Handleiding: Gebruikersinterface met Python

Inhoudsopgave

Gebruikershandleiding PyGame

Algemene mal voor een PyGame venster

De volgende code toont een venster m.b.v. de Pygame module.

```
# Naam programma: .........
# Het programma doet: ........
# Programma gemaakt door: ........
import pygame

# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
WINDOW_SIZE = [1050, 750] #venstergrootte

# -------- FUNCTIE DEFINITIES --------


# -------- PYGAME INITIALISATIE  --------
pygame.init()
screen = pygame.display.set_mode(WINDOW_SIZE) #open scherm
pygame.display.set_caption("Scherm titel") #geef een titel aan het scherm

# -------- HOOFDLOOP VAN HET PROGRAMMA  --------


# -------- AFSLUITING  --------
pygame.quit() #sluit scherm af
```

Scherm coördinaten

Als je scherm 1050 bij 750 is (zoals in dit voorbeeld), dan is:

- (0,0) het coördinaat van de linker bovenhoek.
- (1050,0) het coördinaat van de rechterbovenhoek
- (0,750) het coördinaat van de linkeronderhoek, en
- (1050,750) het coördinaat van de rechteronderhoek.

Algemene mal voor een PyGame spel

Bij een spel zit herhaling, namelijk: zolang de speler niet af is (of je sluit het scherm af door op het kruisje te klikken), krijgt hij weer de beurt. Hiervoor gebruiken we een boolean vlag met de naam *spel_is_afgelopen*. Om te beginnen krijgt *spel_is_afgelopen* de waarde False. Zolang **niet** *spel_is_afgelopen* blijven we doorspelen. Als je op kruisje drukt om het venster af te sluiten, dan wordt *spel_is_afgelopen* op True gezet en stopt het programma.

```python
# Naam programma: .........
# Het programma doet: .........
# Programma gemaakt door: .........
import pygame

# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
WINDOW_SIZE = [1050, 750] # Afmeting van spelscherm instellen (in pixels: [breedte, hoogte])

# -------- FUNCTIE DEFINITIES --------


# -------- PYGAME INITIALISATIE  --------
pygame.init()
screen = pygame.display.set_mode(WINDOW_SIZE) # Spelscherm maken (en opslaan in een variabele screen)
pygame.display.set_caption("Scherm titel") # Titel van spelscherm instellen

clock = pygame.time.Clock()# Zorgt voor verversingssnelheid van het scherm


# -------- HOOFDLOOP VAN HET PROGRAMMA  --------
# Deze boolean laat ons spel straks lopen, totdat er op het kruisje wordt gekikt om af te sluiten
# (deze zet dan spel_is_afgelopen op True)
spel_is_afgelopen = False

while not spel_is_afgelopen:

    # --- Check gebeurtenissen (zoals muiskliks enz.) ---
    for event in pygame.event.get():
        if event.type == pygame.QUIT: # Het kruisje is aangeklikt
            spel_is_afgelopen = True # Het spel moet eindigen dus we zetten spel_is_afgelopen op True
    # --- Teken de graphics ---

    # --- Ververs het beeldschem met de nieuwe graphics ---

    clock.tick(30) # Ververs scherm met 30 frames per seconde
    pygame.display.flip() # Ververs het beeldscherm met de bijgewerkte versie




# -------- AFSLUITING  --------
pygame.quit() #  Sluit het scherm af
```

1. pygameSpelMal.py

Achtergrond van venster aanpassen met een kleur

```python
# --- Teken de graphics ---
screen.fill((255,255,255)) # Maak de achtergrond wit
```

2. vensterAchtergrond.py

Als je een specifieke kleur vaker gaat gebruiken, doe je er goed aan om de kleur als constante te definiëren en daarna te gebruiken:

```
# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
ACHTERGROND_GROEN = ( 10, 254, 10 ) # RGB Kleur groen voor achtergrond

# … <hier is wat code weggelaten >

# --- Teken de graphics ---
screen.fill(ACHTERGROND_GROEN) # Maak achtergrond groen
```

3. vensterAchtergrond.py

Achtergrond van een venster aanpassen met een plaatje

Eerst moet je het plaatje laden. Let er op dat je de juiste extensie gebruikt (bijvoorbeeld .jpg of .png) en dat het plaatje in dezelfde map zit als je python code.

```
# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
# Afbeelding voor achtergrond scherm (spelbord)
achtergrond_plaatje = pygame.image.load("spelbord.jpg")
```

4. vensterAchtergrond.py

Daarna moet je binnen de while-loop de volgende code gebruiken zodat het achtergrond plaatje daadwerkelijk getoond wordt.

```
# --- Teken de graphics ---
# Bepaal afmeting (rectangle van het plaatje)
afmetingBord = achtergrond_plaatje.get_rect()
# Projecteer het plaatje op het scherm
screen.blit(achtergrond_plaatje,afmetingBord)
```

5 vensterAchtergrond.py

Video met uitleg voor het toevoegen van een achtergrond plaatje: http://youtu.be/4YqIKncMJNs

```python
import pygame

# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
ACHTERGROND_PLAATJE = pygame.image.load("spelbord.jpg") # Afbeelding voor achtergrond scherm (

WINDOW_SIZE = [1050, 750] # Afmeting van spelscherm instellen (in pixels: [breedte, hoogte])
ACHTERGROND_GROEN = ( 10, 254, 10 ) # RGB Kleur groen voor achtergrond

# -------- FUNCTIE DEFINITIES --------


# -------- PYGAME INITIALISATIE  --------
pygame.init()
screen = pygame.display.set_mode(WINDOW_SIZE) # Spelscherm maken (en opslaan in een variabele
pygame.display.set_caption("Scherm titel") # Titel van spelscherm instellen

clock = pygame.time.Clock()# Zorgt voor verversingssnelheid van het scherm


# -------- HOOFDLOOP VAN HET PROGRAMMA  --------
# Deze boolean laat ons spel straks lopen, totdat er op het kruisje wordt gekikt om af te slui
# (deze zet dan spel_is_afgelopen op True)
spel_is_afgelopen = False

while not spel_is_afgelopen:

    # --- Check gebeurtenissen (zoals muiskliks enz.) ---
    for event in pygame.event.get():
        if event.type == pygame.QUIT: # Het kruisje is aangeklikt
            spel_is_afgelopen = True # Het spel moet eindigen dus we zetten spel_is_afgelopen

    # --- Teken de graphics ---
    afmetingBord = ACHTERGROND_PLAATJE.get_rect() # Bepaal afmeting (rectangle van het plaatje
    screen.blit(ACHTERGROND_PLAATJE,afmetingBord) # Projecteer het plaatje op het scherm

#    screen.fill(ACHTERGROND_GROEN) # Maak achtergrond groen

    # --- Ververs het beeldschem met de nieuwe graphics ---

    clock.tick(60) # Ververs scherm met 60 frames per seconde
    pygame.display.flip() # Ververs het beeldscherm met de bijgewerkte versie


# -------- AFSLUITING  --------
pygame.quit() #  Sluit het scherm af
```

6. vensterAchtergrond.py

Plaatje tonen op scherm

Eerst moet je het plaatje laden. Let er op dat je de juiste extensie gebruikt (bijvoorbeeld .jpg of .png) en dat het plaatje in dezelfde map zit als je python code.

```python
# -------- CONSTANTEN EN GLOBALE VARIABELEN --------
# Afbeelding voor achtergrond scherm (spelbord)
plaatje = pygame.image.load("dice1.jpg")
```

7. Plaatje tonen op scherm

Daarna moet je binnen de while-loop de volgende code gebruiken zodat het plaatje daadwerkelijk getoond wordt. Geeft de coördinaten van de linkerbovenhoek van het plaatje.

```
screen.blit(plaatje, (100,50) ) # Zet het plaatje op het scherm op (100,50)
```

8 vensterAchtergrond.py

Reageren op input van toetsenbord

In de while-loop die je al hebt controleer je (continu) of er een toets is aangeslagen. Dit heet een gebeurtenis afhandeling (in het Engels: event handeling).

```
# --- Check gebeurtenissen (zoals muiskliks enz.) ---
for event in pygame.event.get():
    if event.type == pygame.QUIT: # Het kruisje is aangeklikt
        spel_is_afgelopen = True # Het spel moet eindigen dus we zetten
spel_is_afgelopen op True

    # Gebruiker heeft een toets ingedukt
    elif event.type == pygame.KEYDOWN:
        # Figure out if it was an arrow key.
        if event.key == pygame.K_LEFT:
            print("Keydown: arrow left")
        elif event.key == pygame.K_RIGHT:
            print("Keydown: arrow right")
        elif event.key == pygame.K_UP:
            print("Keydown: arrow up")
        elif event.key == pygame.K_DOWN:
            print("Keydown: arrow down")
        elif event.key == pygame.K_ESCAPE:
            print("Keydown: ESC")
            done = True
```

9. toetsEventAfhandeling.py

Soms wil je ook dat als de gebruiker een toets loslaat dat er iets gebeurt:

```
# Gebruiker laat een toets los
elif event.type == pygame.KEYUP:
    # If it is an arrow key
    if event.key == pygame.K_LEFT:
        print("Keyup: arrow left")
    elif event.key == pygame.K_RIGHT:
        print("Keyup: arrow right")
    elif event.key == pygame.K_UP:
        print("Keyup: arrow up")
    elif event.key == pygame.K_DOWN:
        print("Keyup: arrow down")
```

10. toetsEventAfhandeling.py

Een overzicht van alle codes is te vinden op: *http://thepythongamebook.com/en:glossary:p:pygame:keycodes*

Schrijf tekst in venster

Je moet eerst aangeven welke lettertype en grootte je wilt. Daarna ook de kleur en welke tekst. Als laatste moet je het naar het scherm afdrukken (met .blit(…)).

```
# --- Teken de graphics ---
font = pygame.font.Font(None, 36) # Bepaal lettertype, hier standaard met grootte
36
tekst = font.render("dit laten zien... ", True, WHITE) # Sla in een variabele de
tekst en kleur op
screen.blit(tekst, [10, 50]) # Drukt de tekst in de variabele af, op positie vanaf
linker-bovenhoek [horizontaal, vertikaal]
```

11. tekstTonen.py

De muis zichtbaar maken op het scherm:

De muis cursor zichtbaar maken doe je met:

```
# -------- PYGAME INITIALISATIE  --------
pygame.init()
pygame.mouse.set_visible(1)# Show the mouse cursor (let op: kun je niet op een
zwarte achtergrond zien
```

En onzichtbaar met:

```
pygame.mouse.set_visible(0)# Hide the mouse cursor
```

Reageren op input van de muis

Muiskliks

Hiermee bepaal je of er op een muisknop gelikt is, en waar de muis dan op het scherm stond. Je kunt de code voor de muispositie ook elders gebruiken zonder dat er geklikt is op de muis.

```
# --- Check gebeurtenissen (zoals muiskliks enz.) ---
for event in pygame.event.get():
    if event.type == pygame.QUIT: # Het kruisje is aangeklikt
        spel_is_afgelopen = True # Het spel moet eindigen dus we zetten spel_is_afgelopen
op True

    elif event.type == pygame.MOUSEBUTTONDOWN:
        pos = pygame.mouse.get_pos()
        mouse_xPos = pos[0]
        mouse_yPos = pos[1]
        print("Mouse clicked at coordinates: ("+str(mouse_xPos)+","+ str(mouse_yPos)+")")
```

12. muisEventAfhandeling.py

https://www.youtube.com/watch?v=ONAK8VZIcI4&feature=youtu.be

Muisbewegingen

Om een plaatje te laten bewegen over het scherm, dan moet elke keer als het scherm opnieuw getekend worden:

1) Teken het achtergrond plaatje
2) Bepaal de muispositie
3) Teken het plaatje op de muispositie
4) Ververs het scherm

```
screen.blit(ACHTERGROND_PLAATJE, ACHTERGROND_POSITIE) # Teken de achtergrond

muis_pos = pygame.mouse.get_pos() # Bepaal huidige positie van muis
screen.blit(SPELER_PLAATJE, muis_pos) # Zet het plaatje op het scherm

pygame.display.flip()# Ververs het scherm
clock.tick(60)
```

13. plaatjesBewegenMetMuis.py

Geluid afspelen

Eerst moet je het geluid eenmalig aan een variabele toekennen.

```
# -------- PYGAME INITIALISATIE  --------
pygame.init()
click_sound = pygame.mixer.Sound("laser5.ogg")
```

En daarna kun je het afspelen:

```
click_sound.play()
```

Kleuren

Kleuren geef je met hun rgb waarde weer. Dit kun je het beste bovenin je bestand bij de ‘Constanten’ definiëren. Hier een paar voorbeelden

```
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)
darkBlue = (0,0,128)
white = (255,255,255)
black = (0,0,0)
pink = (255,200,200)
```

Om een kleur te gebruiken hoef je alleen de naam te gebruiken, zoals hier waar we de scherm zwart maken:

```
screen.fill(black)
```

Tekenen

Bij het tekenen op het scherm moet je twee dingen doen:

1) Aangeven wat er getekend moet worden, bv. `screen.fill( 100, 100, 100 )`
2) Het scherm verversen, bv. met `pygame.display.flip()`

Scherm verversen (na het tekenen)

Na het tekenen moet je altijd het volgende aanroepen zodat het scherm ververst wordt en het dus ook daadwerkelijk zichtbaar wordt:

```
pygame.display.flip()
```

Scherm vullen met een kleur

```
screen.fill(color)
```

Voorbeeld: `screen.fill( 255, 255, 255 ) #255, 255, 255 geven de kleur aan`

Lijnen tekenen

```
pygame.draw.lines(screen, kleur, gesloten, lijstMetPunten, lijndikte)
```

- Bij gesloten geef je True aan als de eerste en laatste punt met elkaar verbonden moeten worden (en dus een gesloten figuur maakt), en anders False.
- Bij lijstMetPunten geef je met vierkante haken een lijst met punten mee, bv: [(100,100), (150,200), (200,400)]
- Bij lijndikte geef je lijndikte in pixels aan.
- Voorbeelden:
  - een lijn tussen twee punten A(100,100) en B(150,200):

```
pygame.draw.lines(screen, white, False, [(100,100), (150,200)], 1)
```

  - een vette rode V:

```
pygame.draw.lines(screen, red, False, [(100,100), (150,200), (200,100)], 3)
```

Rechthoek tekenen

```
pygame.draw.rect(screen, kleur, (x,y,breedte,hoogte), lijndikte)
```

- (x,y,breedte,hoogte): Geef met ronde haken de linker-bovenhoek aan (met x en y), en de breedte en de hoogte van de rechthoek

- Met lijndikte geef je de lijndikte in pixels aan. Als je 0 opgeeft, wordt de rechthoek helemaal gevuld.
- Voorbeeld: pygame.draw.rect(screen, red, [100, 100, 30, 30], 10)

Cirkel tekenen

```
pygame.draw.circle(screen, kleur, (x,y), straal, lijndikte)
```

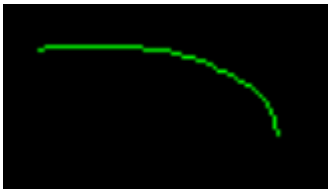
- Met (x,y) geef je de coördinaten van de middelpunt aan
- Met lijndikte geef je de lijndikte in pixels aan. Als je 0 opgeeft, wordt de cirkel helemaal gevuld.
- Voorbeeld: pygame.draw.circle(screen, red, (300,300), 50, 5)

Boog tekenen

```
pygame.draw.arc(screen, kleur, (x,y,breedte,hoogte), start_hoek, stop_hoek,
lijndikte)
```

- Tekent een boog (gedeelte van een ellips)
- Geef met ronde haken de linker-bovenhoek aan (met x en y), en de breedte en de hoogte van de rechthoek die de hele ellips omvat
- start_hoek, stop_hoek: de hoek op een eenheidscirkel in radialen (niet in graden) waar de boog begint en eindigt.
- Bij gelijke breedte en hoogte wordt (een deel van) een cirkel getekend.
- Voorbeeld:

```
pygame.draw.arc(screen, green, (500,500,100,50), 0, 2, 1)
```



Animaties / bewegende beelden

Om beelden te laten bewegen moet je code in je while-loop stoppen dat:

1) Het oude wegveegt (bijvoorbeeld `screen.fill(white)` of `screen.fill(black)`)
2) Het nieuwe tekent (bijvoorbeeld met `pygame.draw.circle(…)`), maar deze keer op een verschillende plek door de coördinaten aan te passen.
3) Het scherm verversen met: `pygame.display.update()`

<u>Uitgewerkt voorbeeld:</u>

Stel je wilt een rechthoek laten vliegen van rechts naar links.

Dan teken je een rechthoek op ergens links op het scherm, dus met x-coördinaat gelijk aan 0, bijvoorbeeld een rode rechthoek (30 bij 30 breed) op coördinaat (0,100):

```
pygame.draw.rect(screen, red, [0, 100, 30, 30], 0)
```

Om van links naar rechts te bewegen moet het x-coördinaat steeds veranderen, steeds iets groter worden. Om dit te doen maak je **buiten** de while-loop een variabele aan dat begint met de waarde 0.

```
x_coordinaat = 0
```

In de while-loop laat je de rechthoek tekenen, maar gebruikt nu de variabele x-coördinaat:

```
pygame.draw.rect(screen, red, [x_coordinaat, 100, 30, 30], 0)
```

Na het tekenen verhoog je de x-coordinaat zodat hij de volgende keer iets verder naar rechts getekend wordt:

```
x_coordinaat += 1
```

Het geheel ziet er dan ongeveer zo uit:

```
spel_is_afgelopen = False

x_coordinaat = 0
while not spel_is_afgelopen:

    # --- Check gebeurtenissen (zoals muiskliks enz.) ---
    for event in pygame.event.get():
        if event.type == pygame.QUIT: # Het kruisje is aangeklikt
            spel_is_afgelopen = True # Het spel moet eindigen dus we zetten
spel_is_afgelopen op True
    # --- Teken de graphics ---

    screen.fill(black) # het oude scherm wegvegen
    pygame.draw.rect(screen, red, [x_coordinaat, 100, 30, 30], 0)
    x_coordinaat += 1 # x-coordinaat verhogen om plaatje naar rechts te schuiven

    # --- Ververs het beeldschem met de nieuwe graphics ---

    clock.tick(60) # Ververs scherm met 60 frames per seconde
    pygame.display.flip() # Ververs het beeldscherm met de bijgewerkte versie

# -------- AFSLUITING  --------
pygame.quit() #  Sluit het scherm af
```