Python 3 - Game of Pong

Opdracht 1 - het scherm

Inleiding

In deze module gaan we een oud spel 'Pong' maken. Pong is een oud Arcade-spel en je keert een hiermee een paar basis concepten van game development.

Deze module is niet makkelijk. Lees goed wat er staat en probeer de code te begrijpen. Begrijp je de code niet, dan loop je vast verderop in de cursus.

Houd je aan de opdracht. Een alternatieve versie van Pong (via AI) wordt niet geaccepteerd. Tevens wordt code waar niet om is gevraagd ook afgekeurd.

Nogmaal lees goed wat er staat en vraag telkens aan het eind van de opdracht of je alles begrijpt. Indien niet lees de opdracht dan nog een keer door. Vraag een mede student en als je er dan nog niet uitkomt vraag de docent.

turtle

We gaan gebruik maken van de turtle module van Python.

```
import turtle
```

De `turtle` module in Python is een eenvoudige manier om grafische tekeningen te maken door middel van een virtuele "schildpad" die over het scherm beweegt en lijnen tekent.

De turtle-module gebruiken we om grafische objecten te maken en te verplaatsen. Het helpt ons om het speelveld, de paddle, en de bal te tekenen.

Nadat de turtle module is geïmporteerd, maken we een window (venster) voor het spel.

```
# Venster instellen
wn = turtle.Screen()
wn.title("Pong voor <Jouw Naam>")
wn.bgcolor("black")
wn.setup(width=800, height=600)
wn.tracer(0)
```

```
input("Press any key to continue...") # tijdelijke toevoeging t.b.v. testen
```

Hier maken we een venster met de titel "Pong voor<Jouw Naam>". Vervang <Jouw Naam> door jouw naam.

De achtergrondkleur is zwart en de afmetingen van het venster zijn 800 bij 600 pixels. Met `wn.tracer(0)` zorgen we ervoor dat het scherm alleen ververst als wij dat willen, wat handig is voor het maken van animaties.

Opdracht

Test de code.

Leg in je eigen woorden uit waarom regel 8 in de code staat. Wat heeft regel 8 te maken met testen?

Wat gebeurt er als deze regel er niet in staat.

Inleveren

Korte uitleg in eigen woorden waarom regel 8 in de code staat.

Opdracht 2 - paddle

In deze opdracht maken we een paddle voor de speler. De paddle is een rechthoek die de speler kan verplaatsen om de bal terug te kaatsen.

We gaan ook leren wat coördinaten zijn.

De code voor de paddle:

```
# Paddle
paddle = turtle.Turtle()
paddle.shape("square")
paddle.color("white")
paddle.shapesize(stretch_wid=6, stretch_len=1)
paddle.penup()
paddle.goto(0, 0)
```

```
wn.update()
```

Uitleg:

- We maken een turtle-object genaamd `paddle`.
- `paddle.speed(0)` zorgt ervoor dat de paddle direct getekend wordt zonder animatie.
- `paddle.shape("square")` geeft de paddle de vorm van een vierkant.
- `paddle.color("white")` maakt de paddle wit.
- `paddle.shapesize(stretch_wid=6, stretch_len=1)` maakt de paddle rechthoekig door de breedte te stretchen.
- `paddle.penup()` zorgt ervoor dat de paddle niet tekent als hij beweegt.
- `paddle.goto(0, 0)` plaatst de paddle op het midden van het scherm verschijnt.
- `wn.update()` zorgt ervoor dat het scherm wordt geüpdatet.

Coördinaten

Het scherm is verdeeld in coördinaten. Een coördinaat is (0,0) en dat is het midden van het scherm.

Het eerste cijfer wordt de x-waarde genoemd en is de horizontale positie (links-rechts).

Het tweede cijfer wordt de y-waarde genoemd en is de verticale positie (boven-beneden).

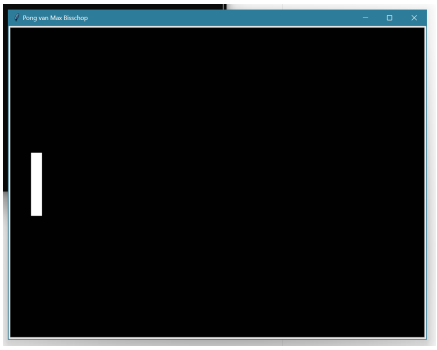
> Het eerste cijfer in een coördinaat is de x-waarde en de tweede coördinaat is de y-waarde. De x-waarde geeft de horizontale postie weer en de y-waarde geeft de vertikale positie weer.

Voorbeelden

(10,0)	x-waarde is 10 en de y-waarde is 0; 10 punten rechts van het midden.
(0,10)	10 punten boven het midden.
(-10,0)	10 punten links van het midden.
(0,-10)	10 punten onder het midden
(10,-10)	15 punten rechts en dan 10 naar beneden vanuit het midden.

Opdracht

Verander de code zodat de paddle aan de linkerkant van het scherm staat. Hiervoor moet je dus de coördinaten aanpassen.



Test je programma uit en zorg ervoor dat je een witte paddle ziet aan de **linkerkant** van het venster.

Inleveren

Screenshot van het scherm waarin je de paddle ziet.

Opdracht 3 - paddle bewegen

We gaan ervoor zorgen dat je de paddle kan bewegen.

De paddle kan twee richtingen op: up (naar boven) en down (naar beneden).

We maken eerst twee functies die de paddle up of down kunnen bewegen.

```
def paddle_up():
    y = paddle.ycor()
    if y < 250:
        y += 20
    paddle.sety(y)

def paddle_down():
    y = paddle.ycor()
    if y > -240:
        y -= 20
    paddle.sety(y)

# Toetsenbordbinding
```

```
wn.listen()
wn.onkeypress(paddle_up, "w")
wn.onkeypress(paddle_down, "s")
```

Uitleg:

- `paddle_up()` verhoogt de y-coördinaat van de paddle met 20 als deze nog niet de bovenkant van het scherm heeft bereikt.
- `paddle_down()` verlaagt de y-coördinaat van de paddle met 20 als deze nog niet de onderkant van het scherm heeft bereikt.

Als we deze code plaatsen dan gebeurt er nog niets. Dat komt omdat we de coördinaten wel veranderen, maar we doen geen scherm-update.

Gedurende het spel moeten we telkens een scherm update uitvoeren. Dit doen we in een loop die we voor nu even 'oneindig maken' (stopt niet vanzelf).

```
while True:
    wn.update()
```

Deze loop wordt ook wel de *game loop* genoemd.

> De game-loop is het programma-onderdeel dat steeds controleert of er iets gebeurt en tekent telkens een nieuw scherm tekent.

Opdracht

Test je spel uit en als het goed is, kan je de paddle bewegen.

Tip: Omdat de code nu een *game-loop* heeft, zal je deze moeten onderbreken met een CTRL-C in de command prompt waar je het spel hebt gestart.

Inleveren

De gehele werkende code (.py) tot nu toe.

Opdracht 4 - de bal

Om te beginnen maken we een bal, op eenzelfde manier als we de paddle hebben gemaakt.

Deze code staat boven de *game-loop*.

```
# Bal
ball = turtle.Turtle()
ball.shape("square")
ball.color("white")
ball.penup()
ball.goto(0, 0)
ball.dx = 0.1
ball.dy = -0.1
```

Uitleg:

- `ball = turtle.Turtle()`: Maakt een nieuw turtle-object genaamd `ball`.
- `ball.speed(1)`: Stelt de animatiesnelheid van de bal in.
- `ball.shape("square")`: Geeft de bal de vorm van een vierkant.
- `ball.color("white")`: Maakt de bal wit.
- `ball.penup()`: Voorkomt dat de bal lijnen tekent tijdens beweging.
- `ball.goto(0, 0)`: Plaatst de bal in het midden van het scherm.
- `ball.dx = 0.2`: Stelt de horizontale snelheid van de bal in.
- `ball.dy = -0.2`: Stelt de verticale snelheid van de bal in.

Test

Test je code uit. Als het goed is zie je een stilstaande bal in het midden van het scherm.

Beweging

Waarom beweegt de bal niet??

Dat komt omdat de coördinatoren van de bal niet worden aangepast.

In de game-loop, voor het updaten van het scherm, gaan we dit doen met deze code.

```
# Beweeg de bal
ball.setx(ball.xcor() + ball.dx)
ball.sety(ball.ycor() + ball.dy)
```

Uitleg:

- `ball.setx(ball.xcor() + ball.dx)`: Verplaatst de bal horizontaal door de huidige x-coördinaat te verhogen met `ball.dx`.
- `ball.sety(ball.ycor() + ball.dy)`: Verplaatst de bal verticaal door de huidige y-coördinaat te verhogen met `ball.dy`.

Test

Test je code uit. Je ziet nu een bewegende bal, die vrij snel het beeld uit verdwijnt.

Opdracht

Leg in eigen woorden uit waarom de bal uit het scherm verdwijnt.

Inleveren

Korte uitleg waarom de bal van het scherm verdwijnt in eigen woorden.

Opdracht 5 - bal stuitert terug

Je weet nu dus waarom de bal van het scherm verdwijnt?

De vraag is nu: wat doen we eraan?

Het punt is dat als de bal de randen van ons scherm nadert de bal terug moet 'stuiteren'.

Dat betekent dat we moeten detecteren wanneer de bal de randen bereikt en dat we dan de richting van de bal moeten veranderen.

Laten we proberen te detecteren als de bal de rand raakt en als dat zo is dan veranderen we de richting.

Daarvoor plaatsen we deze code in de game-loop.

```
# detecteer randen van het scherm
if (ball.xcor() > 200 or ball.xcor() < -200 ):
   ball.dx *= -1
if (ball.ycor() > 200 or ball.ycor() < -200 ):
```

```
    ball.dy *= -1
```

De ball.dx en ball.dy geven de snelheid in horizontalen- en verticale richting aan.

Er staat dus: als de x-coördinaat van de bal groter dan of kleiner dan een bepaalde waarde is, vermenigvuldig dan de horizontale snelheid met -1. Stel de snelheid is 1 dat wordt deze dan -1 en stel de snelheid is -1 dan wordt die 1. Dus als de snelheid naar rechts is, dan gaat die naar links en andersom.

Hetzelfde gebeurt met de y-coördinaat.

Test

Test je code uit. Je ziet nu een bewegende bal, die terug stuitert, maar niet precies bij de randen van het scherm.

Taak 1

Pas de code aan zodat de bal werkelijk aan de rand van het scherm terug 'stuitert'. Pas hiervoor de coördinaten in de code aan.

Taak 2

Zorg ervoor dat de bal aan alle kanten van het scherm terug stuitert, maar **niet** als die aan de linkerkant komt.

Dus de bal stuitert terug als die aan de onder-, boven of rechterkant komt, maar niet als die aan de linkerkant komt.

Inleveren

De aangepaste volledige code tot nu toe.

Opdracht 6 - raak of mis?

De bel verdwijnt nu als die aan de linkerkant van het scherm komt. Dat is goed, behalve als daar de paddle staat. We moeten dus code maken in de game-loop die detecteert of de ball en de paddle elkaar raken.

```
# Detecteer botsing met paddle
if (ball.dx < 0 and ball.xcor() < -350): # ball beweegt naar links en zit bij de linker
```

```
zijkant.
    if (paddle.ycor() - 60 < ball.ycor() < paddle.ycor() + 60): # bal 'raakt' de bal
        ball.dx *= -1 # beweeg de bal de andere kant uit (horizontaal)
        ball.dy *= -1 # beweeg de bal de andere kant uit (verticaal)
    else:
        ball.dx = 0
        ball.dy = 0
```

Uitleg:

- `ball.dx < 0`: Controleert of de bal naar links beweegt.
- `ball.xcor() < -360`: Controleert of de bal bij/over de linker zijlijn is..
- `paddle.ycor() - 60 < ball.ycor() < paddle.ycor() + 60`: Controleert of de bal binnen het bereik van de paddle is. De paddle was size 6. Dit betekent 6 eenheden en in de turtle library is 1 eeheid 20 pixels. De paddle is dus 6*20 = 120 pixels hoog. Vanuit het midden gezien is dat dus 60 pixels omhoog en 60 pixels omlaag.
- `ball.dx *= -1`: Verandert de richting van de bal horizontaal.
- `ball.dx *= 0` en `ball.dy *= 0`: Zet de bal stil. De horizontale- en verticale snelheid wordt 0.

Test

Test je code uit. Op zich moet het spel nu werken. De bal verdwijnt als je hem mist maar als je de bal raakt met de paddle dan stuitert de bal terug.

Opdracht

Maak een variabele score.

```
# Score variabele
score = 0
```

Telkens als je de bal raakt dan verhoog je de score.

Om te testen of het werkt, druk je telkens nadat de score is, verhoogt de variabele score af.

```
score = score + 1
print(score)
```

Speel het spel en houd je cmd-windows in de gaten. Als het goed is wordt de score daar afgedrukt.

Inleveren

Eens schermafdruk van het gehele cmd-window waarin de score wordt afgedrukt.

Opdracht 7 - Score op scherm

Weg hebben een variabele score. Deze gaan we op het scherm plaatsen.

Om de score bij te houden en op het scherm te tonen maken we een nieuw turtle object.

```
# Score variabele
score = 0

# Pen om de score weer te geven
pen = turtle.Turtle()
pen.speed(0)
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Score: 0", align="center", font=("Courier", 24, "normal"))

def update_score():
    pen.clear()
    pen.write("Score: {}".format(score), align="center", font=("Courier", 24, "normal"))
```

We hebben nu een 'lege score', om die te vullen maken we een functie.

```
def update_score():
    pen.clear()
    pen.write("Score: {}".format(score), align="center", font=("Courier", 24, "normal"))
```

We hebben de functie om de score op het scherm af te drukken. Wat we nu moeten doen is de score telkens als die is veranderd opnieuw op het scherm zetten.

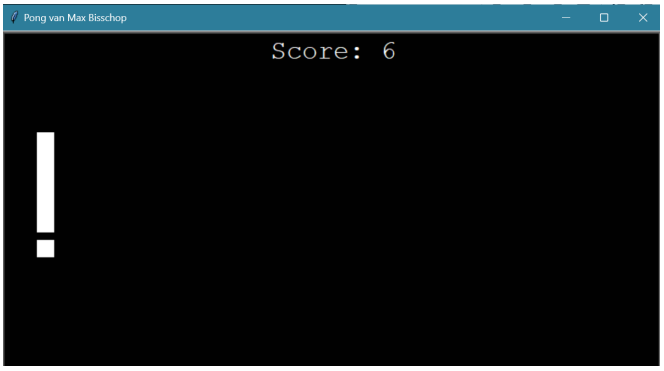
```
update_score()  # Werk de score weergave bij
```

Als de code werkt dan kan de print() die we in de vorige opdracht hebben geplaatst uit de code worden gehaald.

Inleveren

Eens schermafdruk van het spel waarbij de score wordt getoond. Zorg ervoor dat er een score hoger dan 0 wordt getoond.

Voorbeeld



Opdracht 8 - Levens

Telkens als het spel stop omdat je de bal niet hebt geraakt me de paddle, stop het spel.Het spel stopt door de bal stil te zetten. De x- en y-snelheid wordt op 0 gezet.

Dit gaan we aanpassen. Je begint met drie levens en telkens als je de bal mist, stopt het spel voor 3 seconden. Als je nog genoeg levens hebt, zal het spel na 2 seconden weer verder gaan. De bal verandert van richting (terug naar recht) en de snelheid wordt weer op de beginwaarde 0.1 gezet.

Als je levens op zijn dan stopt het spel en wordt er in het midden van het scherm 'Game Over' getoond.

Stappenplan

Stap 1 - Laat "Game Over" zien.

Maak een functie die de tekst game over laat zien.

```
def game_over():
    pen.goto(0, 0)
```

```
    pen.write("Game Over", align="center", font=("Courier", 36, "normal"))
```

Roep deze functie aan op de plaats in de game-loop waar het spel eindigt.

Stap 2a - Maak een variabele die het aantla levens bijhoud

Net zoals je de score bijhoud, houd je ook het aantal levens bij. Maak een variabele en zet de initiële waarde op 3. Zorg dat op de juisten plaats in de game-loop het aantal levens wordt veminderd met één.

Stap 2b - Game stopt alleen als aantal levens 0 is.

Nu veranderd het moment dat we game-over laten zien. Alleen als het aantal levens 0 is, laten we game over zien.

Op de plaats in de code waar je de bal hebt gemist, kunnen er twee dingen gebeuren:

1. Je controleert of het aantal levens 0 is, dan zet je de snelheid van de bal op 0 en roep je de functie game_over() aan.
2. Is het aantal levens niet 0, dan vermidner je het aantal levens met 1 en laat je de bal terug stuiteren alsof je heb had geraakt met de paddle.

Stap 3 - druk jet aantal levens af

Je kunt hiervoor de bestaande functie aanpassen en naast de score ook het aantal levens aanpassen.

```
def update_score():
    pen.clear()
    pen.write("Score: {} Levens: {}".format(score, lifes), align="center", font=("Courier",
24, "normal"))
```

Zorg er wel voor dat als je het aantal levens aanpast, je deze functie aanroept zodat de score wordt ververst.

Zorg er ook voor dat er bij het begin van het spel "Score: 0 Levens: 3" wordt getoond.

Stap 4 - pauzeer de game als je de bal mist

Met het volgende commando kan je de game 3 seconden pauzeren (vergeet niet de module time te importeren).

```
time.sleep(3)
```

Zodra je de bal mist, dan pauzeer je de game voor 3 seconden.

Inleveren

Eens schermafdruk van het spel waarbij de score en het aantal levens wordt getoond.

Voorbeeld



Opdracht 9 - be creative!

Maak een variatie op het spel. Kies zelf één of meer van de volgende opties.

1. De bals stuitert nu altijd op een bepaalde manier. Dat komt omdat de ball.dx en ball.dy een vaste waarde hebben 0,1 of -0.1. Je kunt dit laten variëren. Dat kan je at random doen, maar je kan het ook laten afhangen waar op de paddle je de bal raakt. Raak je hem aan de boven of onderkant dan kan je wat doen met de ball.dy, maak hem dan bijvoorbeeld 01.15 of -0.15.
2. Je zou de snelheid langzaam kunnen opvoeren. Bijvoorbeeld na elke 5 punten gaat de bal iets sneller. Vergeert niet de snelheid weer aan te passen na het verliezen van een leven.
3. Je kan de paddle groter en kleiner maken. Bijvoorbeeld na elke 5 punten iets kleiner. Vergeet hem dan niet na het verliezen van een leven weer op de originele grootte te zetten.
4. Je zou de paddle in het midden een gat kunnen geven. Komt de bal precies in het gat dn krijg je er een leven bij.
5. Pas de kleur van de bal aan al naar gelang je meer punten scoort.

Heb jij nog een ander idee, bespreek dit dan met de docent.

Inleveren

Beschrijf op de eerste regels in de code in commentaar wat je hebt gekozen en hoe je dat gedaan hebt. Lever daarna je gehele code in.

Opdracht 10 - Quiz

Vraag in de klas (op school) of je de bijbehorende quiz mag maken. De quiz bevat 7 vragen over deze module.

Je krijgt 7 vragen en je moet er minimaal 5 goed hebben (dat is 70% of hoger).

Je hebt maximaal 3 pogingen.

Inleveren

Schermafdruk met een uitslag van 70% og hoger.

Revision #4
Created 2024-05-30 08:52:21 UTC by Max
Updated 2024-12-18 14:41:04 UTC by Max