290216], ['IMG', 290216, 335503], ['GRAPHIC'
, 335503, 335511], ['IMG', 335511, 379769], ['GRAPHIC', 379769, 379777], ['IMG', 37977
7, 417604], ['GRAPHIC', 417604, 417612], ['IMG', 417612, 463339], ['GRAPHIC', 463339,
463347], ['IMG', 463347, 501900], ['GRAPHIC', 501900, 501908], ['IMG', 501908, 550387]
, ['GRAPHIC', 550387, 550395], ['IMG', 550395, 597405], ['GRAPHIC', 597405, 597413], [
'IMG', 597413, 603236]]

Unallocated spaces (Boundaries):
[[191075, 216168]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[156779, 184171]]

Press Enter to check again...
```

Ln: 57 Col: 0

Voor dit foutieve block geldt dezelfde werkwijze als hiervoor beschreven.

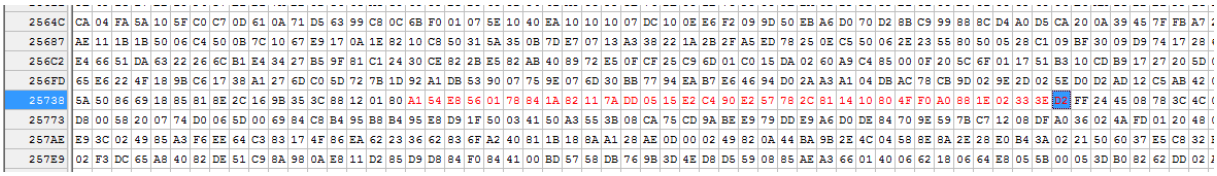
Op hex offset: 0x2576B is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x25649 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Ook in dit onderdeel is het mogelijk dat óf de header van image descriptor mist of er is data midden in het block overschreven. Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 25749 | 2576A | 33 |



Na nogmaals uitvoeren van het script blijkt dat er nog een stuk unallocated space te vinden is:

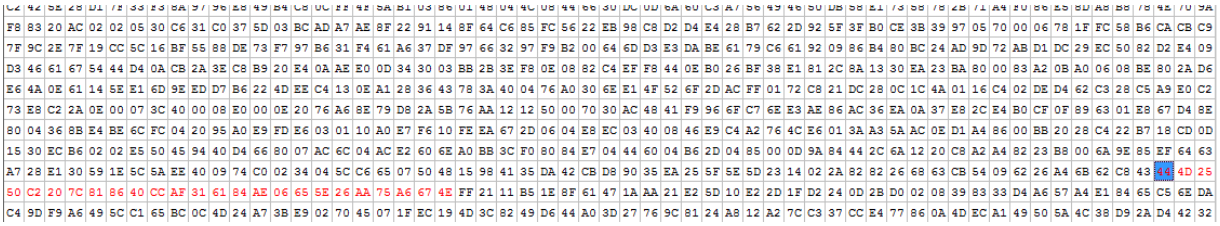
```
Verified Values:
[['APPLIC', 781, 800], ['GRAPHIC', 800, 808], ['IMG', 808, 67132], ['GRAPHIC', 67132,
67140], ['IMG', 67140, 116534], ['GRAPHIC', 116534, 116542], ['IMG', 116542, 191041],
['GRAPHIC', 216134, 216142], ['IMG', 216142, 251141], ['GRAPHIC', 251141, 251149], ['I
MG', 251149, 290174], ['GRAPHIC', 290174, 290182], ['IMG', 290182, 335469], ['GRAPHIC'
, 335469, 335477], ['IMG', 335477, 379735], ['GRAPHIC', 379735, 379743], ['IMG', 37974
3, 417570], ['GRAPHIC', 417570, 417578], ['IMG', 417578, 463305], ['GRAPHIC', 463305,
463313], ['IMG', 463313, 501866], ['GRAPHIC', 501866, 501874], ['IMG', 501874, 550353]
, ['GRAPHIC', 550353, 550361], ['IMG', 550361, 597371], ['GRAPHIC', 597371, 597379], [
'IMG', 597379, 603202]]

Unallocated spaces (Boundaries):
[[191041, 216134]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[]

Press Enter to check again...
```

Naar aanleiding van de hierboven gegeven informatie wordt de cursor verplaatst naar de eerder bepaalde locatie van de voormalige start van een nulblock. De waarde die het dichtst bij de beginwaarde van unallocated spaces ligt wordt gekozen. Dit is offset: 184319. Deze klopt echter niet meer aangezien er reeds waardes zijn verwijderd. Er moet dus boven deze offset worden gezocht naar veranderde Data Block Sizes. Hiervoor geldt dezelfde werkwijze als voor de andere twee vondsten.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 2CF49 | 2CF61 | 24 |



Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['APPLIC', 781, 800], ['GRAPHIC', 800, 808], ['IMG', 808, 67132], ['GRAPHIC', 67132,
67140], ['IMG', 67140, 116534], ['GRAPHIC', 116534, 116542], ['IMG', 116542, 216109],
['GRAPHIC', 216109, 216117], ['IMG', 216117, 251116], ['GRAPHIC', 251116, 251124], ['I
MG', 251124, 290149], ['GRAPHIC', 290149, 290157], ['IMG', 290157, 335444], ['GRAPHIC'
, 335444, 335452], ['IMG', 335452, 379710], ['GRAPHIC', 379710, 379718], ['IMG', 37971
8, 417545], ['GRAPHIC', 417545, 417553], ['IMG', 417553, 463280], ['GRAPHIC', 463280,
463288], ['IMG', 463288, 501841], ['GRAPHIC', 501841, 501849], ['IMG', 501849, 550328]
, ['GRAPHIC', 550328, 550336], ['IMG', 550336, 597346], ['GRAPHIC', 597346, 597354], [
'IMG', 597354, 603177]]

Unallocated spaces (Boundaries):
[]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[]
```

Hieruit blijkt dat er geen foutieve waardes meer zijn gevonden en dat het script geen unallocated spaces meer heeft vastgesteld in het bestand. Daarbij worden de lengtes van het begin van het bestand tot eerste waarde en laatste waarde tot einde van bestand niet meegenomen.

Het bestand is gecontroleerd op validiteit door meerdere tools zoals “Jeffrey's Exif Viewer” op http://regex.info/exif.cgi en http://www.imagemagick.org/MagickStudio/scripts/MagickStudio.cgi. Doordat ImageMagick is een bibliotheek die parsers en interpreters bevat om afbeeldingen te kunnen verwerken. Bij de eerste tool werd geen fout weergegeven, maar bij gebruik van de tweede tool werd de fout “425: Corrupt Image” weergegeven. Naar aanleiding hiervan is in Jeffrey’s Exif Viewer gecontroleerd of er corrupte data op enkele frames stond. Er was geen duidelijke foutmelding zichtbaar in de Exif viewer, maar er stonden wel enkele frames tussen die image data misten. Als hypothese voor het vervolgonderzoek werd gesteld: ImageMagick is niet in staat zelf opvulling te geven aan missende image data. Om het bestand valide te maken voor alle parsers en interpreters zijn de frames 3 en 4 van bestand *e_r.data* verwijderd. Frames zijn te herkennen aan een Graphic Control Extension gevolgd door een Image Descriptor met Block Terminator.

Om deze wijzigingen door te voeren zijn de volgende handelingen in volgorde uitgevoerd:

1. Gezocht naar hex string: 00 21 F9 met XVI32.

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie bijlage: e_r_removed.txt (Block 1) | 1C736 | 34C2C | 99574 |
| Zie bijlage: e_r_removed.txt (Block 2) | 1C736 | 24FF4 | 35006 |

Na het verwijderen van beide frames bleek de GIF-afbeelding wel valide te zijn. Aangezien het format voldeed aan de specificatie van GIF, werden er geen fouten meer weergegeven door het python

script. De foutmelding die eerder via imagemagick werd weergegeven kwam niet meer voor. De herstelde afbeelding bestaat uit 11 functionerende frames.

Het bestand is te openen in een photo viewer en er wordt een animatie weergegeven. Er bevindt zich echter nog wat ruis in de afbeelding. In de uitvoer van het script onder “Verified Values” is te zien dat er één IMG block (Image Descriptor Block) veel groter is dan de rest van de Image Descriptors *[‘IMG’, 116542, 216109]*. Dit zou kunnen betekenen dat er een header en een einde van een Image Descriptor zijn weggevallen. Dit kan resulteren in het missen van bijvoorbeeld een frame of andere image data. Nadere uitleg staat in hoofdstuk 2. Zie de bijlage op github voor het resultaat.

Onderstaande resultaten hebben betrekking op het bestand *f_z.data*.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 102624], ['GRAPHIC', 172734, 172742], ['IMG', 172742, 204884], ['GR
APHIC', 273641, 273649], ['IMG', 273649, 307321], ['GRAPHIC', 374013, 374021], ['IMG', 3
74021, 409733], ['GRAPHIC', 449311, 449319], ['IMG', 449319, 471261], ['GRAPHIC', 545263
, 545271], ['IMG', 545271, 593988], ['GRAPHIC', 640098, 640106], ['IMG', 640106, 715151]
, ['GRAPHIC', 715151, 715159], ['IMG', 715159, 791819], ['GRAPHIC', 791819, 791827], ['I
MG', 791827, 866593], ['GRAPHIC', 866593, 866601], ['IMG', 866601, 942432]]

Unallocated spaces (Boundaries):
[[102624, 172734], [204884, 273641], [307321, 374013], [409733, 449311], [471261, 545263
], [593988, 640098]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[]

More specific unallocated spaces: [[102399, 122880], [204799, 245760], [307199, 327680],
[348159, 368640], [409599, 430080], [471039, 491520], [511999, 532480], [593919, 614400]
]
Above spaces are filled by: b'\x00'

Start of block: [102399, 184319, 245759, 266239, 307199, 348159, 368639, 430079]
```

De betekenissen van de waardes die hierboven zijn weergegeven zijn beschreven in het gedeelte van *e_r.data*. In bovenstaande afbeelding worden offsets in decimalen weergegeven. Dit houdt dus in dat in het bestand op de genoemde locaties bij “More specific unallocated spaces” de grenzen te vinden zijn van de datablocks die gevuld zijn met byteobject *\x00*.

Dit houdt dus specifiek in dat de volgende handelingen zijn uitgevoerd door het script:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset dec) | Rechts block (offset dec) | Lengte (dec) |
|---|---|---|---|
| \x00 | 102399 | 122880 | 20481 |
| \x00 | 204799 | 245760 | 40961 |
| \x00 | 307199 | 327680 | 20481 |
| \x00 | 348159 | 368640 | 20481 |
| \x00 | 409599 | 430080 | 20481 |
| \x00 | 471039 | 491520 | 20481 |

Aangezien de inhoud van deze datablocks gelijk was aan *\x00*, voldoet deze niet aan een structuur dat gedefinieerd is in de GIF-specificaties. Met deze reden zijn deze datablocks verwijderd.

Na de eerste bewerking om de nulblokken te verwijderen is het script nog eens uitgevoerd om te controleren of er nog fouten in het bestand staan:

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 186646], ['GR
APHIC', 212201, 212209], ['IMG', 212209, 264713], ['GRAPHIC', 271613, 271621], ['GRAPHIC
', 326431, 326439], ['IMG', 271621, 381423], ['IMG', 326439, 381423], ['GRAPHIC', 381423
, 381431], ['IMG', 381431, 455778], ['GRAPHIC', 455778, 455786], ['IMG', 455786, 530831]
, ['GRAPHIC', 530831, 530839], ['IMG', 530839, 607499], ['GRAPHIC', 607499, 607507], ['I
MG', 607507, 682273], ['GRAPHIC', 682273, 682281], ['IMG', 682281, 758112]]

Unallocated spaces (Boundaries):
[[186646, 212201], [264713, 271613], [271621, 326431]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[351440, 368780], [379410, 381195], [379410, 381195]]

Press Enter to check again...
```

Uit bovenstaande resultaten is op te maken dat er specifieke structuurproblemen zijn in een image descriptor block of meerdere image descriptor blocks nog structuurfouten zitten. Aan de hand van de offsets wordt bepaald waar de problematische datablokken zich bevinden.

Om te bepalen of er sprake is van afbraak van Data Block Sizes is dezelfde werkwijze gehanteerd als bij *e_r.data*. Het verschil is echter dat in bestand *f_z.data* gebruik wordt gemaakt van *FE* als data block size en niet *FF.* Er zijn nog zes blocks waar potentiële fouten in zitten. Ook hier wordt de dichtstbijzijnde waarde gezocht per block om het begin vast te stellen. Deze wordt uit de eerste afbeelding gehaald van *f_z.data* bij “start of block”.

Voor unallocated space offset 186646, geldt dat offset 184319 de positie aangeeft van de voorheen door nullen gesplitste blocks. De resultaten bij het zoeken naar de juiste Data Block Size zijn de volgende:

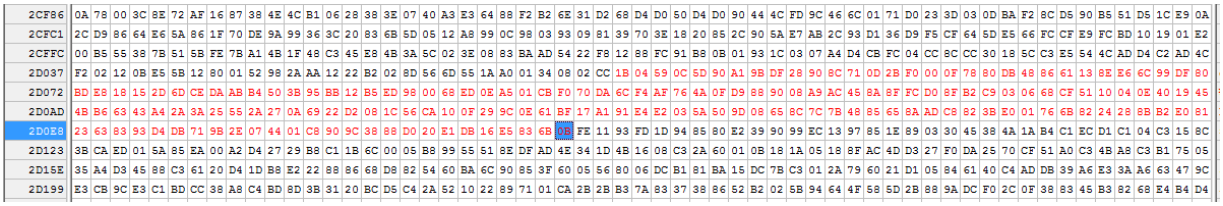
Op hex offset: 0x2D003 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x2CF54 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 2D053 | 2D101 | 174 |



Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 212026], ['GR
APHIC', 212026, 212034], ['IMG', 212034, 264538], ['GRAPHIC', 271438, 271446], ['GRAPHIC
', 326256, 326264], ['IMG', 271446, 381248], ['IMG', 326264, 381248], ['GRAPHIC', 381248
, 381256], ['IMG', 381256, 455603], ['GRAPHIC', 455603, 455611], ['IMG', 455611, 530656]
, ['GRAPHIC', 530656, 530664], ['IMG', 530664, 607324], ['GRAPHIC', 607324, 607332], ['I
MG', 607332, 682098], ['GRAPHIC', 682098, 682106], ['IMG', 682106, 757937]]

Unallocated spaces (Boundaries):
[[264538, 271438], [271446, 326256]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[351265, 368605], [379235, 381020], [379235, 381020]]

Press Enter to check again...
```

Voor unallocated space offset 264538, geldt dat offset 266239 de positie aangeeft van de voorheen door nullen gesplitste blocks. Deze waarde is inmiddels kleiner, omdat de data ervoor is aangepast. Het geeft echter een indicatie om in de buurt te zoeken. De resultaten bij het zoeken naar de juiste Data Block Size zijn de volgende:

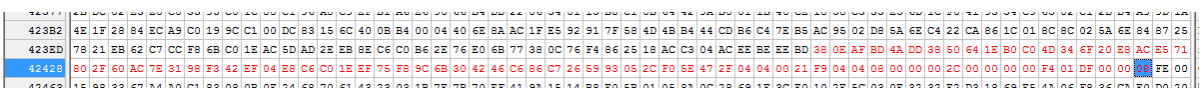
Op hex offset: 0x42461 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x42315 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 42414 | 42460 | 76 |



Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 212026], ['GR
APHIC', 212026, 212034], ['IMG', 212034, 264538], ['GRAPHIC', 326179, 326187], ['IMG', 3
26187, 381171], ['GRAPHIC', 381171, 381179], ['IMG', 381179, 455526], ['GRAPHIC', 455526
, 455534], ['IMG', 455534, 530579], ['GRAPHIC', 530579, 530587], ['IMG', 530587, 607247]
, ['GRAPHIC', 607247, 607255], ['IMG', 607255, 682021], ['GRAPHIC', 682021, 682029], ['I
MG', 682029, 757860]]

Unallocated spaces (Boundaries):
[[264538, 326179]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[379158, 380943]]

Press Enter to check again...
```

Uit bovenstaande afbeelding blijkt dat dezelfde unallocated space nog steeds is opgenomen in het bestand. Dit houdt in dat er een andere unallocated space is opgelost in plaats van deze. Om de fout van deze unallocated space alsnog te vinden moet worden teruggezocht vanaf offset 245759 of de *FE*-reeks meerdere malen doorbroken wordt. De resultaten bij het zoeken naar de juiste Data Block Size zijn de volgende:

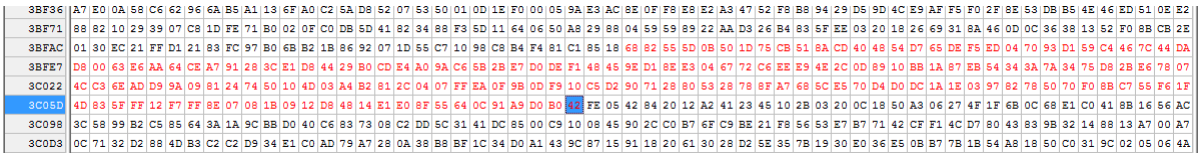
Op hex offset: 0x3BF79 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x3BECA is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 3BFC9 | 3C077 | 174 |



Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 212026], ['GR
APHIC', 212026, 212034], ['GRAPHIC', 326004, 326012], ['IMG', 212034, 380996], ['IMG', 3
26012, 380996], ['GRAPHIC', 380996, 381004], ['IMG', 381004, 455351], ['GRAPHIC', 455351
, 455359], ['IMG', 455359, 530404], ['GRAPHIC', 530404, 530412], ['IMG', 530412, 607072]
, ['GRAPHIC', 607072, 607080], ['IMG', 607080, 681846], ['GRAPHIC', 681846, 681854], ['I
MG', 681854, 757685]]

Unallocated spaces (Boundaries):
[[212034, 326004]]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[351013, 368353], [378983, 380768], [378983, 380768]]

Press Enter to check again...
```

Uit bovenstaande afbeelding blijkt dat er nieuwe fouten zijn opgetreden. Dit kan zijn door een foutieve wijziging of een juiste wijziging van het bestand. Omdat het bestand niet meer voldoet aan de oude structuur kan het zo zijn dat de structuur later in het bestand door elkaar raakt. Aangezien het begin van block 212034 – 326004 al onderzocht is wordt gekeken naar de laatste waarde (307199 ligt dichtstbij).

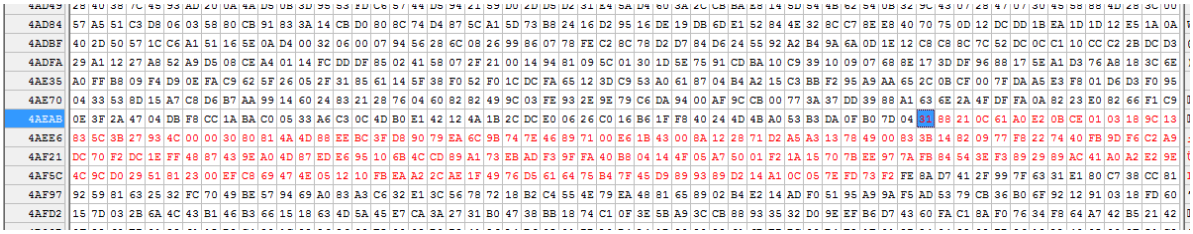
Op hex offset: 0x4AE89 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x4ADDA is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 4AED9 | 4AF87 | 174 |



In bovenstaande bewerking wordt 31 **niet** meegenomen.

Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 212026], ['GR
APHIC', 212026, 212034], ['IMG', 212034, 325829], ['GRAPHIC', 325829, 325837], ['IMG', 3
25837, 380821], ['GRAPHIC', 380821, 380829], ['IMG', 380829, 455176], ['GRAPHIC', 455176
, 455184], ['IMG', 455184, 530229], ['GRAPHIC', 530229, 530237], ['IMG', 530237, 606897]
, ['GRAPHIC', 606897, 606905], ['IMG', 606905, 681671], ['GRAPHIC', 681671, 681679], ['I
MG', 681679, 757510]]

Unallocated spaces (Boundaries):
[]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[[378808, 380593]]

Press Enter to check again...
```

In een van de Image Descriptors zit nog een fout. Dezelfde onderzoekswijze wordt toegepast zoals hierboven beschreven met offset 368639.

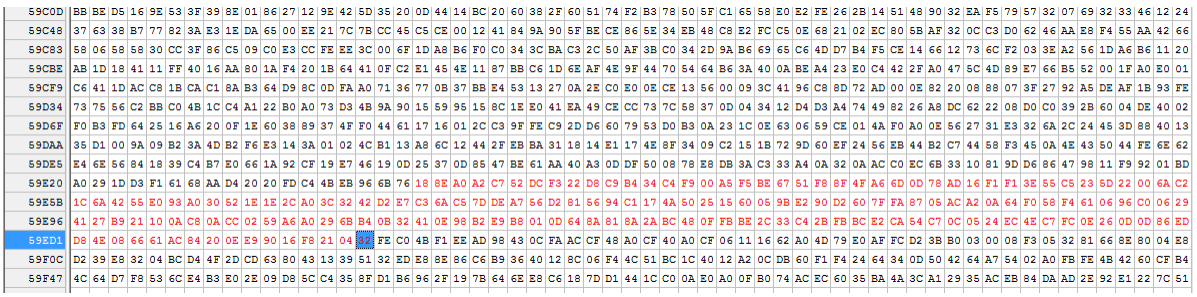
Op hex offset: 0x59DE2 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock eronder.

Op hex offset: 0x59D33 is de laatste Data Block Size te vinden die voldoet aan de rest van de Data Block sizes in het datablock erboven.

Om de reeks van data block sizes kloppend te maken is de volgende bewerking uitgevoerd:

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie onderstaande afbeelding | 59E32 | 59EE0 | 174 |



Om te bepalen of er nog nieuwe fouten zijn opgetreden in het bestand wordt het script nogmaals uitgevoerd.

```
Verified Values:
[['GRAPHIC', 781, 789], ['APPLIC', 789, 808], ['IMG', 808, 76339], ['GRAPHIC', 76339, 76
347], ['IMG', 76347, 152254], ['GRAPHIC', 152254, 152262], ['IMG', 152262, 212026], ['GR
APHIC', 212026, 212034], ['IMG', 212034, 325829], ['GRAPHIC', 325829, 325837], ['IMG', 3
25837, 380646], ['GRAPHIC', 380646, 380654], ['IMG', 380654, 455001], ['GRAPHIC', 455001
, 455009], ['IMG', 455009, 530054], ['GRAPHIC', 530054, 530062], ['IMG', 530062, 606722]
, ['GRAPHIC', 606722, 606730], ['IMG', 606730, 681496], ['GRAPHIC', 681496, 681504], ['I
MG', 681504, 757335]]

Unallocated spaces (Boundaries):
[]

Broken Values (Boundaries):
[]

Problems found in IMG Blocks:
[]
```

Hieruit blijkt dat er geen foutieve waardes meer zijn gevonden en dat het script geen unallocated spaces meer heeft vastgesteld in het bestand. Daarbij worden de lengtes van het begin van het bestand tot eerste waarde en laatste waarde tot einde van bestand niet meegenomen.

Er bevindt zich echter nog wat ruis in de afbeelding. In de uitvoer van het script onder “Verified Values” is te zien dat er één IMG block (Image Descriptor Block) veel groter is dan de rest van de Image Descriptors *[‘IMG’, 212034, 325829]*. Dit zou kunnen betekenen dat er een header en een einde van een Image Descriptor zijn weggevallen. Dit kan resulteren in het missen van bijvoorbeeld een frame of andere image data. Nadere uitleg staat in hoofdstuk 2.

Het bestand is gecontroleerd op validiteit door meerdere tools zoals “Jeffrey's Exif Viewer” op http://regex.info/exif.cgi en http://www.imagemagick.org/MagickStudio/scripts/MagickStudio.cgi. Doordat ImageMagick is een bibliotheek die parsers en interpreters bevat om afbeeldingen te kunnen verwerken. Bij de eerste tool werd geen fout weergegeven, maar bij gebruik van de tweede tool werd de fout “425: Corrupt Image” weergegeven. Naar aanleiding hiervan is in Jeffrey’s Exif Viewer gecontroleerd of er corrupte data op enkele frames stond. Er was geen duidelijke foutmelding zichtbaar in de Exif viewer, maar er stonden wel enkele frames tussen die image data misten. Om het bestand valide te maken voor alle parsers en interpreters zijn de frames 3, 4 en 5 van bestand *f_z.data* verwijderd. Daarnaast is de Graphic Control Header op positie 781 (decimaal) omgewisseld met de Application Extension op positie 789 (decimaal), omdat de Graphic Control Header invloed heeft op image data. De instellingen in de Graphic Control Header hebben invloed op het eerstvolgende block. Frames zijn te herkennen aan een Graphic Control Extension gevolgd door een Image Descriptor met Block Terminator.

Om deze wijzigingen door te voeren zijn de volgende handelingen in volgorde uitgevoerd:

1. Gezocht naar hex string: 00 21 F9 met XVI32.

**Verwijderde datablocks:**

| Gevonden Data | Links block (offset hex) | Rechts block (offset hex) | Lengte (dec) |
|---|---|---|---|
| Zie bijlage: f_z_removed.txt (Block 1) | 4F8C5 | 5CEE5 | 99574 |
| Zie bijlage: f_z_removed.txt (Block 2) | 33C39 | 4F8C4 | 35006 |
| Zie bijlage: f_z_removed.txt (Block 3) | 252BE | 33C39 | 59771 |

Na het verwijderen van beide frames bleek de GIF-afbeelding wel valide te zijn. Aangezien het format voldeed aan de specificatie van GIF, werden er geen fouten meer weergegeven door het python script. De foutmelding die eerder via imagemagick werd weergegeven kwam niet meer voor. De herstelde afbeelding bestaat uit 11 functionerende frames.

Het bestand is te openen in een photo viewer en er wordt een animatie weergegeven. Zie de bijlage op github voor het resultaat.

# 2. How we found it

Om tot de hierboven beschreven resultaten te komen is er een Python script ontworpen door groep 4. In dit gedeelte zal beschreven worden hoe dit Python script tot stand is gekomen en het zal een korte handleiding bevatten om het Python script te bedienen.

## 2.1 Handelingen voor resultaten

In dit gedeelte worden de uitgevoerde handelingen beschreven om tot de hierboven beschreven resultaten te komen.

### 2.1.1 Python script

Om de resultaten in het Python script weer te geven is het Python script uitgevoerd. De bestanden *f_z.data* en *e_r.data* moet in dezelfde map staan als het Python script. In 2.3 Uitleg Python script wordt beschreven hoe het Python script is opgebouwd. In 2.2 Controle uitvoer Python script wordt de controle van de uitvoer van het Python script beschreven. Ter bediening van het script hoeft de gebruiker alleen gebruik te maken van de Enter toets en op meerdere plekken in het bestand de naam van het input-bestand in het script te wijzigen.

## 2.2 Controle uitvoer Python script & zoeken naar afwijkende data block sizes

De uitvoer van het Python script wordt gecontroleerd met een hex-editor (XVI32). In de hex-editor zijn de bestanden *f_z.data* en *e_r.data* geopend.

In het python script worden boundaries als resultaten gegenereerd. Deze boundaries worden gebruikt om de locaties vast te stellen van de problematische image descriptor blocks. De kans is groot dat er problemen optreden met de data in het bestand op de plekken waar de nulblocks verwijderd zijn. Daarom wordt de start waarde van deze blocks opgeslagen en wordt deze vergeleken met de waarde waar ongeveer de fout in zou moeten zitten. In de buurt van deze waardes is gezocht naar een afwijking in de data block size.

Bij afbeelding *e_r.data* werd voornamelijk gebruik gemaakt van *FF* als data block size, terwijl bij afbeelding *f_z.data* voornamelijk gebruik werd gemaakt van *FE* als data block size. Aangezien deze twee waardes bekend zijn, is er gezocht naar deze waardes in het door het script gespecificeerde block. Om te controleren of de data block sizes overeenkomen, kan gebruik worden gemaakt van de *Jump/Goto* functie in XVI32. Deze functie is in staat om vanaf de offset van één *FF* te springen naar de volgende, volgens GIF-specificaties gedefinieerde, data size block. Als deze meermaals afwijkt van de waarde *FF* dan kan het zo zijn dat er foutieve data in staat. Daarom wordt rond het punt waar de laatste kloppende *FF* is gevonden gezocht naar een andere *FF* die wel klopt in het logische block dat volgt. Dit was het geval en dus kon er een stuk van de data tussen deze *FF's* verwijderd worden om de lengte tussen deze specificaties kloppend te maken.

Uit enkele sizes van image descriptors was op te maken dat deze veel groter waren dan de rest van de image descriptors. Binnen deze image descriptors waren dan ook fouten opgetreden. Om deze fout te corrigeren is er een block data verwijderd. Aangezien deze image descriptors veel groter waren, bestaat de mogelijkheid dat de image descriptor header van een andere image descriptor is overschreven, waardoor het block dat volgt niet meer wordt herkend als image descriptor. In deze gevallen is de image data dan ook samen gevoegd.

## 2.3 Uitleg Python script

Het Python script is te vinden in de bijbehorende Github-repository van groep 4. Onderstaand worden delen van het script kort uitgelegd.

```
from struct import *
import sys
import os
import binascii
import re
import random
```

In bovenstaande afbeeldingen zijn enkele libraries weergegeven waarvan gebruik wordt gemaakt in het script. Struct is een library om binaire data te herkennen als unsigned integer, unsigned short etc. Sys wordt gebruikt om bestandspaden nader te definiëren. Os wordt gebruikt om de grootte van een bestand te kunnen bepalen. Binascii wordt gebruikt om te converteren van binaire data naar hexadecimaal of omgekeerd. Re wordt gebruikt om data in een bestand te zoeken op basis van een regex string. Random wordt op dit moment niet gebruikt, het was een module die gebruikt zou worden om data te ordenen op basis van random gegenereerde nummers.

```
#GIF Header definitie
intDefinition = 6 #Signature + Version = Length 6 Bytes

#Logical screen descriptor chunk
#Should be read as little endian
intWidth = 2
intHeight = 2
intPacked = 1
intBackgroundColorandAspect = 2
```

In bovenstaande afbeelding zijn voor enkele waardes zoals de GIF header en de Logical Screen Descriptor beschreven uit welke elementen ze bestaan en wat de grootte in bytes is van deze elementen.

```
#Function for finding left boundary for specific data block
def CheckBlockLeft(f, currentPos, strBinary):
    blnEndOfData = False
    count = 1
    while blnEndOfData == False:
        f.seek(currentPos - count)
        if file.read(1) == strBinary:
            count = count + 1
        else:
            blnEndOfData = True
            if count == 1:
                return False
            else:
                return (f.tell()-1)
```

In bovenstaande afbeelding wordt de functie *CheckBlockLeft* weergegeven. Dit is een functie die aan de hand van ingevoerde data de linkergrens van een block met dezelfde ingevoerde data kan vaststellen en terugsturen. De functie *CheckBlockRight* wordt gebruikt voor dezelfde doeleinden, maar dan voor de rechtergrens.

```
#Function to check the validity of the image descriptor block
def CheckIMGDescriptor(f, currentPos, MaxWidth, MaxHeight):
    f.seek(currentPos)
    f.read(5)
    arrWrongBoundaries = []
    intWidth = unpack("<H",f.read(2))[0]
    intHeight = unpack("<H", f.read(2))[0]
    if (intWidth > MaxWidth or intHeight > MaxHeight) or (intWidth == 0 and intHeight == 0):
        return 0, f.tell(), arrWrongBoundaries
    intPacked = str(bin(unpack("<B", f.read(1))[0]))
    arrPacked = list(intPacked[2:len(intPacked)])
    if arrPacked[0] == "1":
        intColorTable = int(intPacked[len(intPacked)-3:len(intPacked)],2)
        intSizeLocalColor = 3*2**(intColorTable+1)
        LocalColor = f.read(intSizeLocalColor)
    f.read(1)
    blnTerminated = False
    count = 0
    arrDBSize = [] #Used to define biggest size used, mostly only a few blocks differ from this
    while blnTerminated == False:
        blockPos = f.tell()
        intDBSize = unpack("B", f.read(1))[0]
        if intDBSize <= 255 and intDBSize > 0: #It is very unlikely for image data to be 0.
            f.read(intDBSize)
            arrDBSize.append(intDBSize)
            if intDBSize < max(arrDBSize):
                arrWrongBoundaries.append(blockPos)
                arrWrongBoundaries.append(f.tell())
            intBlockTerm = unpack("B", f.read(1))[0]
            if intBlockTerm == 0:
                blnTerminated = True
                return 2, f.tell(), arrWrongBoundaries
            else:
                f.seek(f.tell()-1)
        else:
            return 2, f.tell(), arrWrongBoundaries
```

In bovenstaande afbeelding wordt de functie *CheckIMGDescriptor* weergegeven. Deze functie wordt gebruikt om de validiteit van de Image Descriptor te controleren. Er wordt gecontroleerd op de volgende factoren:

- Zijn de Width en Height groter dan de Width en Height gedefinieerd in de Logical Screen Descriptor?
- Is er sprake van een local color table?
- Is de gespecificeerde data block size kleiner of gelijk aan 255 of groter dan 0?
- Is de gespecificeerde data block size in overeenkomst met de grootste waarde die voorkomt in de lijst vorige data block sizes?

```
#Function to check the validity of the Application Extension header
def CheckApplicChunk(f, currentPos):
    f.seek(currentPos+2)
    intBlockSize = unpack("B", f.read(1))[0]
    f.read(11)
    intDBSize = unpack("B", f.read(1))[0]
    strData = f.read(intDBSize)
    intBlockTerm = unpack("B", f.read(1))[0]
    if intBlockSize == 11 and intBlockTerm == 0:
        return 2, f.tell()
    else:
        return 0, f.tell()
```

In bovenstaande afbeelding wordt de functie *CheckApplicChunk* weergeven. In deze functie wordt gecontroleerd of de Application Extension Chunk valide is. Dit wordt gedaan aan de hand van de *fixed* blocksize die altijd 11 moet zijn en te controleren of de Block terminator op de juiste

locatie staat. Hetzelfde wordt ook uitgevoerd in de functies *CheckPlainChunk, CheckGraphicChunk* en *CheckCommentChunk* met fixed blocksizes.

```
blnSolved = False
arrNew = []

while blnSolved == False:
    file = open("f_z.data", "rb")
    content = file.read()

    file.seek(0)
    magic_val = file.read(6)
    strWidth = unpack("<H",file.read(2))[0]
    strHeight = unpack("<H",file.read(2))[0]

    arrKnownValues = [["APPLIC", b"\x21\xff"],["GRAPHIC",b"\x21\xf9"],["COMMENT",b"\x21\xfe"],["PLAIN",b"\x21\x01"],["IMG", b"\x00\x2c"],["TRAILER",b"\x3b"]]
    arrPossibleValues =[]
    arrVerifiedValues = []
    arrBrokenValues = []
    arrUnallocated = []
    arrDataBlockBounds = []
```

Uit bovenstaande afbeelding blijkt dat er een *While-Loop* wordt uitgevoerd na de functie definities. Dit geldt voor de rest van de data. Deze *While-Loop* wordt gebruikt om bijvoorbeeld het bestand nogmaals te controleren op fouten. In dit gedeelte wordt ook gedefinieerd welk bestand in binaire leesmodus geopend moet worden. Vervolgens worden de eerste paar waardes van de GIF-header gelezen en opgeslagen. In het array *arrKnownValues* worden de fixed identifiers voor alle bekende blocks gedefinieerd. Deze komen uit de tabel op https://www.w3.org/Graphics/GIF/spec-gif89a.txt. Vervolgens worden er nog vier arrays gemaakt om waardes in op te slaan.

```
#Determine possible locations of specific headers
for i in range(0,len(arrKnownValues)):
    for m in re.finditer(arrKnownValues[i][1],content):
        arrPossibleValues.append([arrKnownValues[i][0],m.start()])
```

Aan de hand van de bekende data uit *arrKnownValues* wordt in het geopende bestand gezocht naar deze specifieke prefixes. Deze waardes worden toegewezen aan het array *arrPossibleValues*.

```
#Take action based on the found data
for i in range(0,len(arrPossibleValues)):
    if arrPossibleValues[i][0] == "APPLIC":
        test = CheckApplicChunk(file, arrPossibleValues[i][1])
        if test[0] != 0:
            # = Application block
            arrVerifiedValues.append([arrPossibleValues[i][0], arrPossibleValues[i][1], test[1]])
    elif arrPossibleValues[i][0] == "GRAPHIC":
        test = CheckGraphicChunk(file, arrPossibleValues[i][1])
        if test[0] != 0:
            # = Graphic Control Extension
            arrVerifiedValues.append([arrPossibleValues[i][0], arrPossibleValues[i][1], test[1]])
    elif arrPossibleValues[i][0] == "COMMENT":
        test = CheckCommentChunk(file, arrPossibleValues[i][1])
        if test[0] != 0:
            # = Comment Extension
            arrVerifiedValues.append([arrPossibleValues[i][0], arrPossibleValues[i][1], test[1]])
    elif arrPossibleValues[i][0] == "PLAIN":
        test = CheckPlainChunk(file, arrPossibleValues[i][1])
        if test[0] != 0:
            # = Plain Text Extension
            arrVerifiedValues.append([arrPossibleValues[i][0], arrPossibleValues[i][1], test[1]])
    elif arrPossibleValues[i][0] == "IMG":
        arrPossibleValues[i][1] = arrPossibleValues[i][1] + 1
        test = CheckIMGDescriptor(file, arrPossibleValues[i][1], strWidth, strHeight)
        if test[0] == 1:
            # = Broken Image Descriptor
            arrBrokenValues.append(arrPossibleValues[i])
        elif test[0] == 2:
            # = Image Descriptor
            arrVerifiedValues.append([arrPossibleValues[i][0], arrPossibleValues[i][1], test[1]])


        if isinstance(test[2], list):
            if len(test[2]) > 5:
                i = 1
                while i < len(test[2])-1:
                    if test[2][i+1]-test[2][i] > 0:
                        arrDataBlockBounds.append([test[2][i],test[2][i+1]])
                    i = i + 2
```

Voor alle waardes die zijn toegewezen aan *arrPossibleValues* wordt gecontroleerd of het block dat erachter zit daadwerkelijk het type block is die hoort bij de gespecifieerde identifier. Als dit het geval is worden de naam, linkergrens en rechtergrens van het datablock toegewezen aan het array *arrVerifiedValues*. Nadat dit is uitgevoerd wordt specifiek voor het image descriptor block extra gecontroleerd of de teruggegeven waarde van de functie *CheckIMGDescriptor* een list is. Als deze ook groter dan 5 is worden de linkergrens en rechtergrens van de foutieve data in de image descriptor toegewezen aan het array *arrDataBlockBounds*.

```
arrVerifiedValues = sorted(arrVerifiedValues, key=lambda x: x[2])
print("Verified Values:\n" + str(arrVerifiedValues))
```

Alle geverifieerde waarden worden gesorteerd op basis van de tweede waarde in het 2D-array. Dit is de rechtergrens (van klein naar groot). Deze waarden worden vervolgens weergegeven aan de gebruiker.

```
for i in range(0,len(arrVerifiedValues)-1):
    if (int(arrVerifiedValues[i+1][1]) - int(arrVerifiedValues[i][2])) > 0:
        arrUnallocated.append([arrVerifiedValues[i][2], arrVerifiedValues[i+1][1]])
print("\nUnallocated spaces (Boundaries):\n" + str(arrUnallocated))
print("\nBroken Values (Boundaries):\n" + str(arrBrokenValues))
print("\nProblems found in IMG Blocks:\n" + str(arrDataBlockBounds))
```

Uit bovenstaande afbeelding blijkt dat de grenzen van de unallocated spaces worden vastgesteld. Deze worden weergegeven aan de gebruiker, samen met de broken values en grenzen van de foutieve blocks in de image descriptor.

```
if not arrUnallocated and not arrBrokenValues and not arrDataBlockBounds:
    blnSolved = True
#elif arrNew:
    #if arrBrokenValues:
    #    strResult = Reorder(file, arrNew, brokenVal = arrBrokenValues)
    #else:
    #    strResult = Reorder(file,arrNew)
    #file.close()
    #file = open("e_r.data", "r+b")
    #file.seek(arrNew[0])
    #file.write(strResult)
    #file.close()
else:
    #Search for same character (filled with zeroes) and determine blocks
    arrSameChar = []
    for i in range(0,len(arrUnallocated)):
        file.seek(arrUnallocated[i][0])
        strTest = file.read(1)
        blnEndOfBlock = False
        count = 0
        while blnEndOfBlock == False:
            intLeftPos = CheckBlockLeft(file, arrUnallocated[i][0] + count, strTest)
            if intLeftPos != False:
                intRightPos = CheckBlockRight(file, arrUnallocated[i][0] + count, strTest)
                if intRightPos != False:
                    if [intLeftPos,intRightPos] not in arrSameChar:
                        arrSameChar.append([intLeftPos,intRightPos])
            count=count+10000 #check with steps of 10000 whether there is another block with same value in the same unallocated block
            if count > (300000):
                blnEndOfBlock = True
```

In bovenstaande afbeelding is een *if-statement* weergegeven. Deze statement bepaald of er waardes zijn in de arrays *arrUnallocated, arrBrokenValues of arrDataBlockBounds*. Als dit niet het geval is wordt de Boolean *blnSolved* die de *While-Loop* aan de gang houdt veranderd naar true. Als dit wel het geval is wordt bepaald of er grote blocks data uit het array *arrUnallocated* gelijk zijn aan elkaar. Hierbij worden de functies *CheckBlockLeft* en *CheckBlockRight* gebruikt. Deze waardes worden opgeslagen in het array *arrSameChar*.

```
#If blocks with same character found, continue to write edited data to file
if arrSameChar != []:
    file.seek(0)
    strFileSize = os.path.getsize("f_z.data")
    data = file.read(arrSameChar[0][0]+1)
    arrBlockMarkers = []
    if len(arrSameChar)==1:
        file.seek(arrSameChar[0][1])
    else:
        for i in range(0,len(arrSameChar)-1):
            file.seek(arrSameChar[i][1])
            data = data + file.read(arrSameChar[i+1][0]-(arrSameChar[i][1]-1))
            arrBlockMarkers.append(arrSameChar[i+1][0]-(arrSameChar[i][1]-1))
    arrNew.append(arrSameChar[0][0])
    for i in range(0,len(arrBlockMarkers)):
        arrNew.append(arrNew[i] + arrBlockMarkers[i])
    print("\nMore specific unallocated spaces: " + str(arrSameChar))
    print("Above spaces are filled by: " + str(strTest))
    print("\nStart of block: " + str(arrNew))

    try:
        file.seek(arrSameChar[len(arrSameChar)-1][1])
        data = data + file.read(strFileSize)
    except:
        print("EOF")
    file.close()
    f = open("f_z.data", "wb")
    f.write(data)
    f.close()
    file.close()
x = input("\nPress Enter to check again...")
```

Als er data is gespecificeerd in het array *arrSameChar* wordt de betreffende data in één variabele geplaatst (*data)*. Daarnaast worden de specifiekere unallocated spaces aangegeven uit *arrSameChar*. Deze heeft namelijk door de functies *CheckBlockRight* en *CheckBlockLeft* de nauwkeurige grenzen van de nulblokken kunnen vaststellen. Daarnaast worden ook de waarden weergegeven van de starts van de blokken (met nulwaarden) met behulp van *arrNew*.

De data die weggelaten moet worden wordt niet opgenomen in de variabele *data*. De gegevens die gedefinieerd zijn in variabele *data* worden vervolgens naar hetzelfde bestand als het input bestand geschreven. Vervolgens wordt de optie aangeboden om het script nogmaals uit te voeren door op Enter te drukken. Dit kan bijvoorbeeld gebruikt worden om te controleren of er meer fouten zitten in de blocks van een afbeelding.