Snake

0 Snake Lessenserie

datasource

Welkom bij de Snake lessenserie! In deze serie leer je stap voor stap hoe je het klassieke spelletje **Snake** maakt met `Python` en `Pygame Zero`. Je begint met het tekenen van een blokje en eindigt met een compleet werkend spel – inclusief groeiende slang, score en Game Over!

? Overzicht van de lessen

1. Les 1 – Teken de slangkop
2. Les 2 – Beweeg de slang
3. Les 3 – Botsing met schermrand
4. Les 4 – Richting automatisch volgen
5. Les 5 – Teken een appel en detecteer botsing
6. Les 6 – Laat de slang groeien
7. Les 7 – Begrijp insert() en pop()
8. Les 8 – Vertraagde beweging
9. Les 9 – Score en Game Over
10. Les 10 – Challenge en uitbreiding

? Leerdoelen

- Je leert werken met de `draw()` en `update()` functies van Pygame Zero
- Je oefent met variabelen, lists en toetsenbordinvoer
- Je leert wat een game loop is en hoe je controle krijgt over beweging

- Je leert je eigen code uitbreiden, testen en verbeteren

?? Benodigdheden

- Thonny geïnstalleerd met de package `pgzero`
- Een beetje basiskennis van Python (variabelen, if, functies)

? Hoe werk je?

Elke les bevat uitleg, voorbeeldcode, opdrachten én een extra uitdaging. Voer je code steeds uit in Thonny en probeer alle opdrachten echt zelf op te lossen. Je mag hulp vragen of AI gebruiken, maar probeer te begrijpen wat er gebeurt.

? Klaar?

Lever je eindspel in, samen met je antwoorden op de reflectievraag. Veel succes en vooral: veel plezier met programmeren!

?? Opdracht

We gaan nog een projectje maken met Thonny (uit de vorige les) en bij dit project gaan we gebruik maken van de standaard Python library:

pgzero

Weet je nog hoe je een package installeert in Thonny?

Yep, Tools - Manage Packages en dan pgzero zoeken en installeren.

? Inleveren

Maak een neiuw bestand en bewaar dit lege bestand op je computer. Maak een screenshot van Thonny met een leeg bestand.

> Tip: Kopieer je vorige project en noem het anders.

Voorbeeld

image.png

1 Teken de slangkop

In deze les gaan we beginnen met het maken van een eigen versie van Snake in Python met Pygame Zero.

We gaan eerst de kop van de slang tekenen op het scherm. Je ziet dan een vierkantje dat straks kan gaan bewegen.

Wat gaan we doen?

We maken een slangkop als vierkant met een bepaalde positie en grootte, en tekenen die in de `draw()`-functie.

? Benodigdheden

- Een werkende Python-omgeving met Pygame Zero (zoals Thonny)
- Geen plaatjes nodig – we tekenen de slang met blokken

? Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 600
HEIGHT = 400

# Startpositie van de slangkop
snake_x = 100
snake_y = 100
tile_size = 20

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")

#start programma
pgzrun.go()
```

?? Uitleg

- `WIDTH` en `HEIGHT`: grootte van het spelvenster
- `snake_x` en `snake_y`: positie van de slangkop
- `tile_size`: grootte van het blokje
- `screen.draw.filled_rect(...)`: tekent een gevuld vierkantje op het scherm

?? Opdracht

- Pas de waarde van `snake_x` en `snake_y` aan – wat gebeurt er?
- Maak het vierkant groter of kleiner door `tile_size` te wijzigen
- Verander de kleur van het vierkant in bijvoorbeeld `"blue"` of `"orange"`

⬜⬜ Extra uitdaging

Teken een tweede blokje naast de slangkop alsof er al een stukje staart is. Gebruik nog een `screen.draw.filled_rect()`.

? Inleveren

1. Maak een screenshot van je 'slangkop' op het scherm (dus geen code) waarbij je de 'slangkop'duidelijke op een andere positie hebt gezet (bijvoorbeeld rechtsonder in het schrem, of in het midden).

2 Beweeg de slang

In deze les gaan we de slangkop laten bewegen met de pijltjestoets die je indrukt.

We gebruiken daarvoor de `update()`-functie van Pygame Zero en de toetsenbordinput.

Wat gaan we doen?

We maken een richting-variabele en passen de positie van de slang aan op basis van de pijltjes die je indrukt.

? Code

Dit is een **deel van de code**, je moet jouw bestaande code aanpassen aan de hand van deze nieuwe code.

Je hoeft dus niet alles opnieuw te typen. Plaats de `update()` functie en pas eventueel `snake_x`, `snake_y`, `tile_size` en `step` aan in jouw bestaande code.

```
# Startpositie van de slangkop
snake_x = 200
snake_y = 200
tile_size = 10
step = 1

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")

def update():
    global snake_x, snake_y

    if keyboard.left:
        snake_x -= step
    if keyboard.right:
        snake_x += step
    if keyboard.up:
        snake_y -= step
    if keyboard.down:
        snake_y += step
```

?? Uitleg

- `update()` wordt automatisch meerdere keren per seconde uitgevoerd
- `keyboard.left` controleert of de linkerpijl is ingedrukt
- Telkens als je een toets indrukt, verandert de positie van de slang
- `step` bepaalt hoe ver de slang per stap beweegt

?? Opdracht

- Beweeg de slang door het scherm met de pijltjes

- Pas `step` aan naar een andere waarde – wat merk je?
- Laat de slang sneller of langzamer bewegen door minder of meer pixels per keer te verplaatsen

◫ Extra uitdaging

Laat de slang automatisch blijven bewegen in de laatst gekozen richting:

- Gebruik een variabele `richting` die je bijwerkt met `on_key_down()`
- Laat de slang dan elke update in die richting verder bewegen

? Inleveren

1. Leg uit wat de variable step doet.
2. Welke waarde heb je gekozen, waarom?

(je kunt dit inleveren in het tekst veld of in een .txt. bestandje)

3 Automatische beweging

In deze les gaan we de slang automatisch laten blijven bewegen in de richting van de laatste pijltjestoets die je hebt ingedrukt.

? Wat gaan we doen?

- We maken een nieuwe variabele `richting`.
- Als je op een pijltjestoets drukt, verandert de waarde van `richting`.
- In de `update()` functie verplaatst de slang zich elke keer opnieuw in de gekozen richting – ook als je de toets niet ingedrukt houdt!

? Code

Gebruik deze versie als nieuwe code (je mag dit toevoegen aan of combineren met je bestaande code):

```
import pgzrun

WIDTH = 600
HEIGHT = 400

snake_x = 200
snake_y = 200
tile_size = 20
step = 10
richting = "right"

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")

def update():
    global snake_x, snake_y, richting

    if richting == "left":
        snake_x -= step
    elif richting == "right":
        snake_x += step
    elif richting == "up":
        snake_y -= step
    elif richting == "down":
        snake_y += step

def on_key_down(key):
    global richting

    if key == keys.LEFT:
        richting = "left"
    elif key == keys.RIGHT:
        richting = "right"
    elif key == keys.UP:
        richting = "up"
    elif key == keys.DOWN:
        richting = "down"
```

```
pgzrun.go()
```

?? Uitleg

- `richting`: deze variabele onthoudt de laatst gekozen richting
- `on_key_down()`: dit is een functie die wordt uitgevoerd als je op een toets drukt
- `update()`: deze functie verplaatst de slang elke keer in de gekozen richting, ook als je geen toets indrukt

?? Opdracht

- Test of je slang automatisch blijft bewegen nadat je een pijl indrukt
- Wat gebeurt er als je twee keer snel op een andere richting drukt?
- Probeer met `tile_size` en `step` te spelen voor een ander effect

□□ Extra uitdaging

Voeg grenzen toe aan je spel: laat de slang stoppen of ergens anders naartoe gaan als hij de rand van het scherm raakt.

? Inleveren

1. Leg in je eigen woorden uit wat de functie `on_key_down()` doet

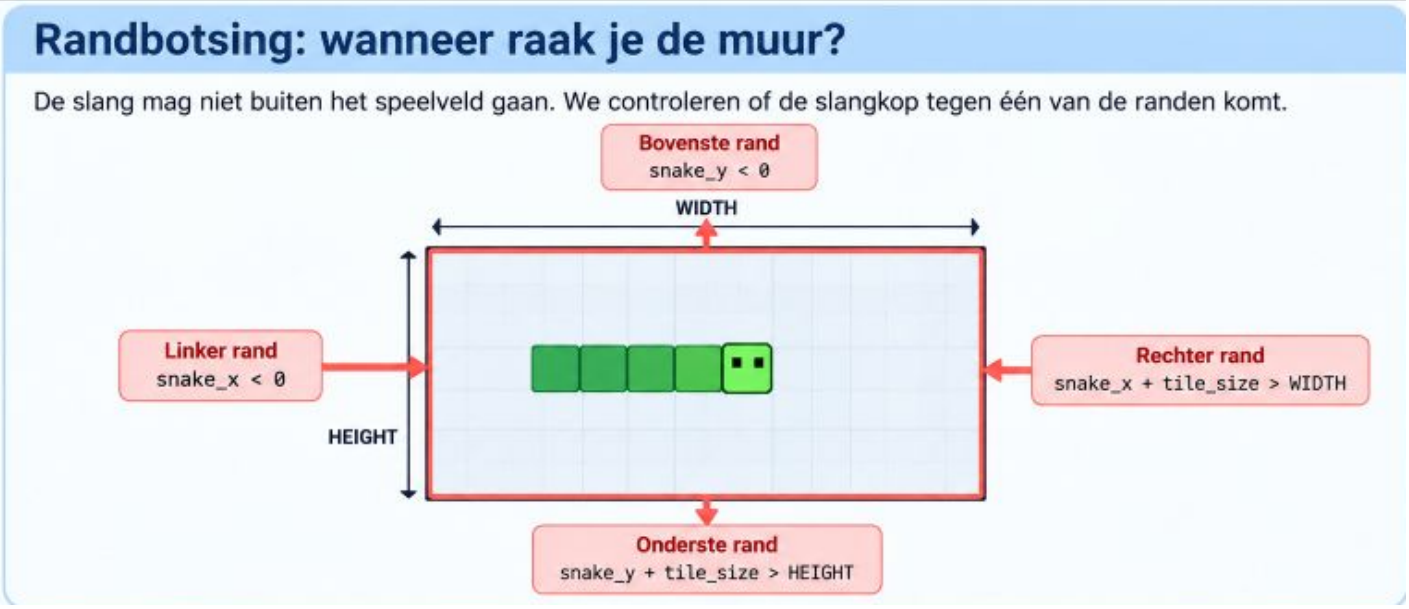
(Lever dit in via het tekstvak of via een upload.)

4 Randbotsing en automatische richting

In deze les zorgen we ervoor dat de slang niet zomaar het scherm verlaat. Zodra hij een rand raakt, verandert hij automatisch van richting. Zo beweegt hij steeds binnen het scherm!

? Wat gaan we doen?

- We controleren of de slang de rand van het scherm raakt.
- Als dat zo is, passen we automatisch de richting aan.



`snake_x` en `snake_y` geven de positie van de **linkerbovenhoek van de slangkop** weer. De kop zelf heeft ook een breedte en hoogte van `tile_size`.

? Code (let op: onvolledig!)

Hieronder zie je de code. De code is onvolledig en moet worden aangepast.

> Hint: kijk naar de ... (drie puntjes)

```
import pgzrun

WIDTH = 600
HEIGHT = 400

snake_x = 200
snake_y = 200
tile_size = 20
step = 5
richting = "right"

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")

def update():
    global snake_x, snake_y, richting
```

```
    if richting == "left":
        snake_x -= step
    elif richting == "right":
        snake_x += step
    elif richting == "...":
        snake_y -= step
    elif richting == "...":
        snake_y += step

    # □□Hier controleren we of de slang de rand raakt
    if snake_x < 0:
        richting = "right"
    elif snake_x + tile_size > WIDTH:
        richting = "left"
    elif snake_y < 0:
        richting = "..."
    elif snake_y + tile_size > HEIGHT:
        richting = "..."

def on_key_down(key):
    global richting

    if key == keys.LEFT:
        richting = "left"
    elif key == keys.RIGHT:
        richting = "right"
    elif key == keys.UP:
        richting = "up"
    elif key == keys.DOWN:
        richting = "down"

pgzrun.go()
```

Als je de linker- of rechterkant raakt, dan 'stuitert' je terug. Dat gebeurt niet als je de boven- of onderkant aanraakt. Pas de code aan zodat je ook naar boven en beneden kan bewegen en zodat je ook terugstuiters als je de onder-of bovenkant aanraakt.

?? Uitleg

- `snake_x + tile_size > WIDTH`: betekent dat de slang voorbij de rechterrand gaat
- Als de slang een rand raakt, verander je de variabele `richting`

?? Opdracht

- Pas de code aan zodat de slang alle richtingen uit kan bewegen en zodat hij als hij de rand raakt (link, recht en boven en beneden) terug 'stuiterd'.
- Test of de slang de alle richtingen op gaat en hij aan alle kantne goed terug 'stuiterd'!

◫ Extra uitdaging

Laat de slang bij elke randbotsing van kleur veranderen! (Tip: maak een variabele `kleur` en gebruik bijvoorbeeld `random.choice()`)

? Inleveren

1. Leg in **eigen woorden** uit **wat** je hebt aangepast op regel 35 en 36 en **hoe dat werkt.**

Vind je dit lastig? Dan *kun* je de deze template gebruiken:

```
Ik heb op regel 35 .... aangepast en ik heb op regel 36 .... aangepast.
Wat er dan gebeurt is dat als de Snake ......... en de toets ..... wordt ingedrukt dan
verandert de .......... van .......... naar ............
```

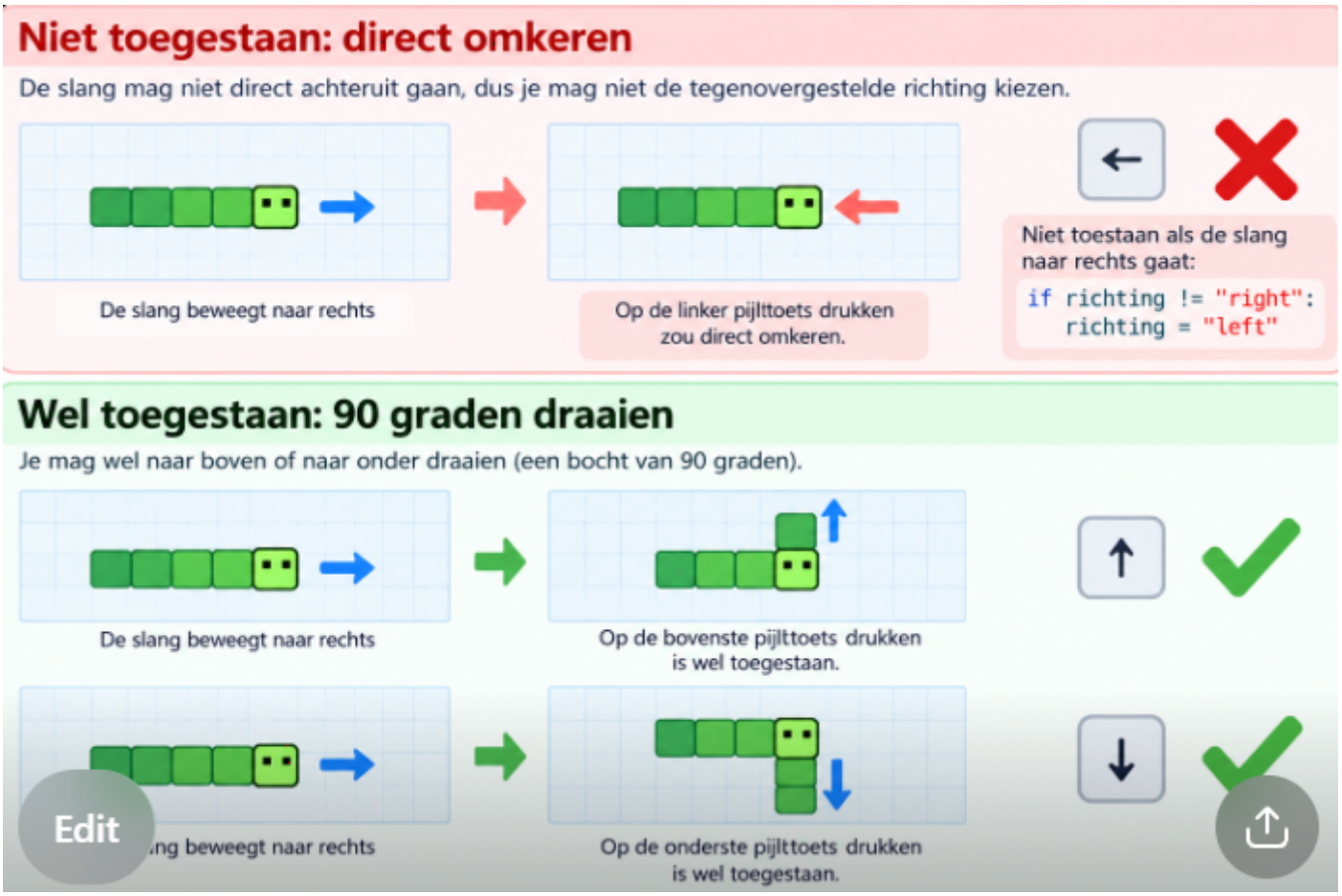
(Lever dit in via het tekstvak of via een upload.)

5 Geen omkeren toegestaan

In deze les zorgen we ervoor dat de slang niet meteen omkeert. In een echte Snake-game kun je namelijk niet ineens van rechts naar links bewegen – dan zou de slang zichzelf opeten!

? Wat gaan we doen?

- We voorkomen dat de slang direct de tegenovergestelde richting kiest.
- We vergelijken de huidige richting met de nieuwe richting voordat we die veranderen.



? Code (let op: onvolledig!)

De volgende code voorkomt omkeren, maar jij moet één voorwaarde nog zelf aanvullen.

```
import pgzrun

WIDTH = 600
HEIGHT = 400

snake_x = 200
snake_y = 200
tile_size = 20
step = 1
richting = "right"

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")

def update():
```

```
    global snake_x, snake_y, richting

    if richting == "left":
        snake_x -= step
    elif richting == "right":
        snake_x += step
    elif richting == "up":
        snake_y -= step
    elif richting == "down":
        snake_y += step

def on_key_down(key):
    global richting

    if key == keys.LEFT and richting != "right":
        richting = "left"
    elif key == keys.RIGHT and richting != "left":
        richting = "right"
    elif key == keys.UP and richting != "down":
        richting = "up"
    elif key == keys.DOWN and richting != "...":
        richting = "down"

pgzrun.go()
```

Stuiteren tegen randen werkt niet meer

Het botsen tegen de randen wat in stap 4 is gedaan, zit niet in deze code. Dat is express gedaan. We proberen op deze manier een ding gelijktijdig uit te leggen en het voorkomt dat je op dit moment "*door de bomen het bos niet meer ziet*".

?? Uitleg

- We gebruiken een `if`-voorwaarde om te voorkomen dat je teruggaat in de tegenovergestelde richting.
- De ontbrekende regel is die voor `keys.DOWN`: welke richting is dan niet toegestaan?

?? Opdracht

- Vul de ontbrekende voorwaarde aan zodat de slang niet van "up" naar "down" mag keren
- Test alle richtingen: kun je nog steeds normaal draaien? Keren lukt niet meer, toch?

◫ Extra uitdaging

Laat de slang een geluidje maken als je op een toets drukt, maar de richting wordt niet veranderd (bijv. als je wél op links drukt, maar dat mag niet).

Met deze code kan je ene geluidje afspelen.

```
sounds.beep.play()
```

Om een sound af te kunnen spelen moet jouw project folder een mapje maken sounds en daarin moet een beep.wav komen.

```
mijn_project/
├── main.py
└── sounds/
    └── beep.wav
```

Geluidjes toevoegen

Je kunt zelf geluidjes toevoegen, stel je hebt een geluidje **bel.wav** dan plaats je dat in de sounds directory en dan gebruik je het commando

`souds.bel.play()`

? Inleveren

1. Leg uit **wat** je hebt veranderd en **hoe** het werkt (dit zijn dus twee vragen: **wat** en **hoe**).

Vind je dit lastig? Dan *kun* je de deze template gebruiken:

```
Ik heb op regel ...     ..... aangepast.
Wat er dan gebeurt is dat als de Snake ......... en de toets ..... wordt ingedrukt dan
.........
```

(Lever dit in via het tekstvak of via een upload.)

6 Appel tekenen en raken

In deze les voegen we een appel toe aan het spel. De appel verschijnt op een willekeurige plek op het scherm. Als de slang de appel raakt, verschijnt er een bericht in de console.

? Wat gaan we doen?

- We gebruiken de `random`-module om een appel op een willekeurige positie te tekenen.
- We tekenen de 'appel' met een geel cirkeltje of vierkantje.
- We detecteren of de slang de appel raakt.

? Benodigdheden

- Je slangkop beweegt al automatisch over het scherm
- Je hebt al gewerkt met `tile_size` en `snake_x`/`snake_y`

? Code (let op: onvolledig!)

Onderstaande code tekent een slangkop en een appel. De botsing werkt al, maar na een botsing blijft de appel op dezelfde plek staan. Vul zelf aan wat er moet gebeuren!

```
import pgzrun
import random

WIDTH = 600
HEIGHT = 400

tile_size = 20
snake_x = 100
snake_y = 100
richting = "right"
step = 3

# Appelpositie (willekeurig op het grid)
apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size
```

```
def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((snake_x, snake_y), (tile_size, tile_size)), "green")
    screen.draw.filled_circle((apple_x + tile_size // 2, apple_y + tile_size // 2), tile_size
// 2, "yellow")

def on_key_down(key):
    global richting

    if key == keys.LEFT and richting != "right":
        richting = "left"
    elif key == keys.RIGHT and richting != "left":
        richting = "right"
    elif key == keys.UP and richting != "down":
        richting = "up"
    elif key == keys.DOWN and richting != "up":
        richting = "down"

def update():
    global snake_x, snake_y, apple_x, apple_y, richting

    if richting == "left":
        snake_x -= step
    elif richting == "right":
        snake_x += step
    elif richting == "up":
        snake_y -= step
    elif richting == "down":
        snake_y += step

    # Botst de slang met de appel?
    snake_rect = Rect((snake_x, snake_y), (tile_size, tile_size))
    apple_rect = Rect((apple_x, apple_y), (tile_size, tile_size))
    if snake_rect.colliderect(apple_rect):
        print("□□Appel geraakt!")
        # □□VUL HIER AAN: genereer een nieuwe positie voor de appel
        # apple_x = ...
        # apple_y = ...
```

```
pgzrun.go()
```

?? Uitleg

- `random.randint()`: geeft een willekeurig geheel getal terug
- De appelpositie wordt op het grid berekend, zodat deze altijd netjes uitlijnt
- `filled_circle()`: tekent een geel rondje (je kunt ook `filled_rect()` gebruiken)

?? Opdracht

- Test of de appel op het scherm verschijnt
- Beweeg de slang naar de appel – zie je de console-uitvoer?
- Vul de ontbrekende regels in zodat de appel na een botsing op een **nieuwe plek** verschijnt

◻◻ Extra uitdaging

Laat de appel niet precies op dezelfde plek als de slang verschijnen wanneer hij opnieuw wordt gegenereerd!

? Inleveren

1. Lever de code in .py bestand.

7 De slang groeit

In deze les maak je van je slang een échte slang: eentje die uit meerdere blokjes bestaat en langer wordt als hij een appel eet.

? Wat gaan we doen?

- We veranderen de slang van één blokje naar een lijst van blokjes
- De slang groeit bij het eten van een appel
- We houden de lengte gelijk als er geen appel wordt gegeten

? Strategie

De code wordt nu iets ingewikkelder. Het is niet erg als je niet alles direct begrijpt. Probeer wel te volgen wat er gebeurt. We bespreken de strategie:

1. De slang bestaat uit meerdere delen. De code in de `draw()`-functie tekent elk stukje.
2. Bij elke update wordt er een nieuw blokje toegevoegd aan de kop van de slang.
3. Als de slang een appel raakt: dan blijft de slang langer.
4. Als er geen appel is geraakt: dan wordt het laatste stukje van de slang verwijderd.

Dus in het kort:

1. Teken de slang
2. Voeg een stukje toe aan de kop
3. Geen appel? Verwijder de staart

ChatGPT Image 3 Oct 2026, 18_04_34.jpg

? Code

```
import pgzrun
import random

WIDTH = 600
HEIGHT = 400
tile_size = 20

snake = [(100, 100)]
richting = "right"
step = 3

apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size
```

```
def draw():
    screen.clear()
    for segment in snake:
        screen.draw.filled_rect(Rect(segment, (tile_size, tile_size)), "green")
    screen.draw.filled_circle((apple_x + tile_size // 2, apple_y + tile_size // 2), tile_size
// 2, "yellow")

def on_key_down(key):
    global richting
    if key == keys.LEFT and richting != "right":
        richting = "left"
    elif key == keys.RIGHT and richting != "left":
        richting = "right"
    elif key == keys.UP and richting != "down":
        richting = "up"
    elif key == keys.DOWN and richting != "up":
        richting = "down"

def update():
    global apple_x, apple_y, richting

    head_x, head_y = snake[0]
    if richting == "left":
        head_x -= step
    elif richting == "right":
        head_x += step
    elif richting == "up":
        head_y -= step
    elif richting == "down":
        head_y += step

    new_head = (head_x, head_y)
    snake.insert(0, new_head)

    head_rect = Rect(new_head, (tile_size, tile_size))
    apple_rect = Rect((apple_x, apple_y), (tile_size, tile_size))

    if head_rect.colliderect(apple_rect):
        print("□□Appel geraakt!")
```

```
        apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
        apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size
    else:
        snake.pop()  # staart verwijderen

    if new_head in snake[1:]: # Deze code detecteerd of de slang zichzelf raakt.
        print("□□Game over! De slang raakte zichzelf.")
        exit()

pgzrun.go()
```

?? Uitleg

- `snake = [(x, y), ...]`: lijst van alle slangsegmenten
- `insert(0, new_head)`: voegt een nieuw blokje toe aan de voorkant
- `pop()`: verwijdert het laatste stukje (staart)
- `if new_head in snake[1:]`: controle of slang zichzelf raakt

```
Voorbeeld bij beweging:

Stap 1: slang = [(4,2), (3,2), (2,2)]
Stap 2: kop wordt (5,2) → slang = [(5,2), (4,2), (3,2)]
Stap 3: geen appel → staart eraf → slang = [(5,2), (4,2)]
```

□□ Waarom werkt dit zo?

Door altijd eerst een kop toe te voegen, beweegt de slang vooruit. Als er geen appel is geraakt, halen we de staart weg zodat de lengte gelijk blijft. Wordt er wél een appel geraakt, dan blijft de slang langer: we doen dan geen `pop()`. Dit is hoe de slang groeit!

?? Opdracht

- Controleer of de slang groeit als hij een appel eet
- Controleer of hij even lang blijft als er geen appel wordt geraakt

? Inleveren

1. Beantwoord de vraag: **Wat gebeurt er als je `snake.pop()` vergeet?**

(Lever dit in via het tekstvak of als bestand.)

8 De slang sneller maken met vertraging

In deze les leer je hoe je de slang sneller kunt laten groeien door de `step` te vergroten, zonder dat het spel te snel en onspeelbaar wordt.

? Wat is het probleem?

Je denkt misschien: "Ik wil dat de slang sneller beweegt, dus ik maak `step` groter." Maar als je `step = 20` doet, wordt het spel ineens **veel te snel**.

Dat komt omdat `update()` standaard 60 keer per seconde wordt uitgevoerd. Dus je slang maakt dan 60 sprongen van 20 pixels per seconde: dat is 1200 pixels!

? Fout idee

```
step = 20  # grotere stap
# maar het spel gaat nu te snel!
```

? Goede oplossing: beweging vertragen

We laten de slang **niet elke update** bewegen, maar bijvoorbeeld 1x per 10 frames. Zo kun je `step` groter maken, zonder dat het spel onbestuurbaar wordt.

? Code

```
import pgzrun
import random

WIDTH = 600
HEIGHT = 400
```

```
tile_size = 20
step = 20
vertraging = 10  # aantal frames wachten
frames = 0

snake = [(100, 100)]
richting = "right"

apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size

def draw():
    screen.clear()
    for segment in snake:
        screen.draw.filled_rect(Rect(segment, (tile_size, tile_size)), "green")
    screen.draw.filled_circle((apple_x + tile_size // 2, apple_y + tile_size // 2), tile_size
// 2, "yellow")

def on_key_down(key):
    global richting
    if key == keys.LEFT and richting != "right":
        richting = "left"
    elif key == keys.RIGHT and richting != "left":
        richting = "right"
    elif key == keys.UP and richting != "down":
        richting = "up"
    elif key == keys.DOWN and richting != "up":
        richting = "down"

def update():
    global frames
    frames += 1
    if frames < vertraging:
        return
    frames = 0
    beweeg_slang()

def beweeg_slang():
    global apple_x, apple_y, richting
```

```
    head_x, head_y = snake[0]
    if richting == "left":
        head_x -= step
    elif richting == "right":
        head_x += step
    elif richting == "up":
        head_y -= step
    elif richting == "down":
        head_y += step

    new_head = (head_x, head_y)
    snake.insert(0, new_head)

    head_rect = Rect(new_head, (tile_size, tile_size))
    apple_rect = Rect((apple_x, apple_y), (tile_size, tile_size))

    if head_rect.colliderect(apple_rect):
        print("\U0001F34F Appel geraakt!")
        apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
        apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size
    else:
        snake.pop()

    if new_head in snake[1:]:
        print("\u274C Game over! De slang raakte zichzelf.")
        exit()

pgzrun.go()
```

? Waarom werkt dit?

- `frames += 1`: telt hoe vaak `update()` al is uitgevoerd
- Pas als `frames >= vertraging`, beweegt de slang echt
- `step` bepaalt nu de grootte van de stap, niet de snelheid van het spel

? Opdracht

- Stel `step = 20` in en test of de slang nu sneller groeit
- Experimenteer met `vertraging = 5` of `vertraging = 15`
- Wat is een fijne balans tussen stapgrootte en snelheid?
- Test het spel; wat gebeurt er precies als de slang tegen zijn eigen staart botst?

⬜⬜ Extra uitdaging

Laat de slang naarmate hij langer wordt automatisch steeds iets sneller bewegen (tip: verlaag `vertraging` bij elke appel).

? Inleveren

1. Leg uitwat er gebeurt als de slang tegen zijn staart aankomt.
2. Probeer te vertellen wat je zou willen veranderen; wat moet er gebeuren als de slang tegen zijn slang aan botst?

(Lever dit in via het tekstvak of als bestand.)

9 Score bijhouden en Game Over tonen

In deze les voegen we een score toe aan het spel. Elke keer als je een appel eet, krijg je 1 punt. En als de slang zichzelf raakt, laten we **"Game Over"** op het scherm zien en stopt het spel.

? Wat gaan we doen?

- Een variabele `score` bijhouden
- De score op het scherm tonen met `screen.draw.text()`
- Bij een botsing met jezelf: `game_over = True`
- Het spel stoppen en "Game Over" tonen

? Startercode

```
import pgzrun
import random

WIDTH = 600
```

```
HEIGHT = 400
tile_size = 20
step = 20
vertraging = 10
frames = 0

snake = [(100, 100)]
richting = "right"
score = 0

apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size

game_over = False

def draw():
    screen.clear()
    if game_over:
        screen.draw.text("GAME OVER", center=(WIDTH // 2, HEIGHT // 2), fontsize=60,
color="red")
        screen.draw.text(f"Score: {score}", center=(WIDTH // 2, HEIGHT // 2 + 50),
fontsize=40, color="white")
    else:
        for segment in snake:
            screen.draw.filled_rect(Rect(segment, (tile_size, tile_size)), "green")
        screen.draw.filled_circle((apple_x + tile_size // 2, apple_y + tile_size // 2),
tile_size // 2, "yellow")
        screen.draw.text(f"Score: {score}", topleft=(10, 10), fontsize=30, color="white")

def on_key_down(key):
    global richting
    if key == keys.LEFT and richting != "right":
        richting = "left"
    elif key == keys.RIGHT and richting != "left":
        richting = "right"
    elif key == keys.UP and richting != "down":
        richting = "up"
    elif key == keys.DOWN and richting != "up":
        richting = "down"
```

```
def update():
    global frames
    if game_over:
        return
    frames += 1
    if frames < vertraging:
        return
    frames = 0
    beweeg_slang()

def beweeg_slang():
    global apple_x, apple_y, score, game_over, richting

    head_x, head_y = snake[0]
    if richting == "left":
        head_x -= step
    elif richting == "right":
        head_x += step
    elif richting == "up":
        head_y -= step
    elif richting == "down":
        head_y += step

    new_head = (head_x, head_y)

    if new_head in snake[1:]:
        print("□ Game over! De slang raakte zichzelf.")
        game_over = True
        return

    snake.insert(0, new_head)

    head_rect = Rect(new_head, (tile_size, tile_size))
    apple_rect = Rect((apple_x, apple_y), (tile_size, tile_size))

    if head_rect.colliderect(apple_rect):
        print("\U0001F34F Appel geraakt!")
        score += 1
        apple_x = random.randint(0, (WIDTH - tile_size) // tile_size) * tile_size
        apple_y = random.randint(0, (HEIGHT - tile_size) // tile_size) * tile_size
```

```
    else:
        snake.pop()


pgzrun.go()
```

? Waarom werkt dit?

- We tekenen de score linksboven tijdens het spel
- We zetten `game_over = True` bij een zelfbotsing
- Als `game_over` waar is, stoppen we de beweging en tonen een ander scherm

? Opdracht

- Laat de slang een paar appels eten en kijk of de score klopt
- Laat de slang zichzelf raken en controleer of "Game Over" verschijnt

□□ Extra uitdaging

- Laat de score ook in de console printen bij elke appel
- Laat het spel automatisch herstarten na een paar seconden (of met een toets)

? Inleveren

1. Beantwoord: **Hoe weet de code dat het "Game Over" is en wat gebeurt er dan precies?**
   Leg stap-voor-stap uit wat er gebeurt. Kijk goed naar de code!

(Lever dit in via het tekstvak of als bestand.)

10 Snake – Eindopdracht

?? Houd de slang binnen het scherm

Op dit moment kan de slang zomaar van het scherm verdwijnen als je te ver naar links, rechts, boven of onder beweegt. Bedenk zelf hoe je dat kunt oplossen!

Tip: je kunt controleren of de kop van de slang buiten het scherm terechtkomt. Bijvoorbeeld:

```
if head_x < 0 or head_x >= WIDTH or head_y < 0 or head_y >= HEIGHT:
    game_over = True
```

Op deze manier ben je 'af' als je het scherm raakt. Je zou ook van richting kunnen veranderen, maar dit is wat lastiger om uit te voeren.

?? Opdracht

Zorg ervoor dat de slang niet buiten het beeld kan verdwijnen. Je kunt het natuurlijk zo maken dat als je de rand raakt, het spel klaar is ("Game over").

Een ander optie is om van richting te veranderen. Dat is lastiger, weet je waarom? Hoe zou je het op een goede manier kunnen uitvoeren?

? Inleveren

1. Leg uit waarom het lastig is om de slang van richting te laten veranderen als die de rand van het scherm raakt. Je hoeft het niet te coderen, maar leg uit hoe je de slang precies van richting zou kunnen laten veranderen als die de rand raakt. En werkt dit ook in de hoeken?

   *Probeer te omschrijven hoe het zou kunnen werken. maak als het ware een **AI prompt** (een AI opdracht) om de code op een bepaalde manier aan te passen. Beschrijf **precies** wat er moet gebeuren!*

□□ Extra uitdaging

Als je goed hebt beschreven wat er precies moet gebeuren en je hebt ook rekening gehouden met de hoeken dan kun je misschien AI vragen jouw te helpen om dit echt te bouwen? Probeer maar ! Als het lukt, wordt het spel leuker om te spelen!

--

Revision #44
Created 2025-05-28 07:30:35 UTC by Max
Updated 2026-10-03 16:14:05 UTC by Max