Module 5: Plaatjes — **Toepassing: Een punt laten stuiteren**

In deze toepassing van module 5 bouwen we een programma gebaseerd op grafische elementen.

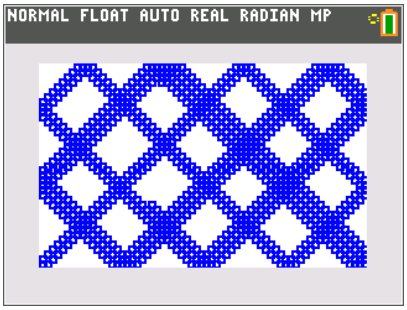
Doelen:

- Weten hoe je kunt bepalen op wat voor rekenmachine een programma wordt uitgevoerd.
- Een punt laten stuiteren over het scherm.

Docenten Tip: Dit is een nogal complex project waarin wat interessante natuurkunde verborgen zit. Het 'deeltje' beweegt door *delta-x* en *delta-y* waarden op te tellen bij de coördinaten tijdens elke iteratie van de lus. Dit zijn de horizontale en verticale componenten van de snelheid. Wanner het deeltje de rand van het scherm raakt, wordt de passende component 'omgekeerd' zodat het deeltje lijkt terug te stuiteren tegen de rand van het scherm.

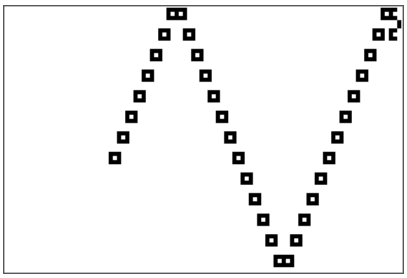
Programma "Pong"

In het originele videospel 'Pong' stuiterde er een pixel over het scherm. De spelers bestuurden de 'batjes' zoals bij pingpong om de bal in het spel te houden. Dit programma zal een stuiterende bal opleveren. Een punt beweegt in een schuine lijn over het scherm en wanneer het een rand raakt zal het van richting veranderen zodat het van de rand terug lijkt te stuiteren.



Het eerste probleem om over na te denken is de schermgrootte, omdat de TI-84 Plus een andere schermgrootte heeft dan de TI-84 C/CE. Het volgende stuk programmeercode zal de schermgrootte bepalen:

```
0→Xmin
1→ΔX                    Ook in het menu [VARS] Venster
If Xmax>95 then
   <het is een C of CE>
Else
   <het is een 84 Plus>
End
```



Hetzelfde programma op twee verschillende rekenmachine.

Op dit punt weten we nu met welke rekenmachine we werken en stellen we de variabelen in in de **Then** en de **Else** blokken om deze te grbuiken in de rest van het programma:

M voor de breedte en **N** voor de hoogte.

Voor de C of CE: **264→M** en **165→N**

Voor de 84 Plus : **94→M** en **63→N**

Variabelen initialiseren

We stellen ook het bereik voor *y* in op mooie waarden:

```
Ø→Ymin
1→ ΔY
```

We beginnen met het punt **(A,B)** op een willekeurige plek, maar niet te dicht bij de rand van het scherm:

```
randInt(1Ø,M-1Ø)→A
randInt(1Ø,N-1Ø)→B
```

We stellen ook twee variabelen in die de beweging voorstellen. Deze zullen worden opgeteld bij de coördinaten van het punt om het punt naar een nieuwe positie te verplaatsen:

```
randInt(2,5)→D        verandering in A
randInt(2,5)→E        verandering in B
```

Docenten Tip: Dit zijn de delta-x en delta-y waarden.

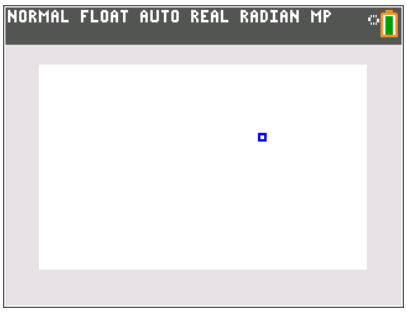
Aan de slag

We zijn klaar om in actie te komen. We gebruiken een oneindige lus …

```
While 1
  Pnt-Aan(A,B,2)          Stijl 2 een grote vierkante stip

    <de rest van het programma>

End
```



Tip: Het is een goed idee om de opdracht **End** *meteen toe te voegen om ze bij te kunnen houden.*

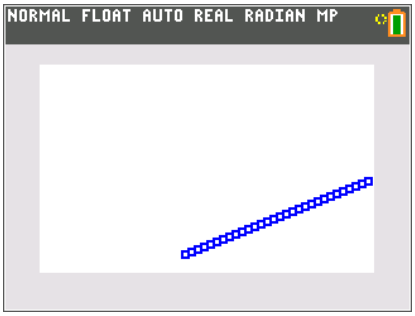
Het laten bewegen

Tel in de lus en na **Pnt-Aan(** de 'veranderingsvariabelen' op bij de coördinaten van het punt:

```
A+D→A
B+E→B
```

Dit verandert de coördinaten van het punt.

Als je het programma nu uitvoert zal je zien dat het punt vertrekt in een bepaalde richting en snel van het scherm afloopt zoals je ziet in het plaatje rechts.



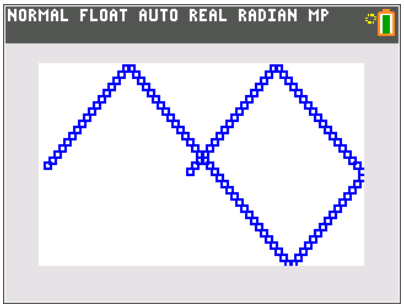
Stuiteren

Om te zorgen dat het punt de randen van het scherm opmerkt voegen we **If**-opdrachten toe…..

```
If A>M of A<Ø
Then

<dit gebeurt wanneer het punt aan de rechter- of linkerkant van het scherm af loopt>

End
```



Krijg jij dit?

Er moeten twee dingen gebeuren binnen de **Then**:

+ het punt terug op scherm plaatsen **A-E→A**
+ De richting veranderen in de tegengestelde **-E→E**

Schrijf een vergelijkbare opdracht om de *y*-coördinaat B en de bijbehorende 'veranderingsvariabele' **E** te behandelen.

Uitbreidingen

Laat geen spoor achter …

De opdracht **Pnt-Uit** zal een punt verbergen. Voeg een opdracht **Pnt-Uit** toe aan je programma zodat het spoor van punten niet wordt getoond, maar alleen het bewegende punt. Pas op: zet niet het punt uit dat je hebt aangezet. Zet het *vorige* punt uit (je hebt hiervoor twee extra variabelen nodig). De nieuwe programma-regels moeten op de juiste plaats in het programma komen.

De oneindige lus kwijtraken …

getKey (in het I/O menu vanp) haalt een variabele op die een 'toets' voorstelt *zonder* het programma te pauzeren net zoals **Prompt** en **Input.** e heeft de waarde 105 (rij 10, kolom 5 op het toetsenbord).

Hier is een 'geraamte van wat er gedaan moet worden:

```
Ø→K
While K≠1Ø5

   <jouw programma komt hier>

getKey→K
End
```

Het is jouw taak om te beslissen waar in het programma deze opdrachten horen zodat het programma stopt wanneer je op e drukt.

Wat dacht je hiervan …?

Als je op CLEAR drukt (welke **getKey** waarde is dat?) start het programma opnieuw met een leeg scherm en met willekeurige waarden en als je op e drukt stopt het.

Voorbeeldantwoord:

```
prgmPONG
FnUit
PlotsUit
AsUit
WisTekenen
Ø     →Xmin
1 ΔX
If Xmax>95
Then
  264  →M
  165  →N
Else
  94  →M
  63  →N
End
Ø     →Ymin
1 ΔY

randInt(1Ø,M-1Ø)  →A
randInt(1Ø,N-1Ø)  →B
randInt(2,5)  →D
randInt(2,5)  →E

While 1
  Pnt-Aan(A,B,2)
  A+D  →A
  B+E  →B
  If A>M of A<Ø
  Then
    A-D  →A
    éD  →D
  End
  If B>N of B<Ø
  Then
    B-E  →B
    éE  →E
  End
End
```