


Pygame installeren

De website van pygame vind je op <https://www.pygame.org>

Je installeert pygame met behulp van pip:

```
python -m pip install -U pygame --user
```

We gaan stapsgewijs een eerste game bouwen. De eerste stap is om te kijken of alles werkt.

 Neem volgend script over. Bij het uitvoeren zou er een zwart scherm met in het midden een witte cirkel tevoorschijn moeten komen.


```
import sys, pygame
from pygame.locals import *

pygame.init()


background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480

screen = pygame.display.set_mode(size)

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()
    screen.fill(background)
    pygame.draw.circle(screen, ball_color, (320, 240), 10)
    pygame.display.flip()
```

 Wat gebeurt er als je met de cijfertjes speelt? Wat betekenen ze?

Beweging

 Vul je script aan zodat de bal begint te bewegen.

```
import sys, pygame
from pygame.locals import *


pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_position = 320, 240

screen = pygame.display.set_mode(size)

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()
    ball_position = ball_position[0] + 2, ball_position[1] + 2
    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_position, 10)
    pygame.display.flip()
```

 Waar is de bal naartoe?

 Wat gebeurt er als je de lijn 'screen.fill(background)' weghaalt?

Binnen het scherm blijven

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_position = 320, 240
ball_speed = 2, 2

screen = pygame.display.set_mode(size)


while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    if ball_position[0] < 0 or ball_position[0] > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_position[1] < 0 or ball_position[1] > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_position = ball_position[0] + ball_speed[0], ball_position[1] + ball_speed[1]

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_position, 10)
    pygame.display.flip()
```

Dit begint al ergens op te lijken! Maar we zijn er nog niet..

 Wat gebeurt er met de snelheid van de bal als je het programma meerdere keren tegelijk start? Waarom heeft dit een effect? Tip: je kan in Visual Studio Code meerdere terminals tegelijk openen door op het plusje naast de dropdown te klikken, en als je het commando copy-paste dat in de Python-terminal staat, zou je het programma een tweede keer moeten kunnen opstarten.

 Wat zou er gebeuren als je je programma op een tragere of snellere computer uitvoert? Wat als je je game over 20 jaar op een toekomstige supercomputer zou willen spelen? Sommige oude games hebben hier wel degelijk last van.. Stel je voor dat we niet meer naar oude films konden kijken of oude boeken niet meer kunnen lezen, omdat onze technologie verbeterd is!

De framerate controleren

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_position = 320, 240
ball_speed = 2, 2

screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

while 1:
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    if ball_position[0] < 0 or ball_position[0] > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_position[1] < 0 or ball_position[1] > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_position = ball_position[0] + ball_speed[0], ball_position[1] + ball_speed[1]

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_position, 10)
    pygame.display.flip()
```

Gebruikersinput

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_position = 320, 240
ball_speed = 2, 2
player_speed = 5
player_position = width / 2, height - 20

screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

while 1:
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    keystate = pygame.key.get_pressed()
    direction = 0
    if keystate[K_RIGHT]:
        direction = 1
    elif keystate[K_LEFT]:
        direction = -1
    player_position = player_position[0] + player_speed * direction, player_position[1]

    if ball_position[0] < 0 or ball_position[0] > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_position[1] < 0 or ball_position[1] > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_position = ball_position[0] + ball_speed[0], ball_position[1] + ball_speed[1]

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_position, 10)
    paddle = pygame.Rect(player_position[0], player_position[1], 60, 10)
    pygame.draw.rect(screen, ball_color, paddle)
    pygame.display.flip()
```

Collision detection - deel 1

We moeten het programma aanpassen zodat de bal kan botsen met de speler. Dit wordt in twee stappen uitgelegd. In de eerste stap wijzigen we de posities van de bal en de speler in bounding boxes, rechthoeken in de spelwereld die de grenzen van de objecten voorstellen. Dat verandert nog niets aan het programma, maar het maakt het makkelijker om nadien collision detection toe te voegen.

Overall waar we in het programma over `ball_position` of `player_position` spraken, gaan we nu over rectangles (`ball_rect`, `player_rect`) spreken.

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_rect = pygame.Rect(320, 240, 10, 10)
ball_speed = 2, 2
player_speed = 5
player_rect = pygame.Rect(width / 2, height - 20, 60, 10)

screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

while 1:
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    keystate = pygame.key.get_pressed()
    direction = 0
    if keystate[K_RIGHT]:
        direction = 1
    elif keystate[K_LEFT]:
        direction = -1
    player_rect.move_ip(player_speed * direction, 0)

    if ball_rect.left < 0 or ball_rect.right > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_rect.top < 0 or ball_rect.bottom > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_rect.move_ip(ball_speed)

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_rect.center, 10)
    pygame.draw.rect(screen, ball_color, player_rect)
    pygame.display.flip()
```

Collision detection - deel 2

Nu kunnen we kijken of de rechthoeken overlappen, en zo ja, de bal terugkaatsen.

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_rect = pygame.Rect(320, 240, 10, 10)
ball_speed = 2, 2
player_speed = 5
player_rect = pygame.Rect(width / 2, height - 20, 60, 10)

screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

while 1:
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    keystate = pygame.key.get_pressed()
    direction = 0
    if keystate[K_RIGHT]:
        direction = 1
    elif keystate[K_LEFT]:
        direction = -1
    player_rect.move_ip(player_speed * direction, 0)

    if ball_rect.colliderect(player_rect):
        ball_speed = (ball_speed[0], -ball_speed[1])
    if ball_rect.left < 0 or ball_rect.right > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_rect.top < 0 or ball_rect.bottom > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_rect.move_ip(ball_speed)

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_rect.center, 10)
    pygame.draw.rect(screen, ball_color, player_rect)
    pygame.display.flip()
```

Game over

Als we de bal niet opvangen, is het game over.

```
import sys, pygame
from pygame.locals import *

pygame.init()

background = 0, 0, 0
ball_color = 255, 255, 255
size = width, height = 640, 480
ball_rect = pygame.Rect(320, 240, 10, 10)
ball_speed = 2, 2
player_speed = 5
player_rect = pygame.Rect(width / 2, height - 20, 60, 10)

screen = pygame.display.set_mode(size)
clock = pygame.time.Clock()

while 1:
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    keystate = pygame.key.get_pressed()
    direction = 0
    if keystate[K_RIGHT]:
        direction = 1
    elif keystate[K_LEFT]:
        direction = -1
    player_rect.move_ip(player_speed * direction, 0)

    if ball_rect.colliderect(player_rect):
        ball_speed = (ball_speed[0], -ball_speed[1])
    if ball_rect.left < 0 or ball_rect.right > width:
        ball_speed = (-ball_speed[0], ball_speed[1])
    if ball_rect.top < 0 or ball_rect.bottom > height:
        ball_speed = (ball_speed[0], -ball_speed[1])
    ball_rect.move_ip(ball_speed)

    screen.fill(background)
    pygame.draw.circle(screen, ball_color, ball_rect.center, 10)
    pygame.draw.rect(screen, ball_color, player_rect)
    pygame.display.flip()
```

Ideetjes

Nu je een basis hebt, kan je het spel beginnen uitbreiden! Hier zijn een aantal mogelijkheden, maar voel je vrij om andere dingen te doen. Kan je..

- de bal tijdens het spel van kleur doen veranderen?
- de bal steeds sneller laten gaan naarmate het spel langer duurt?
- bijhouden hoe vaak je de bal hebt opgevangen, en als score tonen?
- de bal een lichte afwijking geven als hij botst? Hiervoor heb je waarschijnlijk nood aan de 'random' module, zie volgende pagina.
- er een multiplayer-spel van maken, door een tweede speler bovenaan te hebben die met andere toetsen (bvb A en D) speelt?
- het game-over scherm een beetje mooier maken?

Het is met pygame natuurlijk ook mogelijk om graphics te gebruiken en geluiden af te spelen. Meer daarover in de volgende les!

Willekeurige getallen

Bij games heb je regelmatig nood aan willekeurige getallen. Python heeft hiervoor de random-module. Je kan wat experimenteren (in script of in de terminal) met volgende voorbeelden:

```
from random import random, uniform, randint, choice

print("random()")
for i in range(10):
    print(random())

print()
print("uniform(-5, 5)")
for i in range(10):
    print(uniform(-5, 5))

print()
print("randint(1, 10)")
for i in range(10):
    print(randint(1, 10))

print()
print("choice(['a', 'b', 'c'])")
for i in range(10):
    print(choice(['a', 'b', 'c']))
```