Vallende stenen

Cursus: 28567

0 Wat gaan we leren

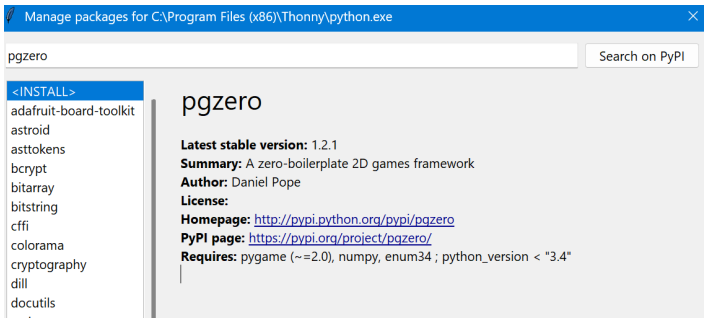
datasource

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.



?? Opdracht

Waarvoor dient **pgzero**?

Hint: toen je de package opzocht in Thonny zag je een omschrijving.

? Inleveren

Korte omschrijving in één of twee zinnen waarvoor jij denkt dat je het package **pgzero** heb moeten installeren.

1 Teken speler en steen

In deze les gaan we de speler en één vallende steen tekenen.

We beginnen met een eenvoudige versie: een blokje onderaan dat je straks kunt besturen, en een steen die we straks laten vallen.

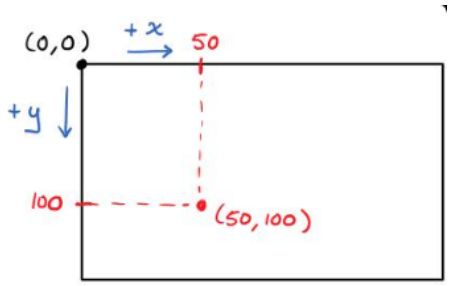
Wat gaan we doen?

We tekenen een rechthoek voor de speler en een rechthoek voor de steen.

We gebruiken vaste posities om de eerste versie werkend te krijgen.

Weet je nog hoe de posities in Pygame werken?

(0,0) staat linkbovenaan, de eerste coordinaat laat zien hoe ver je naar rechts gaat en de tweede how ver je naar bedene gaat.



Pygame Coordinates

? Code

```
import pgzrun

WIDTH = 800
HEIGHT = 600

# Speler onderaan het scherm
player_x = 400
player_y = 550
player_width = 80
player_height = 20

# Vallende steen bovenaan
rock_x = 300
rock_y = 0
```

```
rock_size = 40

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((player_x, player_y), (player_width, player_height)), "blue")
    screen.draw.filled_rect(Rect((rock_x, rock_y), (rock_size, rock_size)), "gray")

pgzrun.go()
```

?? Uitleg

- `player_x` en `player_y`: positie van de speler (een blauw blokje onderaan)
- `rock_x` en `rock_y`: positie van de steen
- `screen.draw.filled_rect(...)`: tekent een blokje op het scherm

?? Opdracht

- Verplaats de steen naar een andere plek op het scherm door `rock_x` en `rock_y` aan te passen
- Maak de speler breder of smaller
- Verander de kleuren van speler en steen

?? Extra uitdaging

- Teken meerdere stenen op het scherm (gebruik meerdere `draw.filled_rect()`)

? Inleveren

1. Maak een screenshot waarop je de speler en minstens één steen ziet (waarbij je de plaats van de steen dus hebt aangepast).

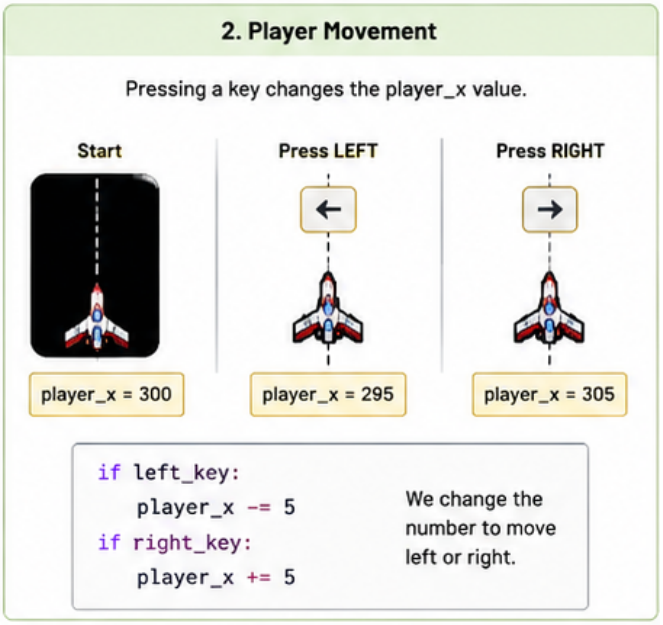
2 Speler bewegen

In deze les gaan we de speler besturen met de pijltjestoetsen.

De speler beweegt alleen naar links en rechts, en mag niet buiten het scherm gaan.

Wat gaan we doen?

We bewerken de `update()`-functie om de `x`-positie van de speler aan te passen als je op pijltjes drukt.



We voegen een maximale en minimale positie toe zodat de speler niet van het scherm glijdt.

? Code

```
import pgzrun

WIDTH = 800
HEIGHT = 600

player_x = 400
player_y = 550
player_width = 80
player_height = 20
player_speed = 5

rock_x = 300
rock_y = 0
rock_size = 40

def draw():
    screen.clear()
```

```
    screen.draw.filled_rect(Rect((player_x, player_y), (player_width, player_height)), "blue")
    screen.draw.filled_rect(Rect((rock_x, rock_y), (rock_size, rock_size)), "gray")

def update():
    global player_x

    if keyboard.left:
        player_x -= player_speed
    if keyboard.right:
        player_x += player_speed

    # Speler binnen scherm houden
    if player_x < 0:
        player_x = 30
    if player_x > WIDTH - player_width:
        player_x = WIDTH - player_width

pgzrun.go()
```

?? Uitleg

- `player_speed`: hoe snel de speler beweegt
- `keyboard.left` en `keyboard.right`: detecteren of een toets is ingedrukt
- `if player_x > WIDTH - player_width`: voorkomt dat de speler buiten beeld schuift

?? Opdracht

- Beweeg de speler heen en weer met je pijltjestoetsen
- Verander `player_speed` – wordt de speler sneller of trager?
- Aan de linker kant van het scherm stuitert de speler terwijl aan de rechterkant de speler netjes op de rand stopt. Zie jij hoe dat komt? Probeer dit aan te passen, probeer gewoon maar wat, je kan niets kapot maken!

> Tip: bij welke `if` wordt gekeken of de speler tegen de linkerkant van het scherm aan zit?

?? Extra uitdaging

- Laat de speler sneller bewegen als je de toets langer inhoudt
- Laat de speler automatisch naar links of rechts bewegen als je een extra toets indrukt (bijvoorbeeld `A` of `D`)

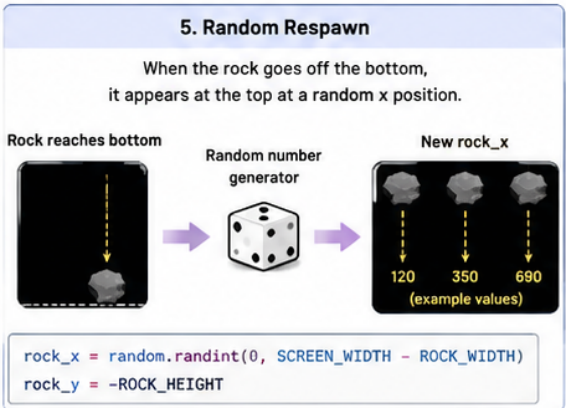
? Inleveren

1. Leg in eigen woorden uit hoe de `player_speed` de snelheid van het spel veranderd.
2. Leg uit waarom de speler aan de linkerkant van het scherm 'stuitert' en wat heb je aangepast om dit te voorkomen?

3 Steen laten vallen

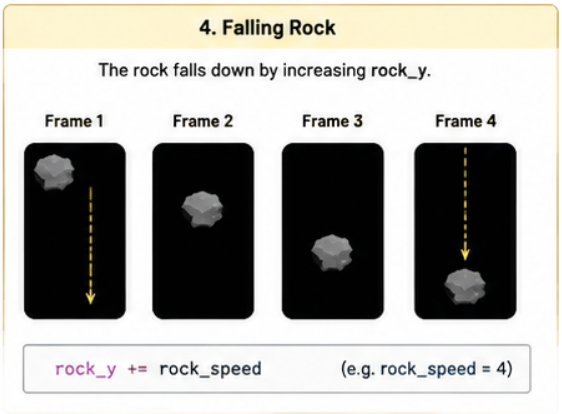
In deze les gaan we de steen automatisch laten vallen.

Zodra de steen de onderkant van het scherm bereikt, verschijnt hij opnieuw bovenaan op een willekeurige plek.



Wat gaan we doen?

We voegen een `rock_speed` toe en laten de `y`-positie van de steen langzaam toenemen in `update()`.



We controleren of de steen het scherm uit valt, en zetten hem dan opnieuw bovenaan met een willekeurige x-positie.

? Code

```
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

player_x = 400
player_y = 550
player_width = 80
player_height = 20
player_speed = 5

rock_x = random.randint(0, WIDTH - 40)
rock_y = 0
rock_size = 40
rock_speed = 3

def draw():
    screen.clear()
    screen.draw.filled_rect(Rect((player_x, player_y), (player_width, player_height)), "blue")
    screen.draw.filled_rect(Rect((rock_x, rock_y), (rock_size, rock_size)), "gray")

def update():
    global player_x, rock_y, rock_x, rock_size
```

```
    if keyboard.left:
        player_x -= player_speed
    if keyboard.right:
        player_x += player_speed

    if player_x < 0:
        player_x = 0
    if player_x > WIDTH - player_width:
        player_x = WIDTH - player_width

    # Steen laten vallen
    rock_y += rock_speed

    # Als de steen onderaan is, zet hem weer bovenaan met random x
    if rock_y > HEIGHT:
        rock_y = 0
        rock_x = random.randint(0, WIDTH - rock_size)

pgzrun.go()
```

?? Uitleg

- `rock_speed`: bepaalt hoe snel de steen valt
- `rock_y += rock_speed`: laat de steen naar beneden bewegen
- `random.randint(...)`: zorgt voor een willekeurige x-positie als de steen opnieuw verschijnt

?? Opdracht

- Verander de `rock_speed` – wat gebeurt er?
- Maak de steen groter of kleiner door `rock_size` aan te passen.
- Zorg er nu voor dat elke keer als de steen opnieuw valt de rock_size wordt aangepast en een random grootte krijgt.
- Met random.randint(1, 10) wordt er een getal tussen 1 en 10 gegenereerd. Bedenk zelf mooie waarden en pas de code aan zodat de grootte van de steen telkens anders wordt.

□□ Extra uitdaging

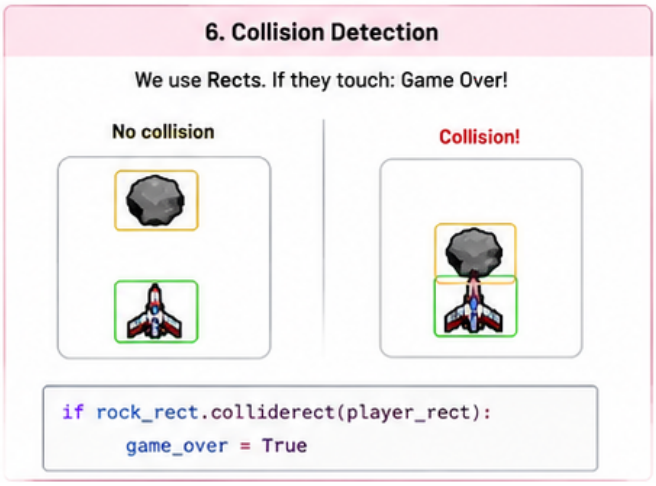
- Laat meerdere stenen tegelijk vallen
- Laat elke steen een willekeurige snelheid hebben

? Inleveren

1. Leg uit hoe je ervoor hebt gezorgdt dat de steen telkens een ander grootte kijgt.

4 Botsing & Game Over

In deze les gaan we controleren of de speler de steen raakt.



Als er een botsing is, stopt het spel en verschijnt er de tekst “Game Over”.

Wat gaan we doen?

We maken een `Rect` van de speler en van de steen, en gebruiken `colliderect()` om te zien of ze elkaar raken.

We gebruiken een variabele `game_over` om te stoppen met het spel als er een botsing is.

? Code

```
import pgzrun
import random
from pygame import Rect

WIDTH = 800
```

```
HEIGHT = 600

player_x = 400
player_y = 550
player_width = 80
player_height = 20
player_speed = 5

rock_x = random.randint(0, WIDTH - 40)
rock_y = 0
rock_size = 40
rock_speed = 3

game_over = False

def draw():
    screen.clear()
    if game_over:
        screen.draw.text("GAME OVER", center=(WIDTH // 2, HEIGHT // 2), fontsize=60,
color="red")
        return

    screen.draw.filled_rect(Rect((player_x, player_y), (player_width, player_height)), "blue")
    screen.draw.filled_rect(Rect((rock_x, rock_y), (rock_size, rock_size)), "gray")

def update():
    global player_x, rock_y, rock_x, game_over

    if game_over:
        return

    if keyboard.left:
        player_x -= player_speed
    if keyboard.right:
        player_x += player_speed

    if player_x < 0:
        player_x = 0
    if player_x > WIDTH - player_width:
```

```
        player_x = WIDTH - player_width

    rock_y += rock_speed

    if rock_y > HEIGHT:
        rock_y = 0
        rock_x = random.randint(0, WIDTH - rock_size)

    # Botsing detecteren
    speler_rect = Rect(player_x, player_y, player_width, player_height)
    steen_rect = Rect(rock_x, rock_y, rock_size, rock_size)

    if speler_rect.colliderect(steen_rect):
        game_over = True

pgzrun.go()
```

?? Uitleg

- `Rect(x, y, w, h)` maakt een rechthoek op de juiste plek
- `colliderect()` kijkt of twee rechthoeken elkaar raken
- `game_over` bepaalt of het spel nog doorgaat

?? Opdracht

- Laat de speler de steen raken en kijk of het spel stopt
- Verplaats de speler naar de andere kant van het scherm – bots je dan nog?
- Verander de “GAME OVER” tekst (bijvoorbeeld kleur of grootte)

?? Extra uitdaging

- Laat het spel herstarten als je op de `R`-toets drukt
- Speel een geluid af bij de botsing (bijv. `sounds.hit.play()`)

? Inleveren

1. Leg eersts stap-voor-stap in he eigen woorden uit hoe er een botsing van de steen met de speler wordt gedetecteerd.
2. Leg daarna stap-voor-stap, in eigen woorden uit wat er allemaal gebeurt als de steen tegen de speler aan komt. Verwijs daarbij naar de code.

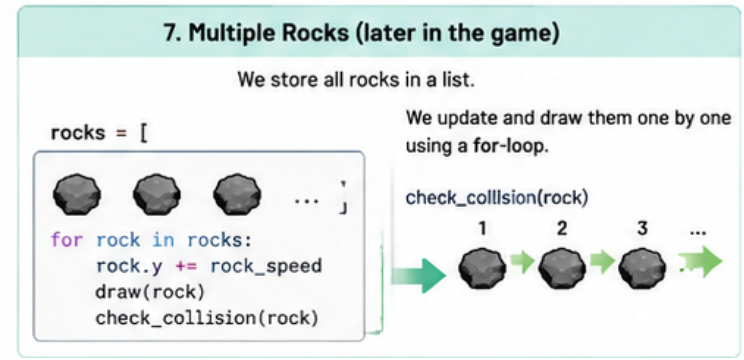
5 Meerdere stenen & moeilijker maken

In deze les gaan we meerdere stenen tegelijk laten vallen.

Bovendien maakt het spel zichzelf moeilijker naarmate je langer speelt: de stenen vallen sneller.

Wat gaan we doen?

We maken een lijst van stenen, elk met hun eigen positie en snelheid.



We laten deze stenen vallen en bij een botsing stoppen we het spel.

We laten het spel steeds moeilijker worden door de snelheid te verhogen na een paar seconden.

? Code

```
import pgzrun
import random
from pygame import Rect

WIDTH = 800
HEIGHT = 600

player_x = 400
player_y = 550
```

```
player_width = 80
player_height = 20
player_speed = 5

rocks = []
game_over = False
rock_timer = 0
rock_interval = 60  # frames
difficulty = 1.0

# Start met een paar stenen
for _ in range(3):
    x = random.randint(0, WIDTH - 40)
    speed = random.uniform(2, 4)
    rocks.append({"x": x, "y": 0, "size": 40, "speed": speed})

def draw():
    screen.clear()
    if game_over:
        screen.draw.text("GAME OVER", center=(WIDTH // 2, HEIGHT // 2), fontsize=60,
color="red")
        return

    screen.draw.filled_rect(Rect((player_x, player_y), (player_width, player_height)), "blue")
    for rock in rocks:
        screen.draw.filled_rect(Rect((rock["x"], rock["y"]), (rock["size"], rock["size"])),
"gray")

def update():
    global player_x, game_over, rock_timer, difficulty

    if game_over:
        return

    if keyboard.left:
        player_x -= player_speed
    if keyboard.right:
        player_x += player_speed
```

```
    player_x = max(0, min(WIDTH - player_width, player_x))

    speler_rect = Rect(player_x, player_y, player_width, player_height)

    for rock in rocks:
        rock["y"] += rock["speed"] * difficulty

        if rock["y"] > HEIGHT:
            rock["y"] = 0
            rock["x"] = random.randint(0, WIDTH - rock["size"])
            rock["speed"] = random.uniform(2, 5)

        rock_rect = Rect(rock["x"], rock["y"], rock["size"], rock["size"])
        if speler_rect.colliderect(rock_rect):
            game_over = True

    # Verhoog de moeilijkheid langzaam
    rock_timer += 1
    if rock_timer % 300 == 0:
        difficulty += 0.2

pgzrun.go()
```

?? Uitleg

- Elke steen is een dict met `x`, `y`, `size` en `speed`
- We gebruiken een `for`-loop om alle stenen te laten vallen
- Met `difficulty` verhogen we langzaam de snelheid van de stenen



?? Opdracht

- Test het spel en kijk of het spel moeilijker wordt na ongeveer 15 seconden!
- Nu gaan we de grootte van de stenen weer aanpassen naar een random waarde zoals we dat bij de vorige opdracht ook hebben gedaan.

  Zorg er dus voor dat bij het aanmaken van de stenen iedere steen een andere (random) grootte krijgt.

  Dat gaat iets anders omdat we nu meerdere 'rocks' hebben. In de `for rock in rocks:` loop worden één voor één alle blokken behandeld. De grootte van de rock staat in `rock["size"]`

> Tip: weet je nog dat je met `random.randint(1,4)` een getal tussen 1 en 4 kan genereren?

◫ Extra uitdaging

- Laat het aantal stenen toenemen naarmate het spel langer duurt
- Laat een andere kleur steen verschijnen bij hogere moeilijkheid.
- Laat elke steen een ander formaat hebben
- Toon je “score” op het scherm: hoe lang heb je overleefd?

? Inleveren

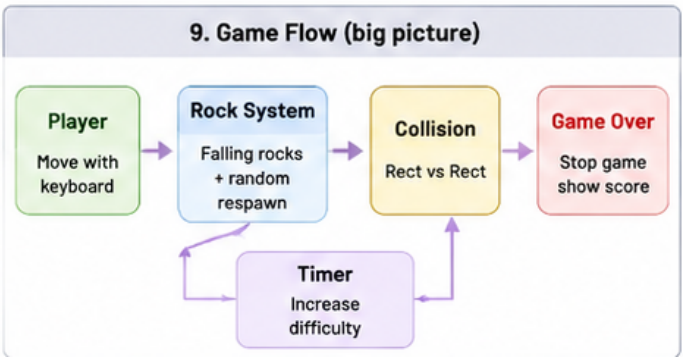
1. Lever je code (.py bestand) in.

6 Score, Reflectie & Uitbreiding

In deze les voeg je een score toe die laat zien hoelang je het hebt volgehouden.

Daarnaast kijk je terug op je eigen werk én kies je een uitbreiding om het spel leuker of moeilijker te maken.

Game flow



Wat gaan we doen?

- We voegen een `score` toe die telt hoeveel frames je overleeft
- We tonen deze score linksboven op het scherm
- Bij Game Over laten we je eindscore zien

? Toevoegingen aan bestaande code

```
score = 0  # bij de andere variabelen

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="black")
        return
    ...
    screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color="black")

def update():
    global score
    if game_over:
        return
    ...
    score += 1
```

?? Uitleg

- `score` wordt bij elke frame 1 hoger → hoe langer je speelt, hoe hoger je score
- `screen.draw.text()` toont je score tijdens én na het spel

? Reflectie

Beantwoord de volgende vragen onderaan je code als commentaar of in een apart document:

- Wat vond je het leukst om te maken in dit spel?
- Wat vond je moeilijk, en hoe heb je dat opgelost?
- Wat heb je geleerd over Python en programmeren?
- Waar ben je trots op?

? Uitbreiding (kies er één)

- □□ Laat een explosie zien of geluid horen bij Game Over
- □□ Voeg bewegende obstakels toe
- □□ Houd een highscore bij (hoogste score van de sessie)
- □□ Geef de speler power-ups: tijdelijk onsterfelijk, versnellen, etc.
- □□ Verander het uiterlijk van de speler of achtergrond na elke 30 seconden

□□ Eigen idee?

Bedenk zelf een uitbreiding en probeer deze te maken. Schrijf kort op wat jouw idee is en hoe je het hebt aangepakt.

? Inleveren

- Lever je eindversie van het spel in (inclusief score en uitbreiding)
- Voeg een korte beschrijving toe van wat je hebt toegevoegd of aangepast
- Optioneel: voeg een screenshot of kort filmpje toe van je spel

7 Reflectieopdracht

Deze reflectieopdracht helpt je om stil te staan bij wat je hebt geleerd tijdens het programmeren van je spel.

Beantwoord de onderstaande vragen eerlijk en in je eigen woorden. Je mag je antwoorden inleveren via een apart document of onderaan je Python-bestand als commentaar zetten.

? Vragen

- Wat vond je het leukste om te maken of uit te proberen?
- Wat vond je lastig? Hoe heb je dat opgelost?
- Wat heb je geleerd over Python (of programmeren in het algemeen)?
- Waar ben je trots op in jouw eindresultaat?
- Wat zou je de volgende keer anders willen doen of verbeteren?

? Inleveren

Lever je reflectie in bij je docent via de afgesproken manier (document, upload of als commentaar in je code).

###

 # Docenten

? Docentenhandleiding

? Overzicht

- **Doelgroep:** Leerlingen met basiskennis Python (13-16 jaar)
- **Duur:** 6 lessen van ±45-60 minuten
- **Software:** Thonny + Pygame Zero
- **Spelconcept:** De speler ontwijkt vallende objecten (stenen) en het spel wordt steeds moeilijker

? Leerdoelen

- Werken met coördinaten en schermlogica
- Gebruik van `draw()` en `update()`
- Toetsenbordbesturing met `keyboard.left` en `keyboard.right`
- Beweging simuleren met variabelen
- Objecten detecteren met `Rect` en `colliderect()`
- Werken met lijsten voor meerdere objecten
- Reflecteren op het leerproces en zelf uitbreidingen bedenken

? Lesoverzicht

Les	Onderwerp	Nieuwe concepten
1	Speler en steen tekenen	`draw()`, rechthoek tekenen, coördinaten
2	Speler beweegt	`update()`, keyboard input, begrenzen
3	Steen valt en komt terug	Beweging, `random`, logica
4	Botsing + Game Over	`Rect`, `colliderect()`, boolean vlag
5	Meerdere stenen + moeilijkheid	Lijsten, `for`-loops, moeilijkheidsschaal
6	Score, Reflectie & Uitbreiding	Scorevariabele, `draw.text()`, vrije opdracht

?? Valkuilen

- Speler glijdt van het scherm → `if player_x > WIDTH - breedte` vergeten
- Botsing werkt niet → verkeerde waarden in `Rect()`
- Alle stenen bewegen tegelijk, maar maar één wordt gecheckt op botsing
- Score telt door na game over → geen check op `if not game_over:`

? Differentiatie

Voor snelle leerlingen

- Voeg power-ups toe (onzichtbaarheid, levens, vertraging)
- Laat speler en stenen met sprites tekenen in plaats van rechthoeken
- Laat leerlingen zelf een nieuw spelelement verzinnen

Voor langzamere leerlingen

- Laat maar één steen vallen (geen lijst)
- Geef per les kant-en-klare startercode mee
- Werk samen in tweetallen

? Beoordeling (optioneel)

Criterium	Omschrijving	Score (1-5)
Werkt het spel	Geen fouten, spel werkt zoals bedoeld	◫◫
Code is leesbaar	Logische structuur, goede naamgeving	◫◫
Uitbreiding toegevoegd	Creatieve of technische aanvulling	◫◫
Reflectie	Antwoorden zijn volledig en met inzicht	◫◫

? Tips voor in de les

- Laat leerlingen na elke wijziging op F5 drukken → directe feedback is motiverend
- Gebruik klassikale demo's bij fouten: "Waarom crasht deze versie?"
- Laat leerlingen zelfstandig kleine uitbreidingen testen vanaf les 5
- Bespreek reflectievragen klassikaal in les 6

? Benodigdheden

- Thonny geïnstalleerd (met Pygame Zero via `pip install pgzero` indien nodig)
- Werkende computer (Windows, Linux of macOS)
- Toetsenbord met pijltjestoetsen

Revision #24
Created 2025-05-21 20:39:46 UTC by Max
Updated 2026-07-08 15:17:53 UTC by Max