PROJECT 7

Bellenschieter

In dit project komen alle vaardigheden aan bod die je in dit hoofdstuk hebt geleerd. Het is een groot project, dus doorloop het stapsgewijs en sla je werk regelmatig op. Probeer de samenhang tussen de onderdelen te begrijpen voordat je verder gaat naar de volgende stap. Aan het eind heb je een spel dat je met je vrienden kunt spelen.

ZIE OOK
- ‹ 154–155 Vensters maken
- ‹ 156–157 Kleur en coördinaten
- ‹ 158–159 Vormen maken

Doel van het spel

Voordat je een codering schrijft, moet je nadenken over het algehele plan voor het spel en hoe het moet werken. Hier de belangrijkste regels die bepalen hoe een spel wordt gespeeld:

De speler bestuurt een duikboot

De duikboot beweeg je met de pijltjestoetsen

Bellen kapotschieten levert punten op

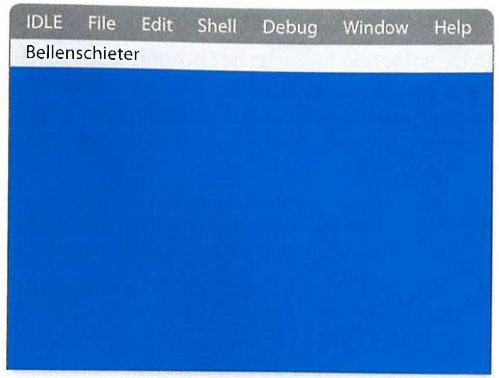
Een tijdsklok loopt 30 seconden vanaf start

Een score van 1000 punten levert extra tijd op

Het spel is afgelopen als de tijd om is

Maak het spelvenster en de duikboot

Begin met het instellen van het speelveld. Open een nieuw Code-venster in IDLE. Voer onderstaande codering in om het speelvenster en de duikboot die je bestuurt, te maken.



1 Met de Tkinter-bibliotheek kun je de grafische gebruikersinterface (GUI) maken. Deze codering zal het hoofdvenster van het spel aanmaken.

```
from tkinter import *
HEIGHT = 500
WIDTH = 800
window = Tk()
window.title('Bubble Blaster')
c = Canvas(window, width=WIDTH, height=HEIGHT, bg='darkblue')
c.pack()
```

Importeert alle Tkinter-functies

Bepaalt het formaat van het venster

Geef het spel een pakkende naam

Stelt donkerblauw in als achtergrondkleur (de zee)

Maakt een canvas waarop je kunt tekenen

2 Een eenvoudige afbeelding zal de duikboot in dit spel vertegenwoordigen. Je kunt deze aanmaken met enkele tekenfuncties uit Tkinter. Typ deze codering en voer hem uit.

De duikboot wordt gevormd door een driehoek in een cirkel

```
ship_id = c.create_polygon(5, 5, 5, 25, 30, 15, fill='red')
ship_id2 = c.create_oval(0, 0, 30, 30, outline='red')
SHIP_R = 15
MID_X = WIDTH / 2
MID_Y = HEIGHT / 2
c.move(ship_id, MID_X, MID_Y)
c.move(ship_id2, MID_X, MID_Y)
```

Tekent een rode driehoek voor de duikboot

De radius (omvang) van de duikboot

De variabelen 'MID_X' en 'MID_Y' geven de coördinaten van het midden van het scherm

Tekent een rode cirkelomtrek

Beweegt beide delen van de duikboot naar het midden van het scherm

Vergeet niet je werk op te slaan

BELLENSCHIETER

De duikboot besturen

In de volgende fase van het programma schrijf je de codering, waardoor de duikboot beweegt als je op de pijltjestoetsen drukt. De codering maakt een functie aan die 'gebeurtenissenmanager' wordt genoemd. De gebeurtenissenmanager controleert op welke toets is gedrukt en beweegt de duikboot.

3 Typ deze code in om een functie met de naam 'move_ship' te maken. Deze functie beweegt de duikboot in de juiste richting als er op een pijltjestoets wordt gedrukt. Probeer het uit en zie hoe het werkt.

```
SHIP_SPD = 10
def move_ship(event):
    if event.keysym == 'Up':
        c.move(ship_id, 0, -SHIP_SPD)
        c.move(ship_id2, 0, -SHIP_SPD)
    elif event.keysym == 'Down':
        c.move(ship_id, 0, SHIP_SPD)
        c.move(ship_id2, 0, SHIP_SPD)
    elif event.keysym == 'Left':
        c.move(ship_id, -SHIP_SPD, 0)
        c.move(ship_id2, -SHIP_SPD, 0)
    elif event.keysym == 'Right':
        c.move(ship_id, SHIP_SPD, 0)
        c.move(ship_id2, SHIP_SPD, 0)
c.bind_all('<Key>', move_ship)
```

De duikboot beweegt tot hier als op een toets gedrukt wordt

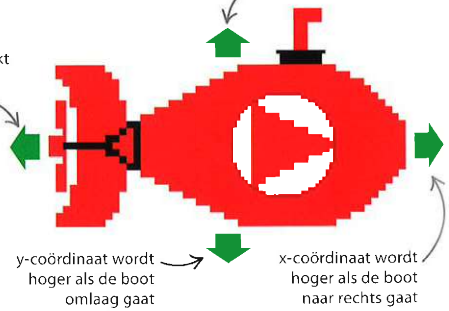
Beweegt de twee delen van de duikboot als op de toets pijltje omhoog gedrukt wordt

Deze regels worden geactiveerd als op de toets pijltje omlaag wordt gedrukt; de duikboot gaat naar beneden

De duikboot gaat naar links als op de toets pijltje naar links wordt gedrukt

Beweegt de duikboot rechts als op de toets pijltje naar rechts wordt gedrukt

Vertelt Python 'move_ship' uit te voeren als op een toets wordt gedrukt



Vergeet niet je werk op te slaan

▷ **Hoe het werkt**

De functie 'move_ship' beweegt de boot in verschillende richtingen. Als je de x- en y-coördinaten optelt, gaat hij naar rechts en omlaag, terwijl hij links en omhoog gaat als je ze aftrekt.

Bereid je voor op de bellen

Nu de duikboot kan bewegen, kun je de bellen aanmaken waarop je kunt schieten. Elke bel heeft een ander formaat en beweegt met een andere snelheid.

4 Elke bel heeft een eigen ID-getal (daaraan kan het programma elke bel herkennen), een formaat en een snelheid nodig.

```
from random import randint
bub_id = list()
bub_r = list()
bub_speed = list()
MIN_BUB_R = 10
MAX_BUB_R = 30
MAX_BUB_SPD = 10
GAP = 100
def create_bubble():
    x = WIDTH + GAP
    y = randint(0, HEIGHT)
    r = randint(MIN_BUB_R, MAX_BUB_R)
    id1 = c.create_oval(x - r, y - r, x + r, y + r, outline='white')
    bub_id.append(id1)
    bub_r.append(r)
    bub_speed.append(randint(1, MAX_BUB_SPD))
```

Dit maakt drie lege lijsten aan om de ID's, radius (formaat) en snelheid van elke bel in op te slaan

Stelt de minimale radius van de bel in op 10 en de maximale op 30

Bepaalt de positie van de bel op het canvas

Kiest een willekeurig formaat voor de bel tussen de maximale en minimale mogelijke waarden

Deze coderingsregel maakt een nieuwe belvorm aan

Voegt de ID, radius en snelheid van de bel toe aan de drie lijsten

TIPS VAN EXPERTS

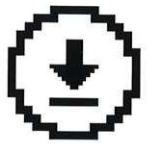
Bellenlijsten

Er zijn drie lijsten om informatie over elke bel in op te slaan. De lijsten zijn leeg en worden gevuld met info over elke bel die jij toevoegt als je hem maakt. Elke lijst bevat een andere hoeveelheid informatie.

bub_id: slaat het ID-getal van de bel op waarmee het programma hem later kan laten bewegen.

bub_r: slaat de radius van de bel op.

bub_speed: slaat op hoe snel de bel door het scherm beweegt.



Vergeet niet je werk op te slaan

BELLENSCHIETER

Laat de bellen bewegen

Je hebt nu lijsten om de ID, het formaat en de snelheid van de bellen, die willekeurig worden aangemaakt, in op te slaan. De volgende stap is het schrijven van de codering waardoor de bellen over het scherm bewegen.

5 Deze functie doorloopt de lijst met bellen en brengt ze om de beurt in beweging.

```
def move_bubbles():
    for i in range(len(bub_id)):
        c.move(bub_id[i], -bub_speed[i], 0)
```

Loopt langs elke bel in de lijst

Beweegt de bel met een bepaalde snelheid over het scherm

6 Dit wordt de hoofdloop voor het spel. Hij wordt eindeloos herhaald als het spel bezig is. Probeer hem eens op te starten!

Vergeet niet je werk op te slaan

```
from time import sleep, time
BUB_CHANCE = 10
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    window.update()
    sleep(0.01)
```

Importeert de functies die je nodig hebt uit de Time-bibliotheek

Genereert een willekeurig getal tussen 1 en 10

Als het willekeurige getal 1 is, maakt het programma een nieuwe bel aan (gemiddeld een op de 10 keer, anders komen er te veel bellen!)

Start de functie 'move_bubbles'

Updatet het venster om de bellen die verdwenen zijn, opnieuw te tekenen

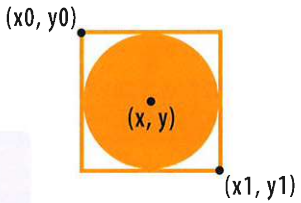
Vertraagt het spel, zodat het niet te snel wordt om te spelen

7 Maak nu een handige functie om uit te zoeken waar een bepaalde bel zich bevindt, gebaseerd op zijn ID. Deze code moet je meteen na de code die je in stap 5 hebt gemaakt, toevoegen.

```
def get_coords(id_num):
    pos = c.coords(id_num)
    x = (pos[0] + pos[2])/2
    y = (pos[1] + pos[3])/2
    return x, y
```

Berekent de x-coördinaat van het midden van de bel

Berekent de y-coördinaat van het midden van de bel



△ **Bellen lokaliseren**
De functie vindt het midden van de bel door het punt halverwege de hoeken van het hok eromheen te nemen.

Hoe je de bellen kunt laten knallen

De speler scoort punten door de bellen te laten knallen, dus dan moet het programma de bellen kunnen laten verdwijnen van het scherm. Dat lukt met de volgende functies.

8 Met deze functie wordt een bel uit het spel verwijderd. Dit doet hij door hem uit alle lijsten te verwijderen en van het canvas. Je moet deze codering toevoegen meteen na de code die je in stap 7 hebt gegeven.

```
def del_bubble(i):
    del bub_r[i]
    del bub_speed[i]
    c.delete(bub_id[i])
    del bub_id[i]
```

Deze functie verwijdert de bel met ID 'i'

Verwijdert de bel uit de radius- en snelheidslijsten

Verwijdert de bel van het canvas

Verwijdert de bel uit de ID-lijst

9 Typ deze codering om een functie te maken die de bellen verwijdert die buiten het scherm zijn beland. Je moet deze code invoeren meteen na de code in stap 8.

```
def clean_up_bubs():
    for i in range(len(bub_id)-1, -1, -1):
        x, y = get_coords(bub_id[i])
        if x < -GAP:
            del_bubble(i)
```

Dit doorloopt de bellenlijst achterwaarts om te voorkomen dat de loop 'for' een fout veroorzaakt als er bellen zijn verwijderd

Zoekt uit waar de bel zich bevindt

Als de bel buiten het veld is, wordt hij verwijderd; anders zou hij het spel langzamer maken

10 Update nu de hoofdloop van het spel (van stap 6) om de nuttige functies die je net hebt aangemaakt, in te voegen. Start hem op en controleer of er geen foutmeldingen zijn.

```
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    window.update()
    sleep(0.01)
```

Maakt een nieuwe bel

Updatet de posities van alle bellen

Verwijdert bellen buiten het veld

Tekent het venster opnieuw om de verschillen te tonen

Vergeet niet je werk op te slaan

BELLENSCHIETER

De afstand tussen twee punten berekenen

In dit spel en in vele andere is het handig om de afstand tussen twee objecten te kennen. Hier leer je een bekende wiskundige formule te gebruiken die de computer kan berekenen.

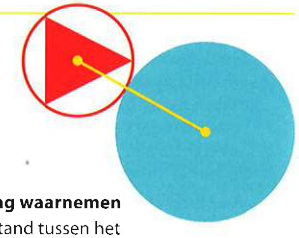
11 De functie berekent de afstand tussen twee objecten. Voeg dit coderingsfragment toe direct na de code die je hebt gemaakt in stap 9.

```
from math import sqrt
def distance(id1, id2):
    x1, y1 = get_coords(id1)
    x2, y2 = get_coords(id2)
    return sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

- Laadt de functie 'sqrt' vanuit de Math-bibliotheek
- Krijgt de positie van het eerste object
- Krijgt de positie van het tweede object
- Geeft de afstand tussen beide

Laat de bellen knappen

De speler scoort punten door de bellen kapot te schieten. Grote en snelle bellen zijn meer punten waard. Het volgende coderingsdeel berekent wanneer een bel knapt aan de hand van zijn radius (de afstand van het midden naar de rand).



▷ **Botsing waarnemen**

Als de afstand tussen het midden van de duikboot en het midden van een bel minder is dan hun beide radiussen opgeteld, hebben ze gebotst.

12 Als de duikboot en een bel tegen elkaar aan knallen, moet het programma de bel laten knappen en de score aanpassen. Dit coderingsfragment moet je direct na de codering in stap 11 plaatsen.

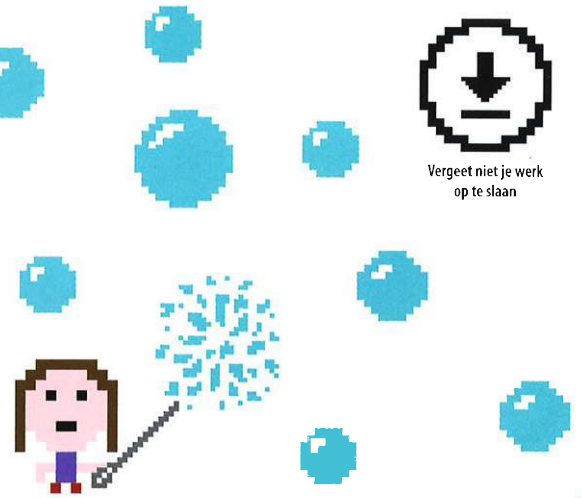
```
def collision():
    points = 0
    for bub in range(len(bub_id)-1, -1, -1):
        if distance(ship_id2, bub_id[bub]) < (SHIP_R + bub_r[bub]):
            points += (bub_r[bub] + bub_speed[bub])
            del_bubble(bub)
    return points
```

- Deze variabele houdt de score bij
- Deze loop doorloopt de hele bellenlijst (hij zoekt achterwaarts om fouten te voorkomen als er bellen worden verwijderd)
- Checkt botsingen tussen duikboot en bellen
- Berekent de puntenwaarde van deze bel en voegt die toe aan 'points'
- Verwijdert de bel
- Geeft de puntenwaarde

13 Update nu de hoofdloop van het spel om de functies te installeren die je zojuist hebt aangemaakt. Onthoud dat de volgorde belangrijk is, dus let erop dat je alles op de juiste plek zet. Voer de code dan uit. De bellen moeten knappen als ze de duikboot raken. Kijk in het Shell-venster wat je score is.

```
score = 0
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    score += collision()
    print(score)
    window.update()
    sleep(0.01)
```

- Zet de score aan het begin van het spel op nul
- Maakt nieuwe bellen aan
- Telt de score op bij het totaal
- Toont de score in het Shell-venster – het zal later correct worden weergegeven
- Dit pauzeert de actie korte tijd – probeer dit te verwijderen en kijk wat er gebeurt



Vergeet niet je werk op te slaan

TIPS VAN EXPERTS

Python-snelkoppeling

De code 'score += collision()' is een snelkoppeling om 'score = score + collision()' te schrijven. Het voegt de score door botsing toe aan de totale score en updatet die daarna. Dit soort codering komt veel voor, dus is een snelkoppeling handig. Je kunt hetzelfde ook doen met het symbool '–'. Zo is 'score –= 10' bijvoorbeeld hetzelfde als 'score = score – 10'.

BELLENSCHIETER

Voeg de 'final touches' toe

De belangrijkste onderdelen van het programma werken nu. Het enige wat rest, is de afwerking: weergeven van de score en een tijdslimiet instellen waarbij wordt afgeteld tot het spel voorbij is.

14 Typ deze code in na de code die je hebt ingevoerd in stap 12. Hiermee instrueer je de computer om de score en de resterende speeltijd weer te geven.

```
c.create_text(50, 30, text='TIME', fill='white' )
c.create_text(150, 30, text='SCORE', fill='white' )
time_text = c.create_text(50, 50, fill='white' )
score_text = c.create_text(150, 50, fill='white' )
def show_score(score):
    c.itemconfig(score_text, text=str(score))
def show_time(time_left):
    c.itemconfig(time_text, text=str(time_left))
```

Maakt de labels 'TIME' en 'SCORE' aan om de speler uit te leggen wat de getallen betekenen

Stelt de scores en de resterende tijd in

Geeft de score weer

Geeft de resterende speeltijd weer

15 Stel nu de tijdslimiet en de vereiste score in om bonustijd te verdienen en bereken de eindtijd van het spel. Dit coderingsfragment komt vlak voor de hoofdloop van het spel.

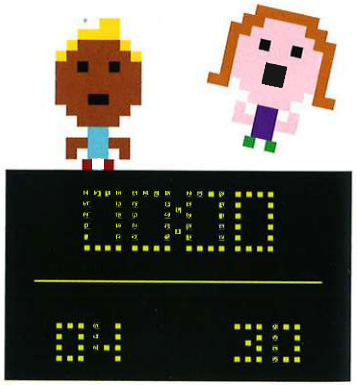
```
from time import sleep, time
BUB_CHANCE = 10
TIME_LIMIT = 30
BONUS_SCORE = 1000
score = 0
bonus = 0
end = time() + TIME_LIMIT
```

Importeert functies vanuit de Time-bibliotheek

Start het spel met een tijdslimiet van 30 seconden

Bepaalt wanneer bonustijd wordt gegeven (als een speler 1000 punten heeft gescoord)

Slaat de eindtijd op in een variabele genaamd 'end'



△ **Scorebord**
Scoreborden zijn een prachtige visuele manier om de spelers in één oogopslag te laten zien hoe ze ervoor staan.

16 Updatet de hoofdloop van het spel om de nieuwe scorings- en tijdsfuncties te installeren.

```
#MAIN GAME LOOP
while time() < end:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    score += collision()
    if (int(score / BONUS_SCORE)) > bonus:
        bonus += 1
        end += TIME_LIMIT
    show_score(score)
    show_time(int(end - time()))
    window.update()
    sleep(0.01)
```

Herhaalt de hoofdloop tot het spel voorbij is

Berekent wanneer er bonustijd wordt gegeven

'print(score)' is vervangen door 'show_score(score)', waardoor de score nu in het spelvenster verschijnt

Geeft de resterende tijd weer

Vergeet niet je werk op te slaan

17 Voeg ten slotte een 'GAME OVER'-graphic toe. Die wordt getoond als de tijd voorbij is. Zet dit helemaal onder in je programma.

```
c.create_text(MID_X, MID_Y, \
    text='GAME OVER', fill='white', font=('Helvetica',30))
c.create_text(MID_X, MID_Y + 30, \
    text='Score: '+ str(score), fill='white')
c.create_text(MID_X, MID_Y + 45, \
    text='Bonus time: '+ str(bonus*TIME_LIMIT), fill='white')
```

Zet de afbeelding in het midden van het scherm

Stelt het lettertype in – 'Helvetica' is een goed type voor grote letters

Vertelt je wat je score was

Stelt de tekstkleur in op wit

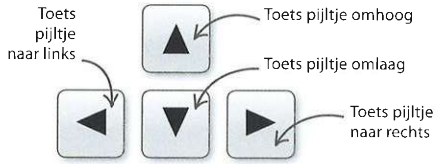
Toont hoeveel bonustijd je hebt verdiend

Vergeet niet je werk op te slaan

BELLENSCHIETER

Tijd om te spelen

Goed gedaan! Je bent klaar met het schrijven van Bellenschieter, dat nu klaar is voor gebruik. Start het programma en probeer het uit. Als iets niet werkt, denk dan aan de debuggingstips – controleer de coderingen op de vorige pagina's goed om uit te sluiten dat je iets verkeerd hebt getypt.



△ **Besturing**
De duikboot wordt bestuurd via de pijltjestoetsen. Je kunt het programma aanpassen, zodat het ook met andere besturingstoetsen werkt.

◁ **Superduikboot**
Je kunt dit spel nu delen met je vrienden. Probeer om de beurt de hoogste score te halen. Na afloop kun je hen de codering achter het spel laten zien en uitleggen hoe het allemaal werkt.

TIPS VAN EXPERTS

Je spel verbeteren

Alle computergames beginnen vanuit een basisidee. Daarna worden ze gespeeld, getest, aangepast en verbeterd. Beschouw dit als een eerste versie van je spel. We geven hier enkele suggesties om je spel te wijzigen en te verbeteren met nieuwe codering:

Maak het spel moeilijker door de tijdslimiet en de score voor bonustijd aan te passen.

Kies een andere kleur voor de duikboot.

Maak een duikboot met meer details.

Maak een speciaal soort bel die de snelheid van de duikboot verhoogt.

Voeg een slimme bom toe die alle bellen verwijdert als je op de spatiebalk drukt.

Bouw een speciaal bord dat de hoogste scores bijhoudt.

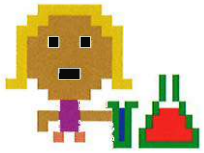
Wat nu?

ZIE OOK
‹ 152–153 Bibliotheken
Computergames
204–205 ›

Nu je de Python-projecten in dit boek hebt uitgevoerd, ben je hard op weg een groot programmeur te worden. Hier volgen enkele ideeën over wat je nog meer kunt doen met Python en hoe je je nog verder kunt ontwikkelen.

Experimenteer

Speel met de coderingsvoorbeelden in dit boek. Ontdek nieuwe manieren om ze te mixen of er nieuwe dingen aan toe te voegen – en wees ook niet bang ze te breken! Dit is je kans om te experimenteren met Python. Onthoud dat het een professionele programmeertaal is met veel mogelijkheden – je kunt er van alles mee doen.



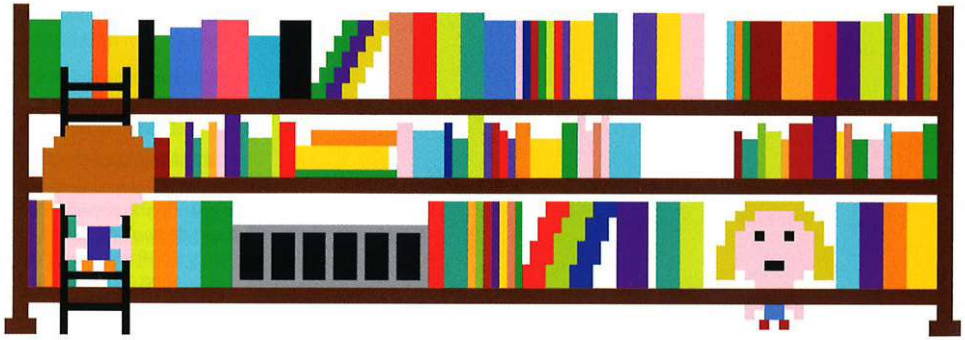
ONTHOUD

Lees veel coderingen

Zoek interessante programma's of bibliotheken van andere mensen en lees de codering en hun commentaar. Probeer te begrijpen hoe de codering werkt en waarom hij zo is geschreven. Hierdoor leer je veel van coderen en zul je telkens stukjes informatie leren over bibliotheken die je in de toekomst kunt gebruiken.

Leg je eigen bibliotheken aan

Programmeurs zijn dol op hergebruik van een codering en delen hun werk graag. Leg je eigen bibliotheek met handige functies aan en deel die ook. Het is een geweldig gevoel als je ziet dat jouw codering door een ander wordt gebruikt. Wie weet bouw je wel zoiets moois als Tkinter of Turtle!



Maak games met Python

Je kunt je eigen game maken met Python. De PyGame-bibliotheek, die je kunt downloaden van internet, bevat heel veel functies en tools waarmee je makkelijker games kunt maken. Begin met eenvoudige spellen en maak ze steeds een beetje moeilijker.

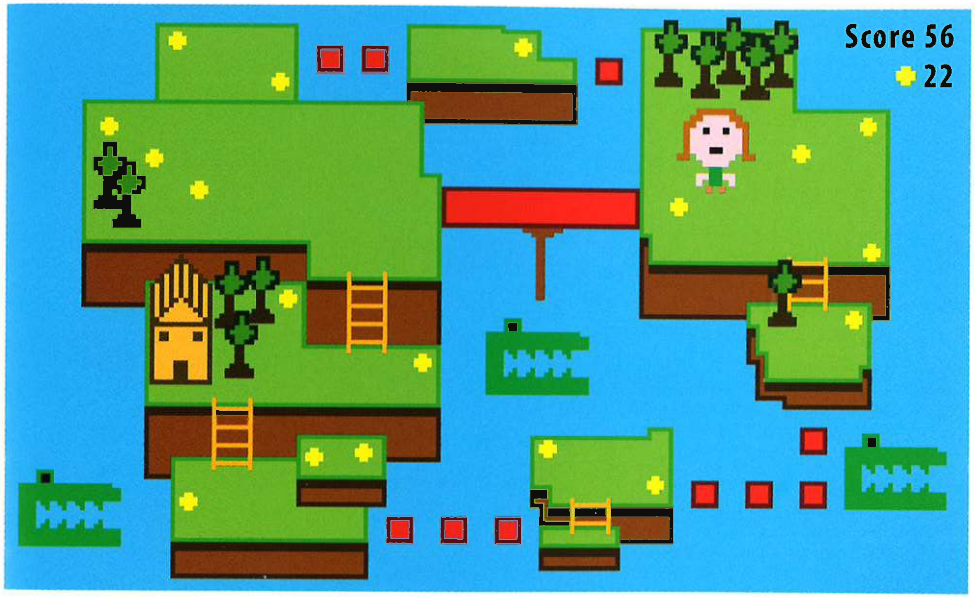
TIPS VAN EXPERTS

Verschillende versies van Python

Als je ergens anders een codering vindt kan die in een andere versie van Python zijn geschreven. De versies zijn vrijwel gelijk, maar je moet misschien wat dingetjes aanpassen.

```
print 'Hello World'          Python 2

print('Hello World')         Python 3
```



Debug je codering

Debugging is een belangrijk onderdeel van programmeren. Geef niet op als iets niet meteen werkt. Onthoud dat computers alleen doen wat ze is opgedragen, dus doorzie de codering en probeer erachter te komen waarom iets niet werkt. Soms helpt het als een andere programmeur meekijkt.