Donkey Kong Randomizer

Randomized boards op basis van randomization op basis van een pointer in de code. Eventueel met mogelijkheid om de seed te bepalen op basis van dip switches en default een random seed…

Bij barrels altijd ladders gebruiken met een lengte met een veelvoud van vier. Dan zijn er niet zo heel veel plekken mogelijk, maar juist door het ook verschuiven van de girders in veelvouden van twee, kunnen ook de tussenliggende posities gebruikt worden. Wel wat lastiger om daar het algoritme voor vast te stellen. Maar heeft wel ook het voordeel dat de ladders nooit vlak naast elkaar zitten. In ieder geval ook gebruik maken van links/rechts op het segment om zodoende te voorkomen dat ladders op dezelfde plek staan.

Dit algoritme is wellicht wat ingewikkelder, maar geen code benodigd om de restrictie van de vier ladders weg te nemen. Hierdoor blijft er wellicht extra ruimte over om ook de andere schermen aan te passen.

Eerst met PowerShell het algoritme gaan vaststellen.

Map met PowerShell code op het bureaublad neergezet. Eerst een originele dkong.zip maken op basis van de eerdere default die goed start en geen DK climbing intro toont. Dat is de eerste versie van DK Duel die daarvoor gebruikt kan worden.

En dan daarin de middelste twee girders en de ladders in de middelste drie gebieden weghalen.

```
3AE4:  02 97 38 68 38    ; top girder where girl sits
3AE9:  02 9F 54 10 54    ; girder where kong sits
3AED:  02 DF 58 A0 55    ; 1st slanted girder at top right
3AF3:  02 EF 6D 20 79    ; 2nd slanted girder (has hammer at left side)
3AF8:  02 DF 9A 10 8E    ; 3rd slanted girder
3AFD:  02 EF AF 20 BB    ; 4th slanted girder
3B02:  02 DF DC 10 D0    ; 5th slanted girder (has hammer at right side)
3B07:  02 FF F0 80 F7    ; bottom slanted girder
3B0C:  02 7F F8 00 F8    ; bottom flat girder where mario starts
3B11:  00 CB 57 CB 6F    ; short ladder at top right
3B16:  00 CB 99 CB B1    ; short ladder at center right
3B1B:  00 CB DB CB F3    ; short ladder at bottom right
3B20:  00 63 18 63 54    ; kong's ladder (right)
3B25:  01 63 D5 63 F8    ; bottom broken ladder
3B2A:  00 33 78 33 90    ; short ladder at left side under top hammer
3B2F:  00 33 BA 33 D2    ; short ladder at left side above oil can
3B34:  00 53 18 53 54    ; kong's ladder (left)
3B39:  01 53 92 53 B8    ; second broken ladder from bottom, on 3rd girder
3B3E:  00 5B 76 5B 92    ; longer ladder under the top left hammer
3B43:  00 73 B6 73 D6    ; longer ladder to left of bottom hammer
3B48:  00 83 95 83 B5    ; center longer ladder
3B4D:  00 93 38 93 54    ; ladder leading to girl
3B52:  01 BB 70 BB 98    ; third broken ladder on right side near top
3B57:  01 6B 54 6B 75    ; fourth broken ladder near kong
3B5C:  AA                ; AA code signals end of data
```

Dat wordt dan:

```
3AE4:  02 97 38 68 38    ; top girder where girl sits
3AE9:  02 9F 54 10 54    ; girder where kong sits
3AEE:  02 DF 58 A0 55    ; 1st slanted girder at top right
3AF3:  02 EF 6D 20 79    ; 2nd slanted girder (has hammer at left side)
3AF8:  02 DF DC 10 D0    ; 5th slanted girder (has hammer at right side)
3AFD:  02 FF F0 80 F7    ; bottom slanted girder
3B02:  02 7F F8 00 F8    ; bottom flat girder where mario starts
3B07:  00 CB 57 CB 6F    ; short ladder at top right
3B0C:  00 CB DB CB F3    ; short ladder at bottom right
3B11:  00 63 18 63 54    ; kong's ladder (right)
3B16:  01 63 D5 63 F8    ; bottom broken ladder
3B1B:  00 53 18 53 54    ; kong's ladder (left)
3B20:  00 93 38 93 54    ; ladder leading to girl
3B25:  01 6B 54 6B 75    ; fourth broken ladder near kong
3B2A:  AA                ; AA code signals end of data
```

Eerst testen met het verschuiven van de girders in stappen van 2 en kijken wat de uiterste verschuivingen zijn (in combinatie) waarbij de barrels nog goed rollen en ook de roll-over goed gaat zonder dat de barrels de lager missen (en uit beeld verdwijnen). Eventueel ook testen of je dit nog kan uitbreiden door de lagere girder verder door te trekken.

Dat gaat om de volgende twee girders:

```
XXXX:  02 EF AF 20 BB    ; 4th slanted girder – onderste van de twee
XXXX:  02 DF 9A 10 8E    ; 3rd slanted girder – bovenste van de twee
```

Scenario's:

Allebei naar beneden verplaatsen: bovenste overgang testen.
Allebei naar boven verplaatsen: onderste overgang testen.
Bovenste naar boven en onderste naar beneden verplaatsen: middelste overgang testen.

Door de random plaatsing krijg je mogelijk al hele lange ladders. Dus wellicht niet te ver verschuiven.

Twee naar beneden verschoven. Gaat al meteen mis bij de roll-over: de barrels bovenaan missen de girder. Kijken door de girder verder naar links door te trekken. Gaat goed.

Daarna vier naar beneden verschoven. Gaat nog goed en niet te springen van girder op girder. Dit is de maximale verschuiving naar beneden.

Daarna twee naar boven verschoven. Gaat ook mis bij de roll-over: de barrels onderaan missen de girder. Kijken door de girder verder naar links door te trekken. Dit door aanpassen van de girder in de definitie (vaste girder):

```
3AF8:  02 DF DC 08 D0    ; 5th slanted girder (has hammer at right side)
```

Daarna vier naar boven verschoven. Gaat nog goed en niet te springen van girder op girder. Dit is de maximale verschuiving naar boven.

Daarna de extreme situatie testen: bovenste girder vier omhoog en de onderste girder vier naar beneden. Dat gaat gewoon goed met de roll-over.

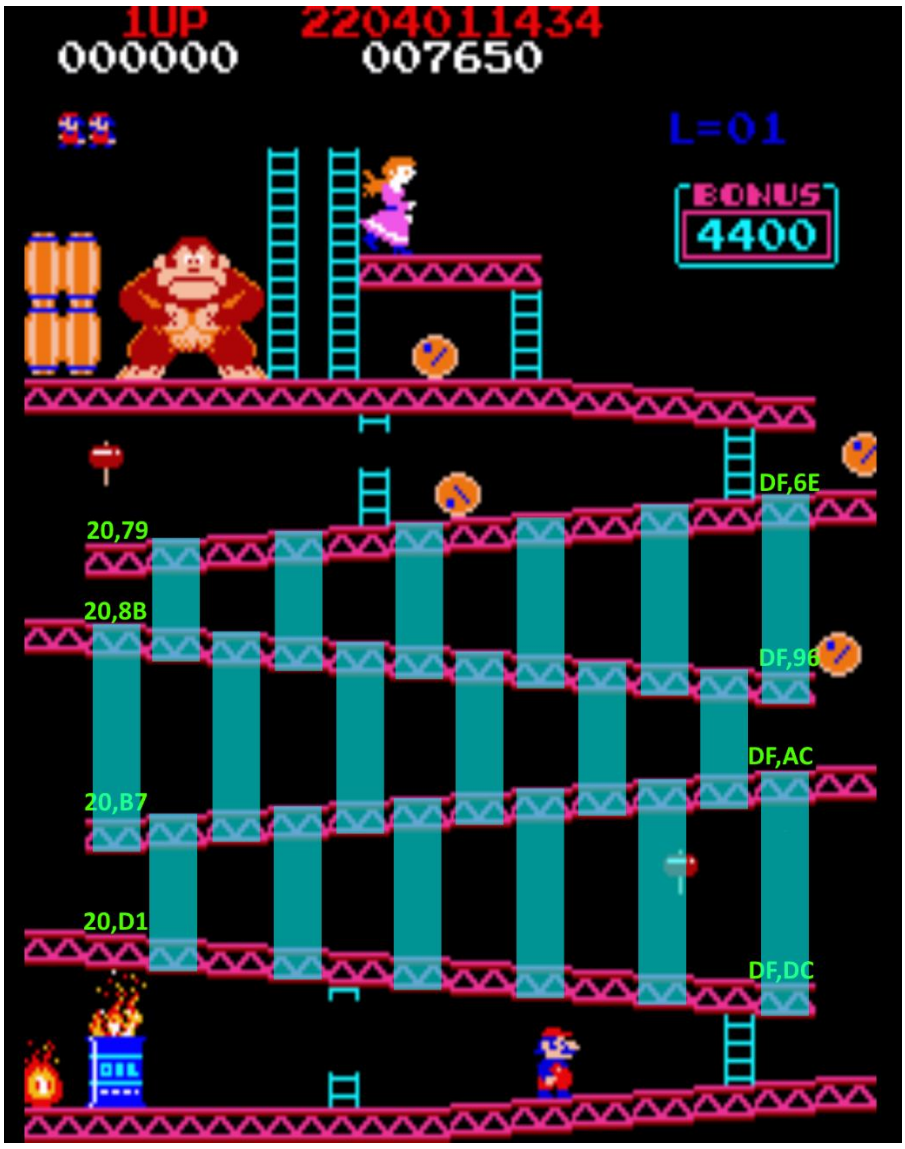
Daarna de andere extreme situatie testen: bovenste girder vier naar beneden en de onderste girder vier naar boven. Dat gaat gewoon goed met de roll-over. Alleen krijg je tussen de twee girders wel een vrij kleine afstand die, zo lijkt het, te springen is zonder ladder. Maar door tuning kunnen we later wel bepalen of dat een probleem is.

In principe is die kleine afstand er alleen wanneer de beide ladders in de uiterste posities staan. Met -4, -2, 0 +2 en +4 zijn er vijf verschillende plekken waar elke girder kan staan. Dat betekent dan dat slechts 1 van de 25 mogelijke combinaties dit probleem heeft. Wat wellicht zelfs interessant kan zijn.

Maar in de programmatuur veel handiger om een willekeurig getal tussen 0 en 3 te bepalen: dat zijn vier verschillende plekken. En dan offset slechts één kant op (+ = omlaag). En dan de beginwaarden bij beiden op de -4 t.o.v. de originele y-waarde zetten. En dan ontstaat er nooit de kleine afstand die gesprongen kan worden…

Dit bouwen in het PowerShell script met randomization van de twee offsets. En dan veel testen op het juiste rollen van de barrels en de juiste roll-over van de barrels. Dat gaat goed.

Dan kijken in de girderposities zonder offset waar de ladders getekend kunnen worden.



Wanneer bijvoorbeeld de bovenste van de twee girders 2 posities naar beneden dan verwisselen de plekken van de twee bovenste gebieden qua x-waarden.

Dus dat is op elk niveau een zestal ladders. Dus in plaats van de 12 verschillende posities, zijn het er nu maar 6. Dat is op zich geen probleem.

Nu dit vertalen naar een algoritme met een loop voor de drie gebieden, met dan random vaststellen van de drie plekken (twee vaste ladders en één gebroken en voor een gebied géén gebroken ladder) en dan bepalen waar deze getekend moeten worden: startwaarden, positie op basis van de twee offsets en aanpassen voor schuine girders.

Werken vanaf de rechterkant. De startwaarden voor de y zonder offset zijn:

6E – (#28/40) – 96 – (#16/22) – AC – (#30/48) – DC
vv van 4 geen vv van 4 vv van 4

Voor later in de code: als bit 0 en bit 1 beiden 0 zijn dan veelvoud van vier.
Als een veelvoud van vier dan op dat segment een ladder mogelijk: start X = #D3.
Als geen veelvoud van vier dan op het vorige segment een ladder mogelijk: start X = #C3 (offset -#10)

Ook nog aanpassen voor plaats op het segment: bij het bovenste en onderste gebied aan de linkerkant van het segment: #X3 en bij het middelste gebied aan de rechterkant van het segment:#XB (offset +#08).

De vaste ladders staan nog niet allemaal goed. De onderste gebroken ladder staat nu nog links op het segment, maar dat moet rechts zijn: aanpassen van de ladder in de definitie:

3B16: 01 6B D5 6B F8 ; bottom broken ladder

Dat is nu goed.

Nog iets te makkelijk. Misschien niet de broken ladder skippen, maar een normale ladder. Maar dan teveel broken ladders (maximum is 4). Dan de bovenste broken ladder (vlak bij Donkey Kong) normaal maken:

3B25: 00 6B 54 6B 75 ; fourth broken ladder near kong

Ja dat gaat al een stuk beter. Interessante velden.

Twee andere observaties die interessant zijn:

- Doordat de girders dichter bij elkaar zitten kom je vaker tegen barrel op de girder boven je aan.
- Als de bovenste girder maximaal naar beneden en de onderste maximaal naar boven dan kun je van girder naar girder springen.

Dit wellicht combineren met elementen van Barrelpalooza en dan dat als nieuwe versie van Barrelpalooza maken? Nee, wel barrel randomizer noemen, met optie om barrelpalooza elementen aan te zetten.

Idee is om wel met levels te werken. En dan steeds drie random barrel levels en dan een rivets level, maar die dan met de rode schuine girders. Dan is er gewoon een L=22 met kill screen. Dat men vaak graag wi hebben.

Bijvoorbeeld:

- Random ladders altijd.
- Andere opties standaard aan, maar uit te zetten via opties met defaults via dipswitches:
    - Random girder spacing (default on)
    - Random hammers placement (default on)
    - Random hole placement (default on)
    - Random slow barrels and super slow barrels (default off)
    - Random blinking hammer (default off)
    - Random red wild barrel (default off)
    - Random falling spring (default off)
    - Random darkness periods (default off)
        - Slow barrels
        - Blinking hammers
        - Falling spring
        - Random darkness periods
        - Rode wild barrel
- Randomizations van board opbouw (ladders, girders, hammers and holes) op basis van een seed, default random seed, via menu seed in te stellen.
- Overige randomizations net als in de eerdere rom-hacks (On The Run, Barrelpalooza)
- Weergave miljoenenscore
- Weergave sublevel
- Vastleggen score + opties + seed + sublevel in High score table

Eerst nog eens kijken naar de PowerShell en kijken of de girders nog verder gevarieerd kunnen worden.

De twee girders krijgen nu een random offset van (0, 2, 4 en 6). En dat is dan gerekend vanaf de default hoogte naar beneden toe. Als je verdere uitslag wilt, dan misschien ook offset van 8 toevoegen: bovenste girder zelfde start-Y, onderste girder start-Y met 2 verlagen. Ja dat werkt.

En zelfs nog verder offset van 10 toevoegen. Dat gaat niet goed.

Dus houden bij een maximale offset van 8. Gaat goed. Alleen broken ladders worden wel heel raar getekend als de bovenste maximale offset en onderste minimale offset (heel dicht bij elkaar). Je kan dan ook van girder naar girder springen. Dan wellicht geen broken ladder aanmaken? Toegevoegd. Dat gaat goed.

Eerst nu de volgende zaken goedzetten:

- Tekenen van de gebroken ladder altijd goed doen: nu soms te zien als hele ladder.
- Met een variabele (geheugenplek straks) mogelijkheid om random girders te disablen.
- Randomization van de ladders als test in de PowerShell.

Daarna:

- Omzetten PowerShell naar pseudo assembly met als voorbeeld de Wizardry omzetting.

- Aanmaken random generator op basis van code met seed en testen.
- Overzetten code random girders naar rom.
- Overzeten code random ladders naar rom.
- Uitgebreid testen.
- Verder uitwerken randomizations en optiemenu met defaults op basis van dipswitches.
- Verder uitwerken volgorde 3 x barrels en 1x rivets.
- Uitwerken rivets met schuine girders en eventueel randomizations (afstand en ladders)?

Tekenen van een gebroken ladder

Als gebroken ladder te kort, dan soms dat onderste en bovenste deel elkaar raken c.q. overlappen waardoor het een normale ladder lijkt. Dit is afhankelijk van de plek waar de gebroken ladder getekend wordt, de schuinte en de schermposities.

Kijken of bij het tekenenen van een korte gebroken ladder alleen de bovenkant getekend kan worden. Dit is al eerder gedaan in DK On The Run en staat in het betreffende ontwerpdocument beschreven op pagina 53 t/m 55. En het is ook eerder gedaan in DK Barrelpalooza en staat in het betreffende ontwerpdocument beschreven op pagina 49 t/m 51.

In beide gevallen zitten er voorwaarden aan qua X-positie en/of screennummer. Dat hoeft nu niet, want dit geldt altijd. Nu wel specifiek ophangen aan de lengte van de ladder.

De onderkant van de gebroken ladder wordt hier getekend:

```
; this is a broken ladder.  draw bottom part of ladder

0E3B  2D        DEC     L               ; decrease HL
0E3C  36C0      LD      (HL),#C0        ; set HL to #C0 - draws bottom part of broken ladder to screen
0E3E  2C        INC     L               ; increase HL

0E3F  3AB063    LD      A,(#63B0)       ; load A with ???
0E42  FE00      CP      #00             ; == 0 ?
0E44  CA4B0E    JP      Z,#0E4B         ; yes, skip next 3 steps

0E47  C6E0      ADD     A,#E0           ; add #E0
0E49  2C        INC     L               ; next HL
0E4A  77        LD      (HL),A          ; store into ???

0E4B  13        INC     DE              ; next table entry
0E4C  C3A70D    JP      #0DA7           ; loop again
```

Dus #0E3B vervangen door sprong naar additionele code. Daar dan checken hoe lang de ladder is en als deze korter is dan een ingestelde treshold, dan de onderkant van de ladder NIET tekenen (terugspringen naar #0E3F) en anders de onderkant van de ladder WEL tekenen (oorspronkelijke code van #0E3B t/m #0E3E uitvoeren en daarna terugspringen naar #0E3F).

Lengte van de ladder bepalen door Y-coördinaat bovenkant af te trekken van de Y-coördinaat aan de onderkant. Op het moment dat je bij #0E3B aankomt staat de DE te wijzen naar de onderste Y-coördinaat. De bovenste Y-coördinaat staat daar twee plekken voor.

Aanpassen:

```
0E3B    C3383F      JP      #3F38       ; jump to additional code – jump to label AAAA
0E3E    00          NOP
```

en:

```
3F38    1A          LD      A,(DE)      ; load A with Y-position bottom – AAAA
3F39    47          LD      B,A         ; save in B
;
3F3A    D5          PUSH    DE          ; store DE
3F3B    1B          DEC     DE          ; decrement DE, DE points to x-position bottom
3F3C    1B          DEC     DE          ; decrement DE, DE points to y-position top
3F3D    1A          LD      A,(DE)      ; load A with Y-position top
3F3E    D1          POP     DE          ; restore DE
;
3F3F    4F          LD      C,A         ; save A in C – C contains y-position top
3F40    78          LD      A,B         ; restore A from B – A contains y-position bottom
3F41    91          SUB     C           ; subtract top from bottom
3F42    FE20        CP      #20         ; Is this a short broken ladder?
3F44    DA3F0E      JP      C,#0E3F     ; yes, jump back without drawing bottom
;
3F47    2D          DEC     L           ; decrease HL
3F48    36C0        LD      (HL),#C)    ; set HL to #C0 – draws bottom
3F4A    2C          INC     L           ; increase HL
3F4B    C33F0E      JP      #0E3F       ; jump back after drawing bottom
```

Gaat redelijk goed, maar had toch nog een ladder met afstand van #14 (dus korter dan #20) die als geheel getekend werd (met time stamp 2204211620). Deze handmatig gaan aanpassen met verschillende tresholds en kijken hoe dit goed gekregen kan worden?

Kijken of in het geval van een korte ladder de erboven getekende ladder gewijzigd kan worden in een kortere.

Dat geldt dan voor de girder-met-ladders graphics van #C1, #C2 en #C3. Hiervoor aangepaste graphics gemaakt in respectievelijk #CA, #CB en #CC.

Aanpassen code:

```
3F42    FE20        CP      #20         ; Is this a short broken ladder?
3F44    3013        JR      NC,#BBBB    ; no, jump to label BBBB
;
```

```
3F46    2D          DEC     L           ; decrease HL
3F47    2D          DEC     L           ; decrease HL
3F48    7E          LD      A,(HL)      ; load A with previously drawn graphic
;
3F49    FEC1        CP      #C1         ; is element smaller than #C1?
3F4B    3807        JR      C,#CCCC     ; yes, jump to label CCCC
3F4D    FEC4        CP      #C4         ; is element greater than #C3?
3F4F    3003        JR      NC,#CCCC    ; yes, jump to label CCCC
;
3F51    C609        ADD     A,#09       ; select changed graphic
3F53    77          LD      (HL),A      ; change graphic
;
3F54    2C          INC     L           ; increase HL - CCCC
3F55    2C          INC     L           ; increase HL
;
3F56    C33F0E      JP      #0E3F       ; yes, jump back without drawing bottom
;
3F59    2D          DEC     L           ; decrease HL - BBBB
3F5A    36C0        LD      (HL),#C0    ; set HL to #C0 – draws bottom
3F5C    2C          INC     L           ; increase HL
3F5D    C33F0E      JP      #0E3F       ; jump back after drawing bottom
```

In een bepaalde situatie wordt er nog steeds een volledige trap getekend. Uit debuggen blijkt dat dit niet de extra code raakt. Dus hoe kan dat, is er ergens een check op veelvoud van iets en als je dan precies op de grens zit, dat er dan niet naar deze onderste trapdeel tekenen gesprongen wordt?

De code die nu aangepast is, gaat uit van het tekenen van een geheel ladder graphic boven het girderdeel als, dan niet met stukje ladder. Maar als de ladder echt te kort is, dan kan dat helemaal niet en wordt dat ook niet gedaan. En dan sluiten ze op elkaar aan. In dat geval de onderste of de bovenste tekenen zonder trap. Maar waar zit dat dan?

In principe kan je dan dus terug naar de vorige versie:

```
3F42    FE20        CP      #20         ; Is this a short broken ladder?
3F44    DA3F0E      JP      C,#0E3F     ; yes, jump back without drawing bottom
;
3F47    2D          DEC     L           ; decrease HL
3F48    36C0        LD      (HL),#C)    ; set HL to #C0 – draws bottom
3F4A    2C          INC     L           ; increase HL
3F4B    C33F0E      JP      #0E3F       ; jump back after drawing bottom
```

En dan bij het tekenen van de onderste girder stuk checken of het een korte gebroken ladder is en dan een stuk tekenen zonder ladder. Eerst met debugger dat uitzoeken. Gevonden. De instructie op #0E0C tekent het bovenste stukje ladder. De instructie op #0E32 tekent het onderste stukje ladder.

```
0E2A  3AB063    LD      A,(#63B0)       ; load A with ???
0E2D  C6D0      ADD     A,#D0           ; add #D0
0E2F  2AAD63    LD      HL,(#63AD)      ;
0E32  77        LD      (HL),A
0E33  3AB363    LD      A,(#63B3)       ; load A with original data item
0E36  FE01      CP      #01             ; == 1 ?  (is this a broken ladder ?)
0E38  C23F0E    JP      NZ,#0E3F        ; no, skip next 3 steps
```

Dus de instructie op #0E32 niet doen als het een kleine ladder is.

Aanpassen:

```
0E2F  C34E3F      JP    #3F4E       ; jump to additional code – jump to label AAAA
```

en:

```
3F4E  F5          PUSH  AF          ; save AF - AAAA
3F4F  2AAD63      LD    HL,(#63AD)  ; original instruction replaced by JP
3F52  3AB363      LD    A,(#63B3)   ; load A with original data item
3F55  FE01        CP    #01         ; == 1 ?  (is this a broken ladder ?)
3F57  C26D3F      JP    NZ,#3F6D    ; no, draw bottom of ladder – jump to label BBBB
;
3F5A  1A          LD    A,(DE)      ; load A with Y-position bottom
3F5B  47          LD    B,A         ; save in B
;
3F5C  D5          PUSH  DE          ; store DE
3F5D  1B          DEC   DE          ; decrement DE, DE points to x-position bottom
3F5E  1B          DEC   DE          ; decrement DE, DE points to y-position top
3F5F  1A          LD    A,(DE)      ; load A with Y-position top
3F60  D1          POP   DE          ; restore DE
;
3F61  4F          LD    C,A         ; save A in C – C contains y-position top
3F62  78          LD    A,B         ; restore A from B – A contains y-position bottom
3F63  91          SUB   C           ; subtract top from bottom
3F64  FE14        CP    #14         ; Is this a short broken ladder?
3F66  D26D3F      JP    NC,#3F6D    ; no, draw bottom of ladder – jump to label BBBB
;
3F69  F1          POP   A           ; restore AF
3F6A  C3330E      JP    #0E33       ; jump back without drawing bottom of ladder
;
3F6D  F1          POP   A           ; restore AF - BBBB
3F6E  C3320E      JP    #0E32       ; no, jump back and draw bottom of ladder
```

Ja dat lijkt goed te gaan. Nu overzetten naar de template rom.

En flink testen.

Wat wel nog blijkt is dat jumpman door blijft klimmen op de ladder vlak bij Donkey Kong (de bovenste broken ladder die normal gemaakt is). Dat nog aanpassen door die te verplaatsen?

Nu versie v0.02 opgeslagen.

Blijven nadenken over de vorm van de uiteindelijke rom-hack. Neemt wel wat ruimte in, dus meest logische om alleen barrel randomization doen. Dus dan een soort Barrelpalooza. Best eenvoudig in principe, en uitnodigend om board running te doen. Dat voorkomen door extra moeilijkheden in te bouwen. Best leuk als je al die zaken tegelijkertijd kan hebben en dan wordt het best uitdagend.

Random plaatsing van de hammers

De plaatsing van de hamers ook random maken. Vier ruimtes tussen de girders waar ze geplaatst kunnen worden. Eerst voor beiden bepalen op welke hoogte ze komen (random tussen 1 en 4) waarbij niet dezelfde gekozen mag worden. Daarna de plaats (links/rechts) bepalen. Over de gehele lengte van de girder of alleen in het hoge deel (dat laatste heeft de voorkeur)? Op basis daarvan de X-positie vaststellen. En dan daarna op basis van de verschuiving van de girders en het schuine van de girders de Y-positie vaststellen.

In de PowerShell dit regelen door de data voor de hammers in de rom aan te passen:

```
3E08:  1E 07          ; 1E is the hammer sprite value.  07 is hammer color
3E0A:  03 09          ; ???
3E0C:  24 64          ; position of top hammer for girders.  24 is X, 64 is Y
3E0E:  BB C0          ; bottom hammer for girders at BB, C0
```

Dus invoegen op #3E0C/D en #3E0E/F. Data structuurtje voorbereiden en die dan kopiëren.

Hier verder mee stoppen.

Gaan voor de DK Randomizer versie. Dat gaat het meeste aanspreken. En dan goed kijken welke board randomizations er nog meer kunnen voor de andere boards.

Daarom de ladder onder Donkey Kong weer broken maken, geen girder offsets meer, en de girders weer standaard tekenen.

Ladder weer broken maken:

3B25: 01 6B 54 6B 75   ; fourth broken ladder near kong

En de girders weer opnemen in de template rom.

3B2A: 02 EF AF 20 BB   ; 4th slanted girder – onderste van de twee
3B2F: 02 DF 9A 10 8E   ; 3rd slanted girder – bovenste van de twee

Aangepast en de PowerShell opgeschoond. Hiermee eventjes goed testen en dan kijken of dit script (wat heel erg lijkt op het script voor wizardry, gaan uitwerken naar assembly code). Dat dan opnemen en daarna andere randomizations toevoegen.

Nu v0.03 opgeslagen.

Hier niet meer mee verdergaan.

Twijfels over het volgende:

- Random barrels is vrij eenvoudig.
- Tweede barrelpalooza zal geen hit worden.
- Randomizer voor de pies en elevators is niet zo spannend.
- In ieder geval al eens gedaan in Wizardy.
- Zonde van mijn tijd als het geen succes wordt.

Toch weer hiernaar kijken. Kijken of het mogelijk is om vier verschillende interessante barrel stages te maken en die dan elk te voorzien van randomized ladders. Dan ruimte vrij doordat elevators, springs, pies, rivets allemaal niet meer nodig zijn. Maar wel de nodige variatie. Je kan dan een JP schema aanhouden en level laten oplopen met KS bij L=22! En de two-player mode maken net als DUEL met dezelfde random boards per speler. Daarnaast de optie om een aantal mechanismen aan of uit te zetten: random placed hammers, blinking hammers, slow and super slow barrels, more wild barrels, into the dark periods, first wild red barrel, random flying birds. En de optie om met een ingestelde seed te beginnen.

Vier barrel stages, mogelijke opties:

- Standaard girders (6 random ladder positions)
- Shifted girders (5 random ladder positions)?
- Remix achtige girders (3 random ladder positions aan elke kant)
- Gelaagde dakjes girders (aantal random ladders te bepalen)
- ~~Drie evenwijdige girders (12 random ladder positions)? – Nee, barrels altijd omlaag !~~

Van elk van de opties eerst met PowerShell bouwen en testen. Daarbij soms ook logica aanpassen, zoals de rollover en de rolrichting. In ieder geval het tekenen van de korte gebroken ladders zit er nu al in. Daarna alles gaan omzetten naar pseudo assembly en dan gaan toevoegen aan de rom.

Shifted girders

De shifted girders uit Barrelpalooza nemen. Nee dat gaat niet goed door verkeerd rollende barrels.

Daarom gebruik maken van een statische versie van de extreme versie waarbij de onderste girder maximaal omhoog geconfigureerd en de bovenste maximaal omlaag geconfigureerd is.

Even die draaien met standaard waarden en dan kijken welke waarden daarbij gebruikt worden.

Uit de aangemaakte rom-file:

02 DF 9E 08 92
02 EF A9 20 B5

De aanpassing aan de girders:

3B2A: 02 EF A9 20 B5 ; 4th slanted girder – onderste van de twee
3B2F: 02 DF 9E 08 92 ; 3rd slanted girder – bovenste van de twee

En de traplengtes overeenkomstig aanpassen in de PowerShell:

```
$ladder_ypos = @(0x6E, 0x9E, 0xAA, 0xDC)
```

Dit is een leuke variatie als één van de vier barrel boards.

Zowel langere als kortere trappen. En door de korte afstand tussen de twee middelste girders zorgt ervoor dat je van girder naar girder kan springen en dat je moet oppassen bij springen dat je je hoofd niet stoot tegen een erboven rollende barrel.

Gaat nog niet goed. Onderin verdwijnen de barrels aan de linkerkant. Roll over gaat niet goed. Wellicht omdat de girder door moet lopen om de hoge val te compenseren.

Origineel:

3B02: 02 DF DC 10 D0 ; 5th slanted girder (has hammer at right side)

Zit nu op #3FA8 als het goed is.

3AF8: 02 DF DC 08 D0 ; 5th slanted girder (has hammer at right side)

Ja dat gaat nu wel goed.

Nu v0.04 opgeslagen.

Hiermee stoppen. Voor de community een Randomizer met de vier standaard boards in een randomized setup lijkt het meest succesvol te worden. Dus de huidige PowerShell om gaan zetten naar code. Daartoe de stappen uit DK Wizardry documentatie volgen. Als dat erin zit, kijken naar de mogelijkheden voor de andere velden om randomization te doen en die veld voor veld ontwikkelen.

Steeds heen en weer. Eigenlijk maar één oplossing mogelijk. Gewoon de huidige randomization van PowerShell gaan omzetten naar de rom en dan kijken of het werkt.

Eerst code maken waarmee een random getal bepaald wordt op basis van de code. En dan aanpassen naar een getal tussen 0 en 5.

H en L opslaan dus daar zijn twee geheugenplaatsen nodig. Daarvoor dan reserveren:

Memory: #6232 - Opslaan H van de pointer into code
Memory: #6233 - Opslaan L van de pointer into code

```
0A7B  3A3262    LD    A,(#6232)    ; get stored value for H – label BBBB
0A7E  67        LD    H,A          ; store in H
```

```
0A7F    3A3362          LD      A,(#6233)       ; get stored value for L
0A82    3C              INC     A               ; store in L – INC A (3C)
0A83    6F              LD      L,A             ; increase L – LD L,A (6F)
0A84    2008            JR      NZ              ; no carry, skip next steps – jump to label CCCC
;
0A86    24              INC     H               ; increase H
0A87    7C              LD      A,H             ; load A with H
0A88    FE40            CP      #40             ; H==#40 ?
0A8A    2002            JR      NZ              ; no, skip next steps – jump to label CCCC
;
0A8C    2601            LD      H,#01           ; reset H
;
0A8E    7E              LD      A,(HL)          ; load opcode from HL – label CCCC
0A8F    47              LD      B,A             ; store A in B
0A90    7C              LD      A,H
0A91    323262          LD      (#6232),A       ; save H
0A94    7D              LD      A,L
0A95    323362          LD      (#6233),A       ; save L
0A98    78              LD      A,B             ; restore A from B
;
0A99    C9              RET                     ; return opcode as random number in A
```

En dan code in de main steeds opnieuw aanroepen, zonder er iets mee te doen en dan met de debugger de getallen bekijken (welke opcode en welk getal onder de twaalf). Daarna hele reeks noteren en kijken naar verdeling en kijken of de rollover bij het einde goed gaat. Daarna ook nog kijken naar het initieel vullen op basis van een seed. En hoe wordt daarmee omgegaan in het geval van een nieuwe game. Dus op welk moment de seed vaststellen en geen onbedoelde reset ergens.

Code om het random getal te laten zien invoegen in de main. Lege plek voor call op #19C2.

```
19C2    CD7B0A          CALL    #0A7B           ; call test rng code – call to label XXXX
```

Dat gaat goed.

Met de debugger met watchpoint op #0A99 is te zien dat er de code doorlopen wordt. De HL wordt netjes opgehoogd (startend vanaf #0001) en elke keer dezelfde reeks van ‘random’ waarden gelezen. En die waarden komen inderdaad overeen met de waarden van de code.

Nu zodanig maken dat de teruggegeven waarde in A tussen de 0 en 5 ligt.

Bepalen op bitgroepen uit de waarde:

Bit 0 – waarde tussen 0 en 1
Bit 1 – waarde tussen 0 en 1
Bit 2,3 – waarde tussen 0 en 3
Deze drie bij elkaar optellen geeft waarde tussen 0 en 5 (1+1+3).

```
0A99    E601            AND     #01             ; keep bit 0 – value between 0 and 1
```

```
0A9B    4F          LD      C,A         ; save to C
;
0A9C    78          LD      A,B         ; restore original opcode value from B
0A9D    1F          RRA                 ; rotate right
0A9E    E601        AND     #01         ; keep bit 0 – value between 0 and 1
0AA0    81          ADD     A,C         ; add C
0AA1    4F          LD      C,A         ; save to C
;
0AA2    78          LD      A,B         ; restore original opcode value from B again
0AA3    1F          RRA                 ; rotate right
0AA4    E603        AND     #03         ; keep bits 0 and 1 – value between 0 and 3
0AA6    81          ADD     A,C         ; add C
;
0AA7    C9          RET                 ; return
```

Watchpoint zetten op #0AA7 en checken.

In A staan dan bij de return alleen maar waarden tussen de 0 en 5. Dus dat gaat goed.

Nu versie v0.05 opgeslagen.

Nu kijken om de routine te maken die de drie geheugenplaatsen voor de unieke ladder posities te vullen

Memory 1: #66A0 - loop counter for ladder level (outer loop)
Memory 2: #66A1 - loop counter for ladders (inner loop)
Memory 3: #66A2 - ladder level on which to skip broken ladder
Memory 4: #66A3 - ladder level Y-value
Memory 5: #66A4 - 1st ladder offset
Memory 6: #66A5 - 2nd ladder offset
Memory 7: #66A6 - 3rd ladder offset

Dus #66A4, #66A5 en #66A6 gebruiken.

Globaal:
Vul A met random waarde -> schrijf naar #66A4
Vul A met random waarde -> schrijf naar #66A5
Vul A met random waarde -> schrijf naar #66A6
Check #66A4 met #66A5
Check #66A4 met #66A6
Check #66A5 met #66A6

Anders drie verschillende waarden gevuld.

```
0AA8    DD21A466    LD      IX,#66A4    ; set IX to index #66A4 – label AAAA
0AAC    CD7B0A      CALL    #0A7B       ; get random number 0-11
0AAF    DD7700      LD      (IX+#00),A  ; store in #66A4
0AB2    CD7B0A      CALL    #0A7B       ; get random number 0-11
```

```
0AB5    DD7701          LD      (IX+#01),A      ; store in #66A5
0AB8    CD7B0A          CALL    #0A7B           ; get random number 0-11
0ABB    DD7702          LD      (IX+#02),A      ; store in #66A6
;
0ABE    DD7E00          LD      A,(IX+#00)      ; get #66A4
0AC1    DD9601          SUB     (IX+#01)        ; subtract #66A5
0AC4    28E2            JR      Z,#0AA8         ; if equal do again – jump to label AAAA
;
0AC6    DD7E00          LD      A,(IX+#00)      ; get #66A4
0AC9    DD9602          SUB     (IX+#02)        ; subtract #66A6
0ACC    28DA            JR      Z,#0AA8         ; if equal do again – jump to label AAAA
;
0ACE    DD7E01          LD      A,(IX+#01)      ; get #66A5
0AD1    DD9602          SUB     (IX+#02)        ; subtract #66A6
0AD4    28D2            JR      Z,#0AA8         ; if equal do again – jump to label AAAA
;
0AD6    C9              RET

en:

19C2    CDA80A          CALL    #0AA8           ; call test rng code – call to label XXXX
```

Berekenen jumpafstanden JR Z operaties (inclusief beide opcodes)
30 – 1E inverse = E2
38 – 26 inverse = DA
46 – 2E inverse = D2

Lijkt goed te gaan. Zorgt er in ieder geval voor dat er altijd drie unieke waarden gekozen worden.
Met debugger gecheckt en dat gaat goed.

Nu versie v0.06 opgeslagen.

Nu de PowerShell code omzetten. Daartoe de code van DK Wizardy pakken en aanpassen. Eerst even heel goed bekijken wat er anders is in deze versie en dan de code gaan verbouwen…

Memory 1: #66A0 - loop counter for ladder level (outer loop)
Memory 2: #66A1 - loop counter for ladders (inner loop)
Memory 3: #66A2 - ladder level on which to skip broken ladder
Memory 4: #66A3 - ladder level Y-value
Memory 5: #66A4 - 1st ladder offset
Memory 6: #66A5 - 2nd ladder offset
Memory 7: #66A6 - 3rd ladder offset
Memory 8: #66A7 - ladder_x temporary storage
Memory 9: #66A8 - ladder_delta temporary storage

De ladder definitie opbouwen in #6B00 en verder:
9*5 (ladders) = 45 + 1 (afsluitcode) = 46 = #2E + #66B0 = #66DE.
HL register is de pointer naar de plek in de datastructuur.

IX register is de pointer naar de ladder offsets
IY register is de pointer naar de ladder y-positions start values

```
0AD7   6E 9A B0 DC                        ; data table: ladder y-positions start
0ADB   FD21D70A      LD     IY,#0AD7      ; load IY with start of data table
;
0ADF   3A1860        LD     A,RngTimer1   ; get random number for broken ladder skip
0AE2   E603          AND    #03           ; random number between 0 and 3
0AE4   FE03          CP     #03           ; A == 3 ?
0AE6   2002          JR     NZ,#AAAA      ; no, skip next step
0AE8   3E01          LD     A,#01         ; A = 1
0AEA   C601          ADD    A,#01         ; random number between 1 and 3
0AEC   32A266        LD     (#66A2),A     ; store in brokenladderskip
;
0AEF   21006B        LD     HL,#6B00      ; load HL with start of datastructure
;
; Outer loop
;
0AF2   3E03          LD     A,#03         ; repeat 3 times – values 3, 2 , 1
0AF4   32A066        LD     (#66A0),A     ; store in outerloopcounter
;
0AF7   E5            PUSH   HL            ; save HL – label AAAA
0AF8   DDE5          PUSH   IX            ; save IX
;
0AFA   CDA80A        CALL   #0AA8         ; calculate 3 different random values and store
;
0AFD   DDE1          POP    IX            ; restore IX
0AFF   E1            POP    HL            ; restore HL
;
; Inner loop
;
0B00   3E03          LD     A,#03         ; repeat 3 times – values 3, 2 , 1
0B02   32A166        LD     (#66A1),A     ; store in innerloopcounter
;
0B05   47            LD     B,A           ; store innerloopcounter in B – label BBBB
;
0B06   DD21A466      LD     IX,#66A4      ; load IX with 1stladderoffset
0B0A   FE01          CP     #01           ; innerloopcounter = 01?
0B0C   2808          JR     Z,#CCCC       ; yes, skip next steps – jump to label CCCC
;
0B0E   DD23          INC    IX            ; load IX with 2ndladderoffset
0B10   FE02          CP     #02           ; innerloopcounter = 02?
0B12   2802          JR     Z,#CCCC       ; yes, skip next steps – jump to label CCCC
;
0B14   DD23          INC    IX            ; load IX with 3rdladderoffset
0B16   DD7E00        LD     A,(IX+#00)    ; load A with correct ladderoffset – label CCCC
0B19   57            LD     D,A           ; store correct ladderoffset in D
;
```

```
0B1A   78       LD    A,B        ; restore innerloopcounter from B
0B1B   FE01     CP    #01        ; innerloopcounter = 01 (last ladder)?
0B1D   3AA066   LD    A,(#66A0)  ; load A with outerloopcounter
0B20   47       LD    B,A        ; store outerloopcounter in B
;
0B21   200B     JR    NZ,DDDD    ; no skip next steps – jump to label DDDD
;
0B23   3AA266   LD    A,(#66A2)  ; load A with brokenladderskip
0B26   90       SUB   B          ; outerloopcounter = brokenladderskip?
0B27   CA9C0B   JP    Z,#QQQQ    ; yes, skip this broken ladder – jump to label QQQQ
;
0B2A   3E01     LD    A,#01      ; load A with 01 – laddertype broken
0B2C   1802     JR    #EEEE      ; skip next step – jump to label EEEE
;
0B2E   3E00     LD    A,#00      ; load A with 00 – laddertype normal – label DDDD
;
0B30   77       LD    (HL),A     ; store laddertype in data structure – label EEEE
0B31   23       INC   HL         ; set HL to next element
;
0B32   7A       LD    A,D        ; restore ladderoffset from D
0B33   CB27     SLA   A          ; shift left 5 times – ladderoffset multiply #20
0B35   CB27     SLA   A          ; shift left
0B37   CB27     SLA   A          ; shift left
0B39   CB27     SLA   A          ; shift left
0B3B   CB27     SLA   A          ; shift left
0B3D   4F       LD    C,A        ; save multiplied ladderoffset in C
;
0B3E   3ED3     LD    A,#D3      ; load A with start value x-position
0B40   91       SUB   C          ; subtract the multiplied ladderoffset
0B41   32A766   LD    (#66A7),A  ; store ladder_x in memory
;
0B44   7A       LD    A,D        ; restore ladderoffset from D
0B45   CB27     SLA   A          ; shift left – ladderoffset multiply #2
0B47   32A866   LD    (#66A8),A  ; store ladder_delta in memory
;
0B4A   FD7E00   LD    A,(IY+#00) ; load A with start value y-pos
0B4D   4F       LD    C,A        ; store in C
0B4E   FD7E01   LD    A,(IY+#01) ; load A with next start value y-pos
0B51   91       SUB   C          ; subtract, A = difference Y positions
;
0B52   E603     AND   #03        ; is A multiple of 4?
0B54   280F     JR    Z,#FF      ; yes, skip next steps – jump to label FFFF
;
0B56   3AA766   LD    A,(#66A7)  ; load ladder_x from memory
0B59   D610     SUB   #10        ; subtract offset – one segment left
0B5B   32A766   LD    (#66A7),A  ; store ladder_x in memory
;
0B5E   3AA866   LD    A,(#66A8)  ; load ladder_delta from memory
```

```
0B61    3C          INC     A           ; add offset – one segment left
0B62    32A866      LD      (#66A8), A  ; store ladder_delta in memory
;
0B65    3AA766      LD      A,(#66A7)   ; load ladder_x from memory – label FFFF
0B68    57          LD      D,A         ; store ladder_x to D
0B69    3AA866      LD      A,(#66A8)   ; load ladder_delta from memory
0B6C    4F          LD      C,A         ; store ladder_delta to C
;
0B6D    78          LD      A,B         ; restore outerloopcounter from B
0B6E    FE02        CP      #02         ; is outerloopcounter 2?
0B70    2018        JR      NZ,GGGG     ; no, skip next steps – jump to label GGGG
;
0B72    7A          LD      A,D         ; restore ladder_x from D
0B73    C608        ADD     A,#08       ; add offset to right on segment
0B75    57          LD      D,A         ; store adjusted ladder_x to D
;
0B76    77          LD      (HL),A      ; store xposition in data structure
0B77    23          INC     HL          ; set HL to next element
;
0B78    FD7E00      LD      A,(IY+00)   ;  load A with start value y-pos
0B7B    91          SUB     C           ; subtract ladder_delta stored in C
0B7C    77          LD      (HL),A      ; store ypositiontop in datastructure
0B7D    23          INC     HL          ; set HL to next element
;
0B7E    7A          LD      A,D         ;  restore ladder_x from D
0B7F    77          LD      (HL),A      ; store xposition in data structure
0B80    23          INC     HL          ; set HL to next element
;
0B81    FD7E01      LD      A,(IY+01)   ;  load A with start value y-pos
0B84    81          ADD     A,C         ; add ladder_delta stored in C
0B85    77          LD      (HL),A      ; store ypositiontop in datastructure
0B86    23          INC     HL          ; set HL to next element
;
0B87    C39C0B      JP      #QQQQ       ; jump to label QQQQ
;
0B8A    7A          LD      A,D         ; restore ladder_x from D – label GGGG
0B8B    77          LD      (HL),A      ; store xposition in data structure
0B8C    23          INC     HL          ; set HL to next element
;
0B8D    FD7E00      LD      A,(IY+00)   ;  load A with start value y-pos
0B90    81          ADD     A,C         ; add ladder_delta stored in C
0B91    77          LD      (HL),A      ; store ypositiontop in datastructure
0B92    23          INC     HL          ; set HL to next element
;
0B93    7A          LD      A,D         ;  restore ladder_x from D
0B94    77          LD      (HL),A      ; store xposition in data structure
0B95    23          INC     HL          ; set HL to next element
;
```

```
0B96    FD7E01      LD      A,(IY+01)       ;  load A with start value y-pos
0B99    91          SUB     C               ; subtract ladder_delta stored in C
0B9A    77          LD      (HL),A          ; store ypositiontop in datastructure
0B9B    23          INC     HL              ; set HL to next element
;
0B9C    3AA166      LD      A,(#66A1)       ; load A with innerloopcounter – label QQQQ
0B9F    D601        SUB     #01             ; decrease innerloopcounter
0BA1    32A166      LD      (#66A1), A      ; store in innerloopcounter
0BA4    C2050B      JP      NZ,#0B05        ; repeat if not zero – jump to label BBBB
;
0BA7    FD23        INC     IY              ; increase pointer into data table
;
0BA9    3AA066      LD      A,(#66A0)       ; load A with outerloopcounter
0BAC    D601        SUB     #01             ; decrease outerloopcounter
0BAE    32A066      LD      (#66A0), A      ; store in outerloopcounter
0BB1    C2F70A      JP      NZ,#0AF7        ; repeat if not zero – jump to label AAAA
;
0BB4    3EAA        LD      A,#AA           ; load A with #AA
0BB6    77          LD      (HL),A          ; store endmark in data structure
;
0BB7    C9          RET                     ; return
```

En aanroep hiernaartoe invoegen na het tekenen van het normale barrels board.

Eerst testen met dummy aanroep zodat via de debugger gekeken kan worden of de code inderdaad via dubbele loop doorlopen wordt, of de waarden goed gevuld worden en de data structuur opgebouwd wordt.

```
19C2    000000      NOP,NOP,NOP     ; three original NOP's
```

De schermen worden hier getekend:

```
; other screens return here

0CC6  CDA70D    CALL    #0DA7           ; draw the screen
```

Aanpassen:

```
0CC6    CDC70B      CALL    #0BC7           ; draw the screen
```

en:

```
0BC7    CDA70D      CALL    #0DA7           ; draw the screen
;
0BCA    3A2762      LD      A,(#6227)       ; load A with screen number
0BCD    FE01        CP      #01             ; is this the girders?
0BCF    C0          RET     NZ              ; no, return
```

```
;
0BD0    CDDB0A      CALL    #0ADB       ; build datastructure random ladders
0BD3    11006B      LD      DE,#6B00    ; load DE with start of datastructure random ladders
0BD6    CDA70D      CALL    #0DA7       ; draw the random ladders
0BD9    C9          RET                 ; return
```

Gaat nog niet goed. De eerste rij (bovenste) wordt goed getekend. De tweede rij lijkt qua Y-positie niet helemaal goed te zijn. De derde rij is qua X-positie niet goed. De top en bottom van de ladders hebben verschillende X-waarden. Hoe kan dat? Dat zou kunnen komen doordat de code de middelste rij anders moet afhandelen dan de ander twee rijen. Maar wellicht is die check niet goed en is het afwijkende gedrag pas bij de laatste rij. En dan nog onderzoeken wat er precies fout gaat.

Klopt inderdaad. De voorwaarde op #0B73 aangepast van 01 (bij PowerShell doorloopt de for-loop de waarden 0, 1 en 2) naar 02 (hier doorloopt de loop de waarden 3,2 en 1).

Nog wel dat nu in de middelste rij de x-posities van de top en bottom van de ladder nog steeds verschillend zijn. Moet fout in de logica zijn. Dat klopt. In de code voor de middelste rij, wordt de X-waarde uit D gelezen aangepast en voor de top ingevuld. Dan voor de bottom wordt de originele X-waarde weer uit D gelezen. Op te lossen door tussendoor de aangepaste waarde in A weer terug te schrijven naar D. Alles verschuiven. Maar dat werkt nu goed.

De ladders worden wel getekend, maar kunnen nog niet beklommen worden omdat ze nog geactiveerd moeten worden.

```
245D    CAB80B      JP      Z,#0BB8     ; if girders, jump to additional code
```

en:

```
0BB8    CD7124      CALL    #2471       ; activate default ladders
;
0BBB    21006B      LD      HL,#006B    ; load HL with start of data structure random ladders
~~0BC0    DD210763    LD      IX,#6307    ; #6307 is used for ladder positions?~~
0BBE    C37524      JP      #2475       ; activate random ladders
```

Ja dat werkt goed.

Nu versie v0.07 opgeslagen.

De titel aanpassen:

Op #36B2 staat de HIGH SCORE tekst:

#36B2: 80 76 18 19 17 18 10 23 13 1F 22 15 3F

Veranderen naar RNDMZR:

#36B2: 80 76 10 10 22 1E 14 1D 2A 22 10 10 3F

En dan in #3F18 de titel maken:
RNDMZR

8E 76 10 22 1E 14 1D 2A 22 10 3F

Nu versie v0.08 opgeslagen.

Nu kijken vrijmaken ruimte: Goofy Kongs verwijderen en alleen one player maken. En dan kijken wat er met de beschikbare ruimte allemaal nog mogelijk is.

Ruimte vrijmaken door de Goofy Kongs niet meer zo te doen. Afkijken bij DK Barrelpalooza.

Eerst hele stuk verwijderen: #0BFD t/m #0C81.

Dan volgende toevoegen:

```
0BFD  21BC75    LD     HL,#75BC      ; load HL with screen location start for goofy kong
;
0C00  0E50      LD     C,#50         ; C := #50 = start graphic for goofy kong
;
0C02  71        LD     (HL),C        ; draw part of goofy kong
0C03  0C        INC    C             ; next graphic
0C04  2B        DEC    HL            ; next screen location
0C05  71        LD     (HL),C        ; draw part of goofy kong
0C06  0C        INC    C             ; next graphic
0C07  2B        DEC    HL            ; next screen location
0C08  71        LD     (HL),C        ; draw part of goofy kong
0C09  0C        INC    C             ; next graphic
0C0A  2B        DEC    HL            ; next screen location
0C0B  71        LD     (HL),C        ; draw part of goofy kong
0C0C  79        LD     A,C           ; load A with graphic number
0C0D  FE67      CP     #67           ; == #67 ? (are we done?)
0C0F  CA820C    JP     Z,#0C82       ; yes, skip forward
;
0C12  0C        INC    C             ; next C
0C13  112300    LD     DE,#0023      ; load DE with offset
0C16  19        ADD    HL,DE         ; add to screen location
0C17  C3020C    JP     #0C02         ; loop again
```

Ja dat werkt. Stuk vrijgekomen #0C1A t/m #0C81.

Ruimte vrijmaken door one player te maken en de two player functionaliteit verwijderen. Afkijken bij DK Wizardry.

Eerste stuk is gebaseerd op gamemode1.

Als de game nu opgestart wordt, wordt in het high score scherm de optie getoond met keuze 1 player of 2 players. De teksten daarvan aanpassen. Schermen gekopieerd uit DUEL design.

De regel met 2 2 weghalen en het scherm met PUSH 1 OR 2 PLAYERS BUTTON niet tonen.

De regel met 2 2 wordt hier getekend:

```
07A3  ED5B2260  LD      DE,(CoinsPerCredit)      ; D := CoinsPer2Credits; E := CoinsPerCredit
07A7  216C75    LD      HL,#756C         ; load HL with screen RAM location
07AA  CDAD07    CALL    #07AD            ; run this sub below twice

07AD  73        LD      (HL),E           ; draw to screen number of coins needed for 1 player game
07AE  23        INC     HL               ;
07AF  23        INC     HL               ; next screen location 2 rows down
07B0  72        LD      (HL),D           ; draw to screen number of coins neeeded for 2 player game
07B1  7A        LD      A,D              ; A := D
07B2  D60A      SUB     #0A              ; subtract #A (10 decimal). result == 0 ?
07B4  C2BC07    JP      NZ,#07BC         ; no, skip next 3 steps

07B7  77        LD      (HL),A           ; else draw this zero to screen
07B8  3C        INC     A                ; increase A, A := 1 now
07B9  328E75    LD      (#758E),A        ; draw 1 to screen in front of the zero, so it draws "10" credits needed for 2 players

07BC  110102    LD      DE,#0201         ; D := 2, E := 1, used for next loop for 1 player and 2 players
07BF  218C76    LD      HL,#768C         ; set screen location to draw for next loop if needed
07C2  C9        RET                      ; return
```

Voldoende om de #07B0 NOP te maken?

Ja dat werkt.

De teksten voor 1 player en 2 player worden hier getoond:

```
08D5  0604      LD      B,#04           ; B := 4 = 0100 binary
08D7  1E09      LD      E,#09           ; E := 9 , code for "ONLY 1 PLAYER BUTTON"
08D9  3A0160    LD      A,(NumCredits)  ; load A with number of credits
08DC  FE01      CP      #01             ; == 1 ?
08DE  CAE408    JP      Z,#08E4         ; yes, skip next 2 steps

08E1  060C      LD      B,#0C           ; B := #0C = 1100 binary
08E3  1C        INC     E               ; E := #0A, code for "1 OR 2 PLAYERS BUTTON"

08E4  3A1A60    LD      A,(FrameCounter)        ; load A with # Timer constantly counts down from FF to 00
08E7  E607      AND     #07             ; mask bits. zero ?
08E9  C2F308    JP      NZ,#08F3        ; no, skip next 3 steps

08EC  7B        LD      A,E             ; yes, load A with E for code of text to draw, for buttons to press to start
08ED  CDE905    CALL    #05E9           ; draw text to screen
08F0  CD1606    CALL    #0616           ; draw credits on screen
```

De tweede tekst weghalen door op #08DE een JP te doen i.p.v. JP Z.

Ja dat werkt.

En dan nu nog ervoor zorgen dat er geen 2 player game opgestart kan worden.

Dat wordt hier afgehandeld:

```
; jump from #08B5 when GameMode2 == 1

08F8  CDD508    CALL    #08D5           ; draws press player buttons and loads A with IN2, masked by possible player numbers
08FB  FE04      CP      #04             ; is the player 1 button pressed ?
08FD  CA0609    JP      Z,#0906         ; yes, skip ahead

0900  FE08      CP      #08             ; is the player 2 button pressed ?
0902  CA1909    JP      Z,#0919         ; yes, skip ahead

0905  C9        RET                     ; return to #00D2
```

Dus geen two -player meer starten wanneer #0900 t/m #0904 in NOPS veranderd worden.

Dat klopt. Nu geen two-player mode meer.

En dan nu ruimte vrijgeven op de plekken waar afhandeling van de two-player zit.

De two-player start kan vrijgegeven worden: #0919 t/m #0937.

Daarna wordt de keuze op verschillende plekken gemaakt. Steeds op basis van de variabelen TwoPlayerGame, PlayerTurnA en PlayerTurnB. Dus daar op zoeken en de aanpassingen doen.

Eerst kijken naar de TwoPlayerGame variabele.

Wordt hier gebruikt:

```
0331  3A0F60    LD      A,(TwoPlayerGame)       ; load A with # of players in game
0334  A7        AND     A               ; is this a 1 player game?
0335  C8        RET     Z               ; yes, return

0336  3A0D60    LD      A,(PlayerTurnA)       ; Load current player #
0339  EE01      XOR     #01             ; change player from 1 to 2 or from 2 to 1
033B  CD4703    CALL    #0347           ; Loads HL with location for score (either player 1 or 2)

033E  3C        INC     A               ; increase A, now it has the number of the player
033F  77        LD      (HL),A          ; draw player number on screen
0340  19        ADD     HL,DE           ; next column
0341  3625      LD      (HL),#25        ; draw "U" on screen
0343  19        ADD     HL,DE           ; next column
0344  3620      LD      (HL),#20        ; draw "P" on screen
0346  C9        RET                     ; return
```

Hier is niets te winnen.

Wordt hier gebruikt:

```
079B  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players in game
079E  FE01      CP      #01             ; 2 player game?
07A0  CCEE09    CALL    Z,#09EE         ; yes, skip ahead to handle
```

Hiermee wordt de 2UP getekend:

```
09EE  3E02      LD      A,#02           ; load A with "2"
09F0  32E074    LD      (#74E0),A       ; write to screen
09F3  3E25      LD      A,#25           ; load A with "U"
09F5  32C074    LD      (#74C0),A       ; write to screen
09F8  3E20      LD      A,#20           ; load A with "P"
09FA  32A074    LD      (#74A0),A       ; write to screen
09FD  C9        RET                     ; return
```

Dat stuk kan vrijgegeven worden: #09EE t/m #09FD.

Code van #09D6 t/m #09FD alleen in two-player mode. Is deels al vrijgegeven.

En dan vrijgeven: #09D6 t/m #09FD.

Wordt hier gebruikt:

```
09BD  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
09C0  A7        AND     A               ; 1 player game?
09C1  210960    LD      HL,WaitTimerMSB ; load HL with timer address
09C4  110A60    LD      DE,GameMode2        ; load DE with game mode2 address
09C7  CAD009    JP      Z,#09D0         ; if 1 player game, skip ahead

; 2 player game

09CA  3678      LD      (HL),#78        ; store #78 into timer
09CC  EB        EX      DE,HL           ; DE <> HL.  HL now has game mode2
09CD  3602      LD      (HL),#02        ; GameMode2 := 2
09CF  C9        RET                     ; return

; 1 player game

09D0  3601      LD      (HL),#01        ; store 1 into timer
09D2  EB        EX      DE,HL           ; DE <> HL.  HL now has game mode2
09D3  3605      LD      (HL),#05        ; GameMode2 := 5
09D5  C9        RET                     ; return
```

Dit aan te passen naar:

```
09BD  210960    LD      HL,WaitTimerMSB     ; load HL with timer address
09C0  110A60    LD      DE,GameMode2        ; load DE with game mode2 address
09C3  3601      LD      (HL),#01        ; store 1 into timer
09C5  EB        EX      DE,HL           ; DE <> HL.  HL now has game mode2
09C6  3605      LD      (HL),#05        ; GameMode2 := 5
09C8  C9        RET                     ; return
```

En dan vrijgeven: #09C9 t/m #09D5.

Let op: voor two-player game wordt GameMode2 op 2 gezet. Checken of die dan alleen dan gebruikt wordt, want dan kan de hele afhandeling van GameMode2 = 2 komen te vervallen?

Wordt hier gebruikt:

```
1315  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
1318  A7        AND     A               ; 1 player game?
1319  2807      JR      Z,#1322         ; yes, skip next 3 steps
```

Niets te winnen.

Wordt hier gebruikt:

```
1336  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
1339  A7        AND     A               ; 1 player game?
133A  CA3F13    JP      Z,#133F         ; yes, skip next step
```

Niets te winnen.

Dan zijn alle TwoPlayerGame variabelen bekeken.

Eerst eens kijken naar de GameMode2 = 2 versus GameMode2 =5 zoals hierboven opgemerkt:

```
06FE  3A0A60    LD      A,(GameMode2)       ; load A with game mode2
0701  EF        RST     #28             ; jump based on what the game state is

0702  86 09                             ; (0) #0986     ; game start = clears screen, clears sounds, sets screen flip if needed
0704  AB 09                             ; (1) #09AB     ; copy player data, set screen, set next game mode based on number of players
0706  D6 09                             ; (2) #09D6     ; clears palettes, draws "PLAYER <I>", draws player2 score, draws "2UP" (2 player game only)
0708  FE 09                             ; (3) #09FE     ; copy player data into correct area (2 player game only)
070A  1B 0A                             ; (4) #0A1B     ; clears palletes, draws "PLAYER <II>", update player2 score, draw "2UP" to screen (2 player game only)
070C  37 0A                             ; (5) #0A37     ; updates high score, player score, remaining lives, level, 1UP
070E  63 0A                             ; (6) #0A63     ; clears screen and sprites, check for intro screen to run
```

Lijkt erop dat een 2 player game GameMode2 via 2 -> 3 -> 4-> 5 loopt, terwijl een 1 player game meteen naar 5 gaat. Dus dan kun je de code voor GameMode2 =2 en 3 en 4 verwijderen.

Dat is vrijgeven van de code van: #09D6 t/m #0A36.

Ja dat gaat nog steeds goed.

Met de debugger kijken welke GameMode2 waardes doorlopen worden. Alle stukken voor niet gebruikte GameMode2 waarden kunnen weg (wel checken of hier geen verwijzingen naar staan: soms gedeelde stukken en wordt er naar een deel van een ander stuk code gesprongen).

GameMode2 geheugenplaats is: #600A daarop monitoren in de debugger.

0 – 6 – 7 (DK logo) – 0 – 1 (High score)– 2 (attract) – 3 (attract) – 4 (dood attract) – 5 – (terug naar 6)
0 - 1 (High Score credit) – 5 – 6 – 7 – 8 -9 – A (How high)– B – C (spel) – D (dood) – E (dood) -
8 – 9 - A - B – C (spel) – D (dood) – E (dood) -
8 – 9 – A – B – C (spel) - D (dood) – E – 10 (Game Over) – 14 – 0 – 1 … en opnieuw

En board halen:

E – 8 – 9 – A – B – C (spel) – 16 (DK + Pauline) – 8 – 9 - ….

En even invincibility aanzetten en dan kijken wat er gebeurd bij einde level (rivets halen) en wat er gebeurd bij game over maar dan entry in high score table.

C (spel rivets) – 16 (DK valt) – 8 – 9 – A – B – C (spel barrels) …

C (spel) – D (dood) – E (dood) – 10 (Game Over) – 14 – 15 (initials) – 14 (registered) – 0 – 1 …

Dan dus niet gebruikt: F, 11, 12, 13 en 17.

Kijken of die vrijgemaakt kunnen worden.

F = vrijgeven #1344 t/m #138E.
11 = vrijgeven #13A1 t/m #13A9.

12 = vrijgeven #13AA t/m #13BA.
13 = vrijgeven #13BB t/m #13C9.
17 = vrijgeven #196B t/m #1976.

Testen met een spel spelen met score invoeren. Gaat dat allemaal goed? Dan ruimte succesvol vrijgemaakt en dan met aanpassingen van rivets beginnen.

Ja dat gaat goed. Tot level 4 gespeeld en daarna afgegaan en initialen ingevoerd.

Nu versie v0.10 opgeslagen.

Eerst de seed regelen: echt random of via dip-switches een aantal vast seeds. Dan wel de HL bij opbouwen van board saven en terugzetten als je afgaat. Nee dat niet doen vooralsnog. Dus puur een random seed bepalen net zoals bij DK Wizardry. Nadeel van terugzetten van HL wanneer je afgaat, is dat een echt moeilijk board je meerdere levens kan kosten. Als je de HL echter niet terugzet, dan krijg je dat bij afgaan, de boards niet meer hetzelfde zijn. Dus daarom eenvoudig houden.

Random seed bepalen en overflow testen. Afkijken bij DK Wizardry.

Wanneer een nieuwe game gestart wordt, wordt het scherm gewist:

```
; arrive from #0701 when GameMode2 == 6
; clears screen and sprites, check for intro screen to run

0A63  DF       RST    #18             ; count down WaitTimerMSB and only continue here if == 0, else return to higher sub.
0A64  CD7408   CALL   #0874           ; clears the screen and sprites
0A67  210960   LD     HL,WaitTimerMSB ; load HL with timer
0A6A  3601     LD     (HL),#01        ; set timer to 1
0A6C  2C       INC    L               ; HL := GameMode2
0A6D  34       INC    (HL)            ; increase game mode2 to 7
0A6E  112C62   LD     DE,#622C        ; load DE with game start flag address
0A71  1A       LD     A,(DE)          ; load A with game start flag
0A72  A7       AND    A               ; is this game just beginning?
0A73  C0       RET    NZ              ; yes, return

0A74  34       INC    (HL)            ; else increase game mode2 to 8 - skip kong intro to begin
0A75  C9       RET                    ; return
```

Daar eventueel het random maken van de seed aan toevoegen?

Eerst testen met seed vlak onder #3FFF en dan kijken of de overflow goed gaat.

Aanpassen:

```
0A67    CD713F      CALL    #3F71       ; call to additional code
```

en:

```
3F71    213262      LD      HL,#6232    ; load HL with startaddress for random index
3F74    363F        LD      (HL),#3F    ; load random index H with #3F
3F76    23          INC     HL
```

```
3F77    36F0        LD      (HL),#F0        ; load random index L with #F0
;
3F79    210960      LD      HL,WaitTimerMSB ; load HL with timer
3F7C    C9          RET                     ; return
```

Ja, de overflow gaat goed.

Nu zo maken dat er een willekeurige waarde ingevuld wordt:

```
3F71    213262      LD      HL,#6232        ; load HL with startaddress for random index
;
3F74    3A1860      LD      A,(RngTimer1)   ; load A with random value (#00-#FF)
3F77    E63F        AND     #3F             ; A is random value (#00-#3F)
3F79    77          LD      (HL),A          ; set random index H
3F7A    23          INC     HL
3F7B    3A1A60      LD      A,(FrameCounter)        ; load A with random value (#00-#FF)
3F7E    77          LD      (HL),A          ; set random index L
;
3F7F    210960      LD      HL,WaitTimerMSB ; load HL with timer
3F82    C9          RET                     ; return
```

Ja dat gaat goed.

Nu versie v0.11 opgeslagen.

Rivets zo maken dat alleen de trappen variabel neergezet worden. En zo maken dat links in de definitie staat en dat er voor rechts alleen een spiegel code staat. Dan dezelfde ladder nogmaals maken met een nieuwe x-positie van 254 – oude x-positie. Variabel door vier mogelijke X-waarden te definiëren en random één van de vier te kiezen. En dan de twee middelste trappen nog apart variabel maken. In ieder geval wel altijd een andere indeling, maar overzichtelijk qua code en geen aanpassingen van de hoogte van de girders. Hierbij overigens eerst de ladders tekenen en dan pas de rest van het scherm zodat de girders over de ladders getekend worden. En misschien random één van de trappen gebroken maken?

Eerste deel met de variabele ladders. Op basis van een gedefinieerde opbouw uitlezen van een definitie en dat omzetten naar een tijdelijke definitie (vanaf #6B00, net als bij barrels board). Sommige ladders nooit broken maken omdat het anders veel te moelijk wordt. Eigenlijk alleen de vier middelste ladders komen hiervoor in aanmerking. Nog kijken hoe dit hierin te brengen.

Dit is de originele definitie van het rivets board:

```
3C8B:  00 7B 80 7B A8   ; center ladder level 3
3C90:  00 7B D0 7B F8   ; bottom center ladder
3C95:  00 33 58 33 80   ; top left ladder
3C9A:  00 53 58 53 80   ; top left ladder (right side)
3C9F:  00 AB 58 AB 80   ; top right ladder (left side)
3CA4:  00 CB 58 CB 80   ; top right ladder
3CA9:  00 2B 80 2B A8   ; level 3 ladder left side
```

```
3CAE:  00 D3 80 D3 A8   ; level 3 ladder right side
3CB3:  00 23 A8 23 D0   ; level 2 ladder left side
3CB8:  00 5B A8 5B D0   ; level 2 ladder #2 of 4
3CBD:  00 A3 A8 A3 D0   ; level 2 ladder #3 of 4
3CC2:  00 DB A8 DB D0   ; level 2 ladder right side
3CC7:  00 1B D0 1B F8   ; bottom left ladder
3CCC:  00 E3 D0 E3 F8   ; bottom right ladder
3CD1:  05 B7 30 48 30   ; girder above kong
3CD6:  05 CF 58 30 58   ; girder kong stands on
3CDB:  05 D7 80 28 80   ; level 4 girder
3CE0:  05 DF A8 20 A8   ; level 3 girder
3CE5:  05 E7 D0 18 D0   ; level 2 girder
3CEA:  05 EF F8 10 F8   ; bottom level girder
3CEF:  AA               ; end code
```

Met groen aangegeven onderdelen zijn de standaard girders die aan het einde getekend dienen te worden.

Standaard structuur voor de definitie is:

- Normal: X1 – X2 – X3 – X4 – YT – YB
- Mirror: FF
- End: AA

Omzetten:

```
3B35: 73 7B 83 8B 80 A8        - center ladder level 3 (#3C8B)
3B3B: 73 7B 83 8B D0 F8        - bottom center ladder (#3C90)
3B41: 53 5B 63 6B A8 D0        - level 2 ladder #2 of 4 (#3CB8)
3B47: FF                       - level 2 ladder #3 of 4 (#3CBD)
;
3B48: 33 43 33 43 58 80        - top left ladder (#3C95)
3B4E: FF                       - mirror – top right ladder (#3CA4)
3B4F: 53 5B 63 53 58 80        - top left ladder right side (#3C9A)
3B55: FF                       - mirror – top right ladder left side (#3C9F)
3B56: 2B 3B 2B 3B 80 A8        -  level 3 ladder left side (#3CA9)
3B5C: FF                       - mirror – level 3 ladder right side (#3CAE)
3B5D: 23 33 43 23 A8 D0        - level 2 ladder left side (#3CB3)
3B63: FF                       - mirror – level 2 ladder right side (#3CC2)
3B64: 1B 2B 3B 1B D0 F8        - bottom left ladder (#3CC7)
3B6A: FF                       - mirror – bottom right ladder (#3CCC)
3B6B: AA                       - end code
```

En een teller laten meelopen en één van de eerste vier op basis van een random gekozen getal tussen 1 en 4 broken maken.

Eerst dit opnemen. Meteen achter de basis definitie van het barrels board aan, zodat e.e.a. gaat aansluiten. Dan houden we wellicht daar nog een interessant stuk vrije ruimte over. Barrels definitie eindigt nu op: #3B34. Dus weer verder gaan op #3B35. Daarmee maken we dan wel een van de

andere boards stuk. Maar dat later herstellen als we met het betreffende board aan de gang gaan door definitie uit een originele rom te kopiëren.

En dan het vaste deel met de girders daar achter plaatsen:

```
3B6C:  05 B7 30 48 30   ; girder above kong
3B71:  05 CF 58 30 58   ; girder kong stands on
3B76:  05 D7 80 28 80   ; level 4 girder
3B7B:  05 DF A8 20 A8   ; level 3 girder
3B80:  05 E7 D0 18 D0   ; level 2 girder
3B85:  05 EF F8 10 F8   ; bottom level girder
3B8A:  AA               ; end code
```

HL laten wijzen naar de structuur waar de definitie opgebouwd wordt. En dan de HL daardoorheen laten lopen. IX laten wijzen naar de structuur waar de definitie nu staat.

```
Memory 1: #66A0     - broken ladder number
Memory 2: #66A1     - ladder counter
```

```
0C1A   DD21353B   LD     IX,#3B35      ; load IX with pointer to input data structure
0C1E   21006B     LD     HL,#6B00      ; load HL with pointer to output data structure
;
0C21   3A1860     LD     A,RngTimer1   ; get random number for broken ladder number
0C24   E603       AND    #03           ; random number between 0 and 3
0C26   C601       ADD    A,#01         ; random number between 1 and 4
0C28   32A066     LD     (#66A0),A     ; store in brokenladderskip
;
0C2B   3E00       LD     A,#00         ; A == #00
0C2D   32A166     LD     (#66A1),A     ; store in ladder counter
;
0C30   DD7E00     LD     A,(IX+#00)    ; load data element – label EEEE
0C33   FEAA       CP     #AA           ; is this the last data element?
0C35   2002       JR     NZ,#AAAA      ; no, skip next steps – jump to label AAAA
;
0C37   77         LD     (HL), A       ; yes, write end to data structure
0C38   C9         RET                  ; return
;
0C39   FEFF       CP     #FF           ; is this a mirror data element? – label AAAA
0C3B   2009       JR     NZ,#BBBB      ; no, skip next steps – jump to label BBBB
;
0C3D   DD23       INC    IX            ; increase pointer to input data structure
0C3F   3EFE       LD     A,#FE         ; yes, load A with mirror aid
0C41   90         SUB    B             ; subtract original x-value
0C42   47         LD     B,A           ; load B with mirror x-value
0C43   C3D409     JP     #09D4         ; add the mirrored ladder – jump to label CCCC
;
0C46   E5         PUSH   HL            ; save HL for later – label BBBB
```

```
0C47    C5          PUSH    BC          ; save BC for later
0C48    CD7B0A      CALL    #0A7B       ; get random value in A
0C4B    C1          POP     BC          ; restore BC
0C4C    E1          POP     HL          ; restore HL
;
0C4D    E603        AND     #03         ; A is value between 0 and 3
;
0C4F    FE00        CP      A,#00       ; A = 00?
0C51    2005        JR      NZ,XXXX     ; no, skip next steps
0C53    DD4600      LD      B,(IX+#00)  ; load B with first x-value
0C56    1815        JR      #DDDD       ; jump to label DDDD
;
0C58    FE01        CP      A,#01       ; A = 01?
0C5A    2005        JR      NZ,XXXX     ; no, skip next steps
0C5BC   DD4601      LD      B,(IX+#01)  ; load B with second x-value
0C5F    180C        JR      #DDDD       ; jump to label DDDD
;
0C61    FE02        CP      A,#02       ; A = 02?
0C63    2005        JR      NZ,XXXX     ; no, skip next steps
0C65    DD4602      LD      B,(IX+#02)  ; load B with third x-value
0C68    1803        JR      #DDDD       ; jump to label DDDD
;
0C6A    DD4603      LD      B,(IX+#03)  ; load B with fourth x-value
;
0C6D    DD23        INC     IX          ; increase pointer to input data  - label DDDD
0C6F    DD23        INC     IX
0C71    DD23        INC     IX
0C73    DD23        INC     IX
;
0C75    C3C909      JP      #09C9       ; jump to continue code

en:

09C9    DD7E00      LD      A,(IX+#00)  ; load C with y-value top
09CC    DD23        INC     IX          ; increase pointer to input data
09CE    57          LD      D,A         ; store into D
09CF    DD4E00      LD      C,(IX+#00)  ; load C with y-value bottom
09D2    DD23        INC     IX
;
09D4    C5          PUSH    BC          ; store BC for later – label CCCC
09D5    3AA166      LD      A,(#66A1)   ; load ladder counter
09D8    3C          INC     A           ; increase ladder counter
09D9    32A166      LD      (#66A1),A   ; store increased ladder counter
;
09DC    47          LD      B,A         ; store ladder counter in B
;
09DD    3AA066      LD      A,(#66A0)   ; load broken ladder number
09E0    90          SUB     B           ; is ladder counter equal to broken ladder number?
```

```
09E1    3E00        LD      A,#00       ; normal ladder
09E3    2002        JR      NZ,#XXXX    ; skip next step
09E5    3E01        LD      A,#01       ; broken ladder
;
09E7    77          LD      (HL),A      ; store laddertype in data structure
09E8    23          INC     HL          ; next element
;
09E9    C1          POP     BC          ; restore BC
09EA    70          LD      (HL),B      ; store xposition top in data structure
09EB    23          INC     HL          ; next element
;
09EC    72          LD      (HL),D      ; store yposition top in data structure
09ED    23          INC     HL          ; next element
;
09EE    70          LD      (HL),B      ; store xposition bottom in data structure
09EF    23          INC     HL          ; next element
;
09F0    71          LD      (HL),C      ; store yposition bottom in data structure
09F1    23          INC     HL          ; next element
;
09F2    C3300C      JP      #0C30       ; loop again – jump to label EEEE
```

En dan ergens ter test aanroepen om met de debugger te kijken of de structuur juist opgebouwd wordt.

Tekenen van het scherm was als volgt aangepast:

```
0CC6    CDC70B      CALL    #0BC7       ; draw the screen
```

en:

```
0BC7    CDA70D      CALL    #0DA7       ; draw the screen
;
0BCA    3A2762      LD      A,(#6227)   ; load A with screen number
0BCD    FE01        CP      #01         ; is this the girders?
0BCF    C0          RET     NZ          ; no, return
;
0BD0    CDDB0A      CALL    #0ADB       ; build datastructure random ladders
0BD3    11006B      LD      DE,#6B00    ; load DE with start of datastructure random ladders
0BD6    CDA70D      CALL    #0DA7       ; draw the random ladders
0BD9    C9          RET                 ; return
```

Dit aanpassen.

```
0CC6    CD4413      CALL    #1344       ; draw the screen
```

en:

```
1344    3A2762          LD      A,(#6227)       ; load A with screen number
1347    FE04            CP      #04             ; is this the rivets?
1349    C2C70B          JP      NZ,#0BC7        ; no, jump to additional code
;
134C    D5              PUSH    DE              ; save DE for later
134D    CD1A0C          CALL    #0C1A           ; build datastructure random ladders
1350    11006B          LD      DE,#6B00        ; load DE with start of datastructure random ladders
1353    CDA70D          CALL    #0DA7           ; draw the random ladders
;
1356    D1              POP     DE              ; restore DE
1357    CDA70D          CALL    #0DA7           ; draw the screen
;
135A    C9              RET                     ; return
```

Ook nog aanpassen dat de rivets definitie voor de rest nu start bij #3B6C.

```
0CC3  116C3B          LD      DE,#6C3B        ; load DE with start of table data for rivets
```

Niet zo handig. De random opcode gelezen uit de tabel wordt standaard naar een waarde tussen 0 en 6 geconverteerd. Beter om dat apart te houden.

```
0A7B    3A3262          LD      A,(#6232)       ; get stored value for H – label BBBB
0A7E    67              LD      H,A             ; store in H
0A7F    3A3362          LD      A,(#6233)       ; get stored value for L
0A82    3C              INC     A               ; store in L – INC A (3C)
0A83    6F              LD      L,A             ; increase L – LD L,A (6F)
0A84    2008            JR      NZ              ; no carry, skip next steps – jump to label CCCC
;
0A86    24              INC     H               ; increase H
0A87    7C              LD      A,H             ; load A with H
0A88    FE40            CP      #40             ; H==#40 ?
0A8A    2002            JR      NZ              ; no, skip next steps – jump to label CCCC
;
0A8C    2601            LD      H,#01           ; reset H
;
0A8E    7E              LD      A,(HL)          ; load opcode from HL – label CCCC
0A8F    47              LD      B,A             ; store A in B
0A90    7C              LD      A,H
0A91    323262          LD      (#6232),A       ; save H
0A94    7D              LD      A,L
0A95    323362          LD      (#6233),A       ; save L
0A98    78              LD      A,B             ; restore A from B
;
~~0A99    E601            AND     #01             ; keep bit 0 – value between 0 and 1~~
~~0A9B    4F              LD      C,A             ; save to C~~
~~;~~
~~0A9C    78              LD      A,B             ; restore original opcode value from B~~
~~0A9D    1F              RRA                     ; rotate right~~
```

```
0A9E    E601            AND     #01             ; keep bit 0 – value between 0 and 1
0AA0    81              ADD     A,C             ; add C
0AA1    4F              LD      C,A             ; save to C
;
0AA2    78              LD      A,B             ; restore original opcode value from B again
0AA3    1F              RRA                     ; rotate right
0AA4    E603            AND     #03             ; keep bits 0 and 1 – value between 0 and 3
0AA6    81              ADD     A,C             ; add C
;
0AA7    C9              RET                     ; return
;
0A99    C9              RET
```

Eigenlijk weer terug naar een C9 op #0A99 en de code daarna vooraf laten gaan door een CALL naar #0A7B.

```
0A25    CD7B0A          CALL    #0A7B           ; get value of opcode
0A28    E601            AND     #01             ; keep bit 0 – value between 0 and 1
0A2A    4F              LD      C,A             ; save to C
;
0A2B    78              LD      A,B             ; restore original opcode value from B
0A2C    1F              RRA                     ; rotate right
0A2D    E601            AND     #01             ; keep bit 0 – value between 0 and 1
0A2F    81              ADD     A,C             ; add C
0A30    4F              LD      C,A             ; save to C
;
0A31    78              LD      A,B             ; restore original opcode value from B again
0A32    1F              RRA                     ; rotate right
0A33    E603            AND     #03             ; keep bits 0 and 1 – value between 0 and 3
0A35    81              ADD     A,C             ; add C
;
0A36    C9              RET                     ; return
```

En dan de aanroep naar #0A7B aanpassen naar #0A25:

```
0AA8    DD21A466        LD      IX,#66A4        ; set IX to index #66A4 – label AAAA
0AAC    CD250A          CALL    #0A25           ; get random number 0-11
0AAF    DD7700          LD      (IX+#00),A      ; store in #66A4
0AB2    CD250A          CALL    #0A25           ; get random number 0-11
0AB5    DD7701          LD      (IX+#01),A      ; store in #66A5
0AB8    CD250A          CALL    #0A25           ; get random number 0-11
```

Ja dat werkt. Hierdoor bruikbaar in de code hierboven.

Werkt nu, na een aantal kleine herstelwijzigingen, goed met het tekenen.

Nu nog zorgen dat de ladders geactiveerd worden.

246E 21006B LD HL,#6B00 ; load HL with table data for rivets

Nu versie v0.14 opgeslagen.

Testen moet uitwijzen hoe moeilijk/makkelijk sommige boarden zijn. Wellicht barrels te makkelijk: kijken of moeilijker maken d.m.v. normal barrel skip i.p.v. broken barrel skip en wellicht rivets te moeilijk en dan wellicht makkelijk door geen gebroken ladder?

Eerst kijken randomization van het pies board.

Dit is de originele definitie van het pies board:

```
3B5D:  06 8F 90 70 90   ; central patch of XXX's
3B62:  06 8F 98 70 98   ; central patch of XXX's
3B67:  06 8F A0 70 A0   ; central patch of XXX's
3B6C:  00 63 18 63 58   ; kong's ladder (right)
3B71:  00 63 80 63 A8   ; center ladder to left of oil can fire
3B76:  00 63 D0 63 F8   ; bottom level ladder #2 of 4
3B7B:  00 53 18 53 58   ; kong's ladder (left)
3B80:  00 53 A8 53 D0   ; ladder under the hat
3B85:  00 9B 80 9B A8   ; center ladder to right of oil can fire
3B8A:  00 9B D0 9B F8   ; bottom level ladder #3 of 4
3B8F:  01 23 58 23 80   ; top broken ladder left side
3B94:  01 DB 58 DB 80   ; top broken ladder right side
3B99:  00 2B 80 2B A8   ; ladder on left platform with hammer
3B9E:  00 D3 80 D3 A8   ; ladder on right platform with umbrella
3BA3:  00 A3 A8 A3 D0   ; ladder to right of bottom hammer
3BA8:  00 2B D0 2B F8   ; bottom level ladder #1 of 4
3BAD:  00 D3 D0 D3 F8   ; bottom level ladder #4 of 4
3BB2:  00 93 38 93 58   ; ladder leading to girl
3BB7:  02 97 38 68 38   ; girder where girl sits
3BBC:  03 EF 58 10 58   ; top conveyor girder
3BC1:  03 F7 80 88 80   ; top right conveyor next to oil can
3BC6:  03 77 80 08 80   ; top left conveyor next to oil can
3BCB:  02 A7 A8 50 A8   ; center ledge
3BD0:  02 E7 A8 B8 A8   ; right center ledge
3BD5:  02 3F A8 18 A8   ; left center ledge (has hammer)
3BDA:  03 EF D0 10 D0   ; main lower conveyor girder (has hammer)
3BDF:  02 EF F8 10 F8   ; bottom level girder
3BE4:  AA               ; end code
```

Met groen aangegeven onderdelen zijn de standaard onderdelen.

Standaard structuur voor de definitie is:

- Normal: X1 – X2 – X3 – X4 – YT – YB
- Mirror: FF
- End: AA

Omzetten:

```
3B8B: 23 2B 33 3B D0 F8      - bottom level ladder #1 of 4 (#3BA8)
3B91: FF                     - mirror - bottom level ladder #4 of 4 (#3BAD)
3B92: 43 53 63 73 D0 F8      - bottom level ladder #2 of 4 (#3B76)
3B98: FF                     - mirror – bottom level ladder #3 of 4 (#3B8A)
;
3B99: 5B 6B 5B 6B A8 D0      - ladder under the hat (#3B80)
3B9F: FF                     - mirror – ladder to right of bottom hammer (#3BA3)
3BA0: 53 63 53 63 80 A8      - center ladder to left of oil can fire (#3B71)
3BA6: FF                     - mirror – center ladder to right of oil can fire (#3B85)
3BA7: 1B 2B 33 3B 80 A8      - ladder on left platform with hammer (#3B99)
3BAD: FF                     - mirror - ladder on right platform with umbrella (#3B9E)
3BAE: AA                     - end code
```

En het vaste deel daar achter:

```
3BAF:  06 8F 90 70 90   ; central patch of XXX's
3BB4:  06 8F 98 70 98   ; central patch of XXX's
3BB9:  06 8F A0 70 A0   ; central patch of XXX's
3BBE:  00 63 18 63 58   ; kong's ladder (right)
3BC3:  00 53 18 53 58   ; kong's ladder (left)
3BC8:  01 23 58 23 80   ; top broken ladder left side
3BCD:  01 DB 58 DB 80   ; top broken ladder right side
3BD2:  00 93 38 93 58   ; ladder leading to girl
3BD7:  02 97 38 68 38   ; girder where girl sits
3BDC:  03 EF 58 10 58   ; top conveyor girder
3BE1:  03 F7 80 88 80   ; top right conveyor next to oil can
3BE6:  03 77 80 08 80   ; top left conveyor next to oil can
3BEB:  02 AF A8 50 A8   ; center ledge
3BF0:  02 E7 A8 C0 A8   ; right center ledge
3BF5:  02 3F A8 18 A8   ; left center ledge (has hammer)
3BFA:  03 EF D0 10 D0   ; main lower conveyor girder (has hammer)
3BFF:  02 EF F8 10 F8   ; bottom level girder
3C04:  AA               ; end code
```

Ervoor zorgen dat dit getekend wordt.

Code voor tekenen rivets en barrels:

```
1344    3A2762      LD      A,(#6227)       ; load A with screen number
1347    FE04        CP      #04             ; is this the rivets?
1349    C2C70B      JP      NZ,#0BC7        ; no, jump to additional code
;
134C    D5          PUSH    DE              ; save DE for later
134D    CD1A0C      CALL    #0C1A           ; build datastructure random ladders
1350    11006B      LD      DE,#6B00        ; load DE with start of datastructure random ladders
1353    CDA70D      CALL    #0DA7           ; draw the random ladders
```

```
;
1356    D1          POP     DE          ; restore DE
1357    CDA70D      CALL    #0DA7       ; draw the screen
;
135A    C9          RET                 ; return
```

Aanpassen:

```
1349    C21909      JP      NZ,#0919    ; no, jump to additional code
```

en:

```
0919    FE01        CP      #01         ; is this the barrels?
091B    CAC70B      JP      Z, 0BC7     ; yes, jump to code that handles barrels
091E    FE03        CP      #03         ; is this the elevators?
0920    CAAAAA      JP      Z, AAAA     ; yes, jump to code that handles elevators
;
0923    CDA70D      CALL    #0DA7       ; draw the screen
;
0926    DD218B3B    LD      IX,#3B8B    ; load IX with pointer to input data structure
092A    21006B      LD      HL,#6B00    ; load HL with pointer to output data structure
;
092D    CD210C      CALL    #0C21       ; build datastructure random ladders
0930    11006B      LD      DE,#6B00    ; load DE with start of datastructure random ladders
0933    CDA70D      CALL    #0DA7       ; draw the random ladders
0936    C9          RET                 ; return
```

en:

```
0CDF  11AF3B   LD     DE,#3BAF      ; load DE with start of table data for conveyors
```

en:

```
2461  21006B   ld     HL,#6B00      ; load HL with start of table data for conveyors
```

Elevators jump op #0920 nu nog leeg laten.

De conveyors moeten na de trappen getekend worden. Anders ziet het er niet goed uit.

```
0919    FE01        CP      #01         ; is this the barrels?
091B    CAC70B      JP      Z, 0BC7     ; yes, jump to code that handles barrels
091E    FE03        CP      #03         ; is this the elevators?
0920    CAAAAA      JP      Z, AAAA     ; yes, jump to code that handles elevators
;
0923    D5          PUSH    DE
0924    DD218B3B    LD      IX,#3B8B    ; load IX with pointer to input data structure
0928    21006B      LD      HL,#6B00    ; load HL with pointer to output data structure
;
```

```
092B   CD210C      CALL    #0C21       ; build datastructure random ladders
092E   11006B      LD      DE,#6B00    ; load DE with start of datastructure random ladders
0931   CDA70D      CALL    #0DA7       ; draw the random ladders
0934   C3F509      JP      #09F5       ; jump to continue code

en:

09F5   D1          POP     DE
;
09F6   CDA70D      CALL    #0DA7       ; draw the screen
;
09F9   C9          RET                 ; return
```

De rectractable ladders worden niet geactiveerd. Nog apart activeren.

De code die dat doet, met HL gevuld met het begin van de table data.:

```
2471  DD210063  LD    IX,#6300      ; #6300 is used for ladder positions?
2475  110500    LD    DE,#0005      ; DE := 5 = offset

2478  7E        LD    A,(HL)        ; load A with the next item of data
2479  A7        AND   A             ; is this item == 0 ?
247A  CA8824    JP    Z,#2488       ; yes, jump ahead

247D  3D        DEC   A             ; no, decrease, was this item == 1 ?
247E  CA9E24    JP    Z,#249E       ; yes, jump down instead

2481  FEA9      CP    #A9           ; was the item == #AA ?
2483  C8        RET   Z             ; yes, return, we are done with this.  AA is at the end of each table

2484  19        ADD   HL,DE         ; if neither then add offset for next HL
2485  C37824    JP    #2478         ; loop again
```

Als HL de start van de nieuwe data structure met vaste elementen van het pies scherm (#3BAF) en als IX een juiste waarde nemen rekening houdend met de al geactiveerde trappen. #6300 is de standaard, en bijvoorbeeld met al 11 geactiveerde trappen dit aanpassen naar #630B. We hebben al 10 trappen geactiveerd, dus nu #630A nemen. Maar waar deze activatie toevoegen?

Het stuk code waarin ook het activeren van de trappen gedaan wordt, start bij #2441 en wordt aangeroepen vanaf 0D62:

```
0D62  CD4124      CALL    #2441       ;
```

Aanpassen:

```
0D62   C3FA09      JP      #09FA       ; jump to additional code – jump to label AAAA

en:
```

```
09FA    CD4124      CALL    #2441       ; label AAAA
;
09FD    3A2762      LD      A,(#6227)   ; load A with screen number
0A00    FE02        CP      #02         ; is this the conveyors?
0A02    C2650D      JP      NZ,#0D65    ; no, jump back
;
0A05    21AF3B      LD      HL,#3BAF    ; load HL with data table conveyors
0A08    DD210A63    LD      IX,#630A    ; load IX with index ladder activation
0A0C    CD7524      CALL    #2475       ; activate ladders
0A0F    C3650D      JP      #0D65       ; jump back
```

Lijkt goed te gaan. Wel alle ladders goed testen. Lijken het allemaal goed te doen.

Nu versie v0.17 opgeslagen.

Kijken of de verdeling van de trappen anders kan. Mogelijke trap vlak naast oil can? Beneden worden soms twee trappen tegen elkaar getekend. Dat ook aanpassen.

Eventjes kijken of er een spannender indeling gemaakt kan worden. Ook kijken of het mogelijk is om bepaalde ladders te verwijderen (door ze buiten beeld te plaatsen)?

```
3B8B: 23 2B 33 3B D0 F8     - bottom level ladder #1 of 4 (#3BA8)
3B91: FF                    - mirror - bottom level ladder #4 of 4 (#3BAD)
3B92: 4B 5B 6B 08 D0 F8     - bottom level ladder #2 of 4 (#3B76)
3B98: FF                    - mirror – bottom level ladder #3 of 4 (#3B8A)
;
3B99: 53 63 73 63 A8 D0     - ladder under the hat (#3B80)
3B9F: FF                    - mirror – ladder to right of bottom hammer (#3BA3)
3BA0: 5B 6B 5B 6B 80 A8     - center ladder to left of oil can fire (#3B71)
3BA6: FF                    - mirror – center ladder to right of oil can fire (#3B85)
3BA7: 1B 2B 33 3B 80 A8     - ladder on left platform with hammer (#3B99)
3BAD: FF                    - mirror - ladder on right platform with umbrella (#3B9E)
3BAE: AA                    - end code
```

of:

```
3B8B: 2B 2B 33 3B D0 F8     - bottom level ladder #1 of 4 (#3BA8)
3B91: FF                    - mirror - bottom level ladder #4 of 4 (#3BAD)
3B92: 4B 5B 6B 73 D0 F8     - bottom level ladder #2 of 4 (#3B76)
3B98: FF                    - mirror – bottom level ladder #3 of 4 (#3B8A)
;
3B99: 53 63 53 23 A8 D0     - ladder under the hat (#3B80)
3B9F: FF                    - mirror – ladder to right of bottom hammer (#3BA3)
3BA0: 5B 6B 5B 6B 80 A8     - center ladder to left of oil can fire (#3B71)
3BA6: FF                    - mirror – center ladder to right of oil can fire (#3B85)
3BA7: 1B 2B 33 3B 80 A8     - ladder on left platform with hammer (#3B99)
3BAD: FF                    - mirror - ladder on right platform with umbrella (#3B9E)
```

3BAE: AA - end code

De ladders op de onderste rij verwijderen (meteen twee dan door mirror) is niet handig want als van de andere twee er dan één gebroken is, dan is er nog maar een ladder om te ontsnappen aan fireballs en kun je snel ingesloten raken.

Ook eens testen met L=05.

Aanpassen naar:

095E 05 73 3A 01 00 00 00 ; #3A65 is start of table data for screens/levels

Is nu goed.

Nu versie v0.18 opgeslagen.

Kijken of de broken ladder skip aangepast kan worden naar een normal ladder skip. Nee, door de specifieke opbouw kan dat niet.

Nu gaan kijken naar de randomization mogelijkheden voor het elevators board. Dat is wel lastig omdat dit bord niet symmetrisch is en ook geen ruimte voor ladder variaties heeft. Dus op een andere manier regelen? Maar ook weer niet te veel afwijken van de standaard indeling.

Dit is de originele definitie van het elevators board:

```
3BE5:  00 63 18 63 58   ; kong's ladder (right)
3BEA:  00 63 88 63 D0   ; center ladder right
3BEF:  00 53 18 53 58   ; long's ladder (left)
3BF4:  00 53 88 53 D0   ; center ladder left
3BF9:  00 E3 68 E3 90   ; far top right ladder leading to purse
3BFE:  00 E3 B8 E3 D0   ; far bottom right ladder
3C03:  00 CB 90 CB B0   ; ladder leading to purse (lower level)
3C08:  00 B3 58 B3 78   ; ladder leading to kong's level
3C0D:  00 9B 80 9B A0   ; ladder to right of top right elevator
3C12:  00 93 38 93 58   ; ladder leading up to girl
3C17:  00 23 88 23 C0   ; long ladder on left side
3C1C:  00 1B C0 1B E8   ; bottom left ladder
3C21:  02 97 38 68 38   ; girder girl is on
3C26:  02 B7 58 10 58   ; kong's girder
3C2B:  02 EF 68 E0 68   ; girder where purse is
3C30:  02 D7 70 C8 70   ; girder to left of purse
3C35:  02 BF 78 B0 78   ; girder holding ladder that leads up to kong's level
3C3A:  02 A7 80 90 80   ; girder to right of top right elevator
3C3F:  02 67 88 48 88   ; top girder for central ladder section between elevators
3C34:  02 27 88 10 88   ; girder that holds the umbrella
3C39:  02 EF 90 C8 90   ; girder under the girder that has the purse
3C4E:  02 A7 A0 98 A0   ; bottom girder for section to right of top right elevator
3C53:  02 BF A8 B0 A8   ; small floating girder
```

```
3C58:  02 D7 B0 C8 B0   ; small girder
3C5D:  02 EF B8 E0 B8   ; small girder
3C62:  02 27 C0 10 C0   ; girder just above mario start
3C67:  02 EF D0 D8 D0   ; small girder on far right bottom
3C6C:  02 67 D0 50 D0   ; bottom girder for central ladder section between elevators
3C71:  02 CF D8 C0 D8   ; small girder
3C76:  02 B7 E0 A8 E0   ; small girder
3C7B:  02 9F E8 88 E8   ; floating girder where the right side elevator gets off
3C80:  02 27 E8 10 E8   ; girder where mario starts
3C85:  02 EF F8 10 F8   ; long bottom girder (mario dies if he gets that low)
3C8A:  AA               ; end code
```

Met groen aangegeven onderdelen zijn de standaard onderdelen.

Standaard structuur voor de definitie is:

- Normal: X1 – X2 – X3 – X4 – YT – YB
- Mirror: FF
- End: AA

Omzetten:

```
3C05: 1B 1B 1B 1B C0 E8          - bottom left ladder (#3C1C)
3C0B: 53 53 53 53 88 D0          - center ladder left (#3BF4)
3C11: 63 63 63 63 88 D0          - center ladder right (#3BEA)
3C17: 9B 9B 9B 9B 80 A0          - ladder to right of top right elevator (#3C0D)
;
3C1D: 23 23 23 23 88 C0          - long ladder on left side (#3C17)
3C23: B3 AB B3 08 58 78          - ladder leading to kong's level (#3C08)
3C29: B3 B3 B3 B3 A8 E0          - far top right ladder leading to purse (#3BF9) – moved down
3C2F: CB D3 CB D3 90 B0          - ladder leading to purse (lower level) (#3C03)
3C35: E3 E3 E3 E3 B8 D0          - far bottom right ladder (#3BFE)
3C3B: 93 93 93 93 38 58          - ladder leading up to girl (#3C12)
3C41: 53 53 53 53 18 58          - kong's ladder (left) (#3BEF)
3C47: 63 63 63 63 18 58          - kong’s ladder (right) (#3BE5)
3C4D: DB DB DB DB 58 70          - additional ladder right side
3C53: AA                         - end code
```

En het vaste deel daarachter:

```
3C54:  02 97 38 68 38   ; girder girl is on
3C59:  02 B7 58 10 58   ; kong's girder
3C5E:  02 EF 70 D8 70   ; girder where purse is
3C63:  02 CF 70 C0 70   ; girder to left of purse
3C68:  02 B0 78 A0 78   ; girder holding ladder that leads up to kong's level
3C6D:  02 9F 80 90 80   ; girder to right of top right elevator
3C72:  02 67 88 48 88   ; top girder for central ladder section between elevators
3C77:  02 27 88 10 88   ; girder that holds the umbrella
3C7C:  02 EF 90 C8 90   ; girder under the girder that has the purse
```

```
3C81:  02 A7 A0 98 A0   ; bottom girder for section to right of top right elevator
3C86:  02 BF A8 B0 A8   ; small floating girder
3C8B:  02 D7 B0 C8 B0   ; small girder
3C90:  02 EF B8 E0 B8   ; small girder
3C95:  02 27 C0 10 C0   ; girder just above mario start
3C9A:  02 EF D0 D8 D0   ; small girder on far right bottom
3C9F:  02 67 D0 50 D0   ; bottom girder for central ladder section between elevators
3CA4:  02 CF D8 C0 D8   ; small girder
3CA9:  02 B7 E0 A8 E0   ; small girder
3CAE:  02 9F E8 88 E8   ; floating girder where the right side elevator gets off
3CB3:  02 27 E8 10 E8   ; girder where mario starts
3CB8:  02 EF F8 10 F8   ; long bottom girder (mario dies if he gets that low)
3CBD: 02 EF 58 C8 58    ; additional girder top right
3CC2:  AA               ; end code
```

Was al rekening gehouden met jump voor afhandeling tekenen elevators:

Aanpassen:

```
0920    CA5B13      JP      Z, 135B         ; yes, jump to code that handles elevators
```

en:

```
135B    CDA70D      CALL    #0DA7           ; draw the screen
;
135E    DD21053C    LD      IX,#3C05        ; load IX with pointer to input data structure
1362    21006B      LD      HL,#6B00        ; load HL with pointer to output data structure
;
1365    CD210C      CALL    #0C21           ; build datastructure random ladders
1368    11006B      LD      DE,#6B00        ; load DE with start of datastructure random ladders
136B    CDA70D      CALL    #0DA7           ; draw the random ladders
136E    C9          RET                     ; return
```

en:

```
0CFA    11543C      LD      DE,#3C54        ; load DE with start of table data for the elevators
```

Dat gaat goed.

Nu nog de ladders activeren.

Aanpassen:

```
2468  21006B        LD      HL,#6B00         ; load HL with start of table data for elevators
```

Gaat goed.

Nog wel iets doen aan de rechter fireball. Die blijft namelijk bovenin omdat hij eenmaal op Kong's girder niet mee naar beneden mag en daardoor wordt het wel vrijwel onmogelijk. Hoe dit te voorkomen? Of ergens anders laten beginnen?

De ladder onder de purse naar beneden verplaatsen en daar de fireball ook laten beginnen. Dan krijg je dat hij in de weg gaat zitten als je de staal loop doet (bijvoorbeeld als je de top shelf niet haalt). En dan het platform met de purse één naar beneden en de purse ook meeverplaatsen. En dan wellicht het gat in de Kong girder groter maken?

De aanpassingen in het board zijn in de bovenstaande definitie groen gearceerd.

De purse #08 naar beneden verplaatsen:

```
; bonus items for elevators

3E48  5B 73 0A C8        ;  hat at 5B,C8
3E4C  E3 74 0A 68        ;  purse at E3,68
3E50  1B 75 0A 80        ;  umbrella on elevator is 80,1B
```

De rechter fireball naar andere platform (lager en iets naar links) verplaatsen:

```
; set up 2nd fireball

1101  DD362001      LD      (IX+#20),#01    ; set fire active
1105  DD3623BB      LD      (IX+#23),#BB    ; set fire X position
1109  DD362EBB      LD      (IX+#2E),#BB    ; set fire X position
110D  DD3625A0      LD      (IX+#25),#A0    ; set fire Y position
1111  DD362FA0      LD      (IX+#2F),#A0    ; set fire Y position
```

Nu versie v0.20 opgeslagen.

Ja dat gaat nu goed. Nog wel steeds de mogelijkheid dat je top shelve moet doen. Maar dan werkt omlopen via de staal loop niet omdat je dan weer op hetzelfde stuk uitkomt waar je niet verder kunt. Wellicht de verplaatste trap ook op de originele plaats laten? Maar dan wellicht makkelijk om altijd de staal loop te doen?

Aanpassen:

```
3C05: 1B 1B 1B 1B C0 E8      - bottom left ladder (#3C1C)
3C0B: 53 53 53 53 88 D0      - center ladder left (#3BF4)
3C11: 63 63 63 63 88 D0      - center ladder right (#3BEA)
3C17: 9B 9B 9B 9B 80 B0      - ladder to right of top right elevator (#3C0D)
;
3C1D: 23 23 23 23 88 C0      - long ladder on left side (#3C17)
3C23: B3 AB B3 08 58 78      - ladder leading to kong's level (#3C08)
3C29: B3 B3 B3 B3 A8 E0      - far top right ladder leading to purse (#3BF9) – moved down
3C2F: DB DB DB 08 90 B8      - ladder leading to purse (lower level) (#3C03)
3C35: E3 E3 E3 E3 B8 D0      - far bottom right ladder (#3BFE)
```

```
3C3B: 93 93 93 93 38 58          - ladder leading up to girl (#3C12)
3C41: 53 53 53 53 18 58          - kong's ladder (left) (#3BEF)
3C47: 63 63 63 63 18 58          - kong's ladder (right) (#3BE5)
3C4D: DB DB DB DB 58 70          - additional ladder right side
3C53: E3 E3 E3 08 70 90          - additional ladder right side                -
3C59: AA                         - end code
```

En het vaste deel daarachter (opschuiven door extra ladder):

```
3C5A:  02 97 38 68 38    ; girder girl is on
3C5F:  02 B7 58 10 58    ; kong's girder
3C64:  02 EF 70 D8 70    ; girder where purse is
3C69:  02 CF 70 C0 70    ; girder to left of purse
3C6E:  02 B0 78 A0 78    ; girder holding ladder that leads up to kong's level
3C73:  02 9F 80 90 80    ; girder to right of top right elevator
3C78:  02 67 88 48 88    ; top girder for central ladder section between elevators
3C7D:  02 27 88 10 88    ; girder that holds the umbrella
3C82:  02 EF 90 D8 90    ; girder under the girder that has the purse
3C87:  02 9F B0 90 B0    ; bottom girder for section to right of top right elevator
3C8C:  02 B7 A8 A8 A8    ; small floating girder
3C91:  02 CF B0 C0 B0    ; small girder
3C96:  02 EF B8 D8 B8    ; small girder
3C9B:  02 27 C0 10 C0    ; girder just above mario start
3CA0:  02 EF D0 D8 D0    ; small girder on far right bottom
3CA5:  02 67 D0 50 D0    ; bottom girder for central ladder section between elevators
3CAA:  02 CF D8 C0 D8    ; small girder
3CAF:  02 B7 E0 A8 E0    ; small girder
3CB4:  02 9F E8 88 E8    ; floating girder where the right side elevator gets off
3CB9:  02 27 E8 10 E8    ; girder where mario starts
3CBE:  02 EF F8 10 F8    ; long bottom girder (mario dies if he gets that low)
3CC3:  02 EF 58 C8 58    ; additional girder top right
3CC8:  AA                ; end code
```

En ook de rechter fireball naar links verplaatsen:

```
1101  DD362001      LD      (IX+#20),#01    ; set fire active
1105  DD3623B3      LD      (IX+#23),#B3    ; set fire X position
1109  DD362EB3      LD      (IX+#2E),#B3    ; set fire X position
110D  DD3625A0      LD      (IX+#25),#A0    ; set fire Y position
1111  DD362FA0      LD      (IX+#2F),#A0    ; set fire Y position
```

Door extra ladder, de definitie van het vaste deel ook opgeschoven. Daarom aanpassen:

```
0CFA    115A3C      LD      DE,#3C5A        ; load DE with start of table data for the elevators
```

De purse kan niet gepakt worden. Wellicht teveel naar rechts. Iets naar links verschuiven.

```
3E4C  DB 74 0A 68          ;  purse at DB,68
```

Nu versie v0.21 opgeslagen.

Weer terugzetten naar L=01 en goede volgorde velden.

Aanpassen naar:

095E 01 65 3A 01 00 00 00 ; #3A65 is start of table data for screens/levels

En dan nu de tekst DESIGN BY PAUL GOES gaan toevoegen.

DESIGN BY PAUL GOES

14 15 23 19 17 1E 10 12 29 10 20 11 25 1C 10 17 1F 15 23

```
35ED: 9C 77 05 24 18 10 10 14 15 23 19 17 1E 10 12 29 10 20 11 25 1C 10 17 1F 15 23 10 3F 00 00 43 00 FC 76
```

Dat gaat goed.

Nu versie v0.22 opgeslagen.

Bij elevators kijken of er bepaalde ladders variabel gemaakt kunnen worden qua plaatsing:

De volgende ladders staan altijd aan de rechterkant van een segment, maar mogen ook aan de linkerkant staan.

3C17: 9B 93 9B 93 80 B0 - ladder to right of top right elevator (#3C0D) ----
3C29: B3 A8 B3 A8 A8 E0 - far top right ladder leading to purse (#3BF9) – moved down

Dat is nu goed.

Nu versie v0.23 opgeslagen.

Nu kijken of de sub-levels weergegeven kunnen worden.

Sub level aanduiding is ook gedaan in DK Twisted Jungle. Daar kijken hoe gedaan.

L=01-1 / L=01-2 / L=02-1 / L=02-2 / L=02-3 / L=03-1 / L=03-2 / L=03-3 / L=03-4 / … / …

Waar wordt de L=xx getekend? Dat gebeurt hier.

```
06D7  210375    LD      HL,#7503        ; load HL with screen location for "L="
06DA  361C      LD      (HL),#1C        ; draw "L"
06DC  21E374    LD      HL,#74E3        ; next location
06DF  3634      LD      (HL),#34        ; draw "="
06E1  3A2962    LD      A,(#6229)       ; load A with level #
06E4  fe64      CP      #64             ; level < #64 (100 decimal) ?
06E6  3805      JR      c,#06Ed         ; yes, skip next 2 steps

06E8  3E63      LD      A,#63           ; otherwise A := #63 (99 decimal)
06Ea  322962    LD      (#6229),A       ; store into level #

06Ed  010Aff    LD      BC,#ff0A        ; B: = #FF, C := #0A (10 decimal)

06f0  04        INC     b               ; increment B
06f1  91        SUB     c               ; subtract 10 decimal
06f2  d2f006    JP      NC,#06f0        ; not carry, loop again (counts tens)

06f5  81        ADD     A,C             ; add 10 back to A to get a number from 0 to 9
06f6  32A374    LD      (#74A3),A       ; draw level to screen (low byte)
06f9  78        LD      A,b             ; load a with b (number of tens)
06fa  32C374    LD      (#74C3),A       ; draw level to screen (high byte)
06fd  c9        RET                     ; return
```

622E is het aantal Goofy Kongs dat getekend moet worden: dat is een indicatie voor het schermnummer binnen een level.

Aanpassing:

```
06FA  C36F13        JP      #136F       ; jump to additional code – jump to label AAAA
```

en

```
136F  32C374        LD      (#74C3),A   ; draw level to screen (high byte) – label AAAA
;
1372  3E2C          LD      A,#2C       ; load A with "-"
1374  328374        LD      (#7483), A  ; draw hyphen
;
1377  3A2962        LD      A,(#6229)   ; load A with level number
137A  FE00          CP      #00         ; is the level number 0?
;
137C  2804          JR      Z, #3820    ; yes, skip next lines
137E  3A2E62        LD      A,(#622E)   ; load A with number of Goofy Kongs to draw
1381  3C            INC     A           ; increment A
1382  326374        LD      (#7463), A  ; draw screen in level number
;
1385  C9            RET                 ; return
```

Dit werkt nog niet omdat de variabele #622E niet geupdate. Dat zat in het stuk bij het tekenen van de Goofy Kongs en dat is eruit gehaald.

Bij finish level girders, conveyors and elevators: #622E met 1 ophogen.
Bij finish level rivets #622E weer op 0 zetten.

Finish level girders, conveyors and elevators laatste step is code vanaf #178E.

```
17B0  3630      LD      (HL),#30        ; set timer to #30
17B2  23        INC     HL              ; HL := GameMode2
17B3  3608      LD      (HL),#08        ; set game mode2 to 8
17B5  C9        RET                     ; return
```

Aanpassen:

```
17B3  C38613        JP      #1386       ; jump to additional code, jump to label AAAA
```

en

```
1386  3608           LD      (HL),#08    ; set game mode2 to 8 – label AAAA
;
1388  212E62         LD      HL, #622E   ; load HL with variable
138B  34             INC     (HL)        ; increase variable
;
138C  C9             RET                 ; return
```

Finish level rivets is code vanaf #18C6.

```
1937  3E6f      LD      A,#6f           ; else A := #6F
1939  32206A    LD      (#6A20),A       ; store A into heart sprite X position
193C  c9        RET                     ; return from sub
```

Aanpassen:

```
1939   C3A113        JP      #13A1       ; jump to additional code, jump to label AAAA
```

en

```
13A1  32206A         LD      (#6A20),A   ; store A into heart sprite X position – label AAAA
;
13A4  3E00           LD      A,#00       ; A := 00
13A6  322E62         LD      (#622E),A   ; store A in variable
;
13A9  C9             RET                 ; return
```

Nu versie v0.24 opgeslagen.

Nu eventjes kijken wat er nodig is om het level en sublevel te registreren bij een nieuwe entry in de high score list. Niet heel veel ruimte over, maar wellicht net voldoende.

De tabelregel wordt initieel gevuld met veertien spaties:

```
; sets #61B7 through #61C4 to #10 (???)

13EB  060E      LD      B,#0E           ; for B = 1 to #E

13ED  3610      LD      (HL),#10        ; store #10 into memory at (HL)
13EF  23        INC     HL              ; next HL
13F0  10FB      DJNZ    #13ED           ; next B
```

Nu dat niet doen. Eerst twee spaties, dan vier posities met het level en sub-level en dan nog acht spaties.

```
13EB  3610          LD      (HL),#10        ; store #10 into memory at (HL)
13ED  23            INC     HL              ; next HL
13EE   C3C93C       JP      #3CC9           ; jump to additional code – jump to label AAAA
13F1  00            NOP

en:

3CC9  3610          LD      (HL),#10        ; store #10 into memory at (HL) – label AAAA
3CCB  23            INC     HL              ; next HL
3CCC  23            INC     HL
;
3CCD   3A2962       LD      A,(#6229)       ; load A with level nr
3CD0   010AFF       LD      BC,#FF0A        ; B: = #FF, C := #0A (10 decimal)
;
3CD3   04           INC     B               ; increment B – label CCCC
3CD4   91           SUB     C               ; subtract 10 decimal
3CD5   30FB         JR      NC,#1631        ; not carry, loop again – jump to label CCCC
;
3CD7   81           ADD     A,C             ; add 10 back to A to get a number from 0 to 9
3CD8   77           LD     (HL),A           ; store level nr (singles) into memory at (HL)
3CD9   2B           DEC     HL              ; next HL
;
3CDA   78           LD      A,B             ; load A with B (number of tens)
3CDB   77           LD      (HL),A          ; store levelnr (tens) into memory at (HL)
3CDC   23           INC     HL              ; next HL
3CDD   23           INC     HL              ; next HL
;
3CDE   362C         LD      (HL),#2C        ; store hyphen into memory at (HL)
3CD0   23           INC     HL              ; next HL
;
3CD1   3A2E62       LD      A,(#622E)       ; load A with board nr
3CE4   3C           INC     A               ; increment A
3CE5   C3833F       JP      #3F83           ; jump to additional code – jump to label DDDD

en:


3F83   77           LD      (HL),A          ; store board nr into memory at (HL)
3F84   23           INC     HL              ; next HL
;
3F85   0608         LD      B,#08           ; for B = 1 to #08 – label BBBB
;
3F87   3610         LD      (HL),#10        ; store #10 into memory at (HL)
3F89   23           INC     HL              ; next HL
3F8A   10FB         DJNZ    #1640           ; next B – jump back to label BBBB
;
3F8C   C3F213       JP      #13F2           ; jump back
```

Ja dat gaat goed. Alleen wordt de naam er overheen geschreven. Dat nog aanpassen:

```
14CC   11F9FF     LD      DE,#FFF9        ; load DE with offset of -8 (decimal)
14CF    19        ADD     HL,DE           ; add offset
14D0   223A60     LD      (#603A),HL      ; store result into ???

en

15D3   0605         LD      B,#05           ; for B = 1 to #6 (11 decimal)
15D5   21E875       LD      HL,#75E8        ; load HL with screen vram address
```

```
15D8  FD2A3A60  LD     IY,(#603A)     ; load IY with ???
15DC  11E0FF    LD     DE,#FFE0       ; load DE with offset of -#20

15DF  7E        LD     A,(HL)         ; load A with
15E0  FD7700    LD     (IY+#00),A     ; store
15E3  FD23      INC    IY             ; next
15E5  19        ADD    HL,DE          ; add offset
15E6  10F7      DJNZ   #15DF          ; next B
```

Gaat nu allemaal goed. Alleen nog de header erboven aanpassen.

De kop moet dan ook waarschijnlijk aangepast worden:

```
367D:  9E 37            19      ; #379E "RANK SCORE NAME"
3790:  19 10 24 19 1D 15 10 10 30 03 00 31 10 3F 92 77  I.TIME.........
37A0:  22 11 1E 1B 10 10 23 13 1F 22 15 10 10 1E 11 1D  RANK..SCORE..NAM
37B0:  15 10 10 10 10 3F 72 77 29 1F 25 22 10 1E 11 1D  E.......YOUR.NAM
```

Dat past niet. Maar in de 'REGI TIME' ervoor zit extra ruimte. Maar dan past nog niet dus ook andere aanpassing: RANK aanpassen naar een #.

Dan dus de 3F op #379D één positie naar voren en de 9277 op#379E ook.

Dan begint RANK SCORE NAME op #379D, dus dan aanpassen:

367D 9D 37

En dan op de plaats van de N moet een hekje komen. Dus twee opschuiven in beeld: #7792 wordt dan #7752.

```
                #       S  C  O  R  E        L  E  V  E  L       N  A  M  E
#379D: 52 77 FA 10 10 23 13 1F 22 15 10 10 1C 15 26 15 1C 10 10 1E 11 1D 15 3F
```

Ja dat is goed. Alleen nog karakter voor # maken en invullen: #FA.

Ja dat gaat allemaal goed.

Nu versie v0.25 opgeslagen.

Het jaartal staat nog op 2021. Dit aanpassen.

In #3F00 de copyright notice:

© 1981-2022 NINTENDO

5C 77 49 4A 10 01 09 08 01 2C 02 00 02 02 10 1E 19 1E 24 15 1E 14 1F 3F

Nu versie v0.26 opgeslagen.

De tekst HOW HIGH CAN YOU GET ? aanpassen.

Wat is een leuke tekst?

RANDOMIZING THE BOARD !!

36CC: 5E 77 22 11 1E 14 1F 1D 19 2A 19 1E 17 10 24 18 15 10 12 1F 11 22 14 10 37

Nu versie v0.27 opgeslagen.

Tunen. Bij de Rivets de twee middelste, alleen op de twee middelste posities plaatsen en de andere wellicht iets breder maken.

3B35: 73 7B 83 8B 80 A8 - center ladder level 3 (#3C8B) –> 7B 83 7B 83 80 A8
3B3B: 73 7B 83 8B D0 F8 - bottom center ladder (#3C90) –> 7B 83 7B 83 D0 F8
3B41: 53 5B 63 6B A8 D0 - level 2 ladder #2 of 4 (#3CB8) –> 53 63 6B 73 A8 D0
3B47: FF - level 2 ladder #3 of 4 (#3CBD)
;
3B48: 33 43 33 43 58 80 - top left ladder (#3C95)
3B4E: FF - mirror – top right ladder (#3CA4)
3B4F: 53 5B 63 53 58 80 - top left ladder right side (#3C9A)
3B55: FF - mirror – top right ladder left side (#3C9F)
3B56: 2B 3B 2B 3B 80 A8 - level 3 ladder left side (#3CA9)
3B5C: FF - mirror – level 3 ladder right side (#3CAE)
3B5D: 23 33 43 23 A8 D0 - level 2 ladder left side (#3CB3)
3B63: FF - mirror – level 2 ladder right side (#3CC2)
3B64: 1B 2B 3B 1B D0 F8 - bottom left ladder (#3CC7)
3B6A: FF - mirror – bottom right ladder (#3CCC)
3B6B: AA - end code

Bij de elevators geen trappen meer weglaten. Wel de ladder naar Kongs level opnemen in de vier die gebroken kunnen worden.

3C05: 1B 1B 1B 1B C0 E8 - bottom left ladder (#3C1C)
3C0B: 53 53 53 53 88 D0 - center ladder left (#3BF4)
~~3C11: 63 63 63 63 88 D0 - center ladder right (#3BEA)~~
3C11: B3 AB B3 AB 58 78 - ladder leading to kong's level (#3C08)
3C17: 9B 9B 9B 9B 80 B0 - ladder to right of top right elevator (#3C0D)
;
3C1D: 23 23 23 23 88 C0 - long ladder on left side (#3C17)
~~3C23: B3 AB B3 AB 58 78 - ladder leading to kong's level (#3C08)~~
3C23: 63 63 63 63 88 D0 - center ladder right (#3BEA)
3C29: B3 A8 B3 A8 A8 E0 - far top right ladder leading to purse (#3BF9) – moved down
3C2F: DB DB DB DB 90 B8 - ladder leading to purse (lower level) (#3C03)
3C35: E3 E3 E3 E3 B8 D0 - far bottom right ladder (#3BFE)
3C3B: 93 93 93 93 38 58 - ladder leading up to girl (#3C12)
3C41: 53 53 53 53 18 58 - kong's ladder (left) (#3BEF)
3C47: 63 63 63 63 18 58 - kong’s ladder (right) (#3BE5)
3C4D: DB DB DB DB 58 70 - additional ladder right side
3C53: E3 E3 E3 E3 70 90 - additional ladder right side -
3C59: AA - end code

Dat gaat goed nu.

Nu versie v0.28 opgeslagen.

Vreemd. De rechter fireball in het elevators bord gaat heel raar links van de trap naar boven en beneden? Komt dat door de start van de fireball? Of door de definitie van de ladder?

Komt doordat de ladder op B3 en A8 geplaatst kan worden. Maar A8 is geen goede ladder plaatsing.

3C29: B3 A8 B3 A8 A8 E0 - far top right ladder leading to purse (#3BF9) – moved down

Aanpassen:

3C29: B3 AB B3 AB A8 E0 - far top right ladder leading to purse (#3BF9) – moved down

Ja gaat nu wel goed.

Nu versie v0.29 opgeslagen.

De Goofy Kong naar boven verschuiven en eventueel de Goofy Kong twee dobbelstenen in z'n handen geven.

Goofy Kong naar boven verschuiven kan met:

0BFD 21BC75 LD HL,#75BC ; load HL with screen location start for goofy kong

Een regel naar boven is -1. Nu #75B3 van gemaakt: is 9 naar boven.

De RANDOMIZE THE BOARDS text ook dan 9 naar boven.

36CC: 5E 77 22 11 1E 14 1F 1D 19 2A 19 1E 17 10 24 18 15 10 12 1F 11 22 14 10 37

Van 775E naar 7755:

Is nu goed.

Nu versie v0.30 opgeslagen.

De Goofy Kong met de twee dobbelstenen halen uit DK Randomized.

Nu versie v0.31 opgeslagen.

Eventjes checken of kill screen klopt.

Aanpassen naar:

095E 15 73 3A 01 00 00 00 ; #3A65 is start of table data for screens/levels

En invincibility aan.

19B3 CD0828 CALL #2808 ; check for collisions with hostile sprites

Ja kill screen werkt wel, maar dan gaat de weergave van het high score screen niet goed. Ziet er zo uit:

Met een jumpman linksonder en linksboven (half half) en rare nullen in de high score tabel. Is dat alleen bij een kill screen of ook als je bijvoorbeeld doodgaat op level 21-6? En als je op 22-1 doodgaat, maar niet door een kill screen? Of doodgaan door timer op?

Rare is, dat je geeneens de initialen mag ingeven.

Ja gebeurd ook al bij 21-6. Ook bij gewoon doodgaan. Hoe komt dit ? Is dit ook al bij eerdere levels en bij welke dan?

Bij level 05 gaat het nog goed. Bij level 09 gaat het nog goed. Bij level 10 niet meer. Dus daar waarschijnlijk een probleem met het weergeven van het high score scherm wanneer dat tientallen bevat. Kijken naar de code die dat doet en herstellen.

Dit stuk code doet de 10-tallen bepalen:

```
3CCD  3A2962        LD      A,(#6229)     ; load A with level nr
3CD0  010AFF        LD      BC,#FF0A      ; B: = #FF, C := #0A (10 decimal)
```

```
;
3CD3  04              INC     B             ; increment B – label CCCC
3CD4  91              SUB     C             ; subtract 10 decimal
3CD5  30FB            JR      NC,#1631      ; not carry, loop again – jump to label CCCC
```

Maar in de debugger is de JR NC naar #3CD2 maar dat moet #3CD3 zijn. Dus een minder terug springen. Dus niet #FB (terug), maar #FC (terug).

Lijkt nu wel goed te gaan.

Nu versie v0.32 opgeslagen.

Kill screen gaat nu goed inclusief het vastleggen van de initialen en weergave van het level en sublevel.

Nu nog testen met doorlopen van de eerste 6 levels. Klopt de level volgorde?

Weer terugzetten naar L=01.

Aanpassen naar:

```
095E  01 65 3A 01 00 00 00          ; #3A65 is start of table data for screens/levels
```

Invincibility weer uitzetten.

```
19B3  CD0828    CALL    #2808       ; check for collisions with hostile sprites
```

Nu versie v0.33 opgeslagen.

Dit wordt de finale versie v1.00.