Pak de Kaas

Status: alles uitgevoerd en getest

*Dit is een lessenserie over een programmeerproject met **Python en Pygame Zero** genaamd "Pak de Kaas". Het leidt leerlingen door zes stappen om een eenvoudig spel te maken waarin een muis kaas verzamelt. De lessen behandelen basisconcepten zoals het weergeven en verplaatsen van **afbeeldingen (sprites)**, het detecteren van **botsingen** tussen objecten, het gebruik van **willekeurige getallen** om de kaas te verplaatsen, het bijhouden van een **score** en het toevoegen van een **tijdslimiet** voor een "Game Over" scenario. Het document bevat ook een **docentenhandleiding** met leerdoelen, potentiële valkuilen en suggesties voor differentiatie en beoordeling.*

1 Pak de Kaas – Les 1

datasource

Terugblik

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 pacakge installeert in Thonny?

Yep, Tools - Manage Packages en dan pgzero zoeken en installeren.

Als je het niet meer weer weet kijk dan even naar de vorige les.

datasource

En nu verder.....

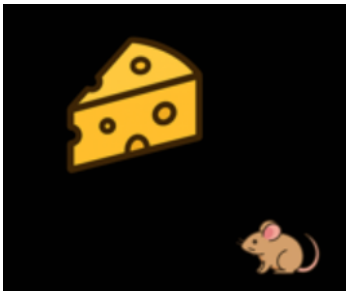
Je hebt pgzero in Thonny geinstalleerd? Nee kijk dan bij de vorige stap.

In deze les gaan we met Python en Pygame Zero leren hoe je plaatjes (sprites) op het scherm kunt zetten.

Je leert hoe je het scherm wist, en hoe je sprites op een bepaalde positie tekent.

Wat gaan we doen?

We gaan een muis en een stuk kaas op het scherm laten verschijnen.



Hiervoor gebruiken we `screen.blit()` in de `draw()`-functie van Pygame Zero.

? Benodigdheden

- Thonny met Pygame Zero geïnstalleerd
- Maak map `images/` met in de folder waar je spel staat.
- Download pak-de-kaas-assets.zip bestand.hierin staat ook een 'giftige kaas' die je later misschien kan gebnruiken.
- Pak het zip bestand uit en plaats de plaatjes in een map `images/` .

? Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

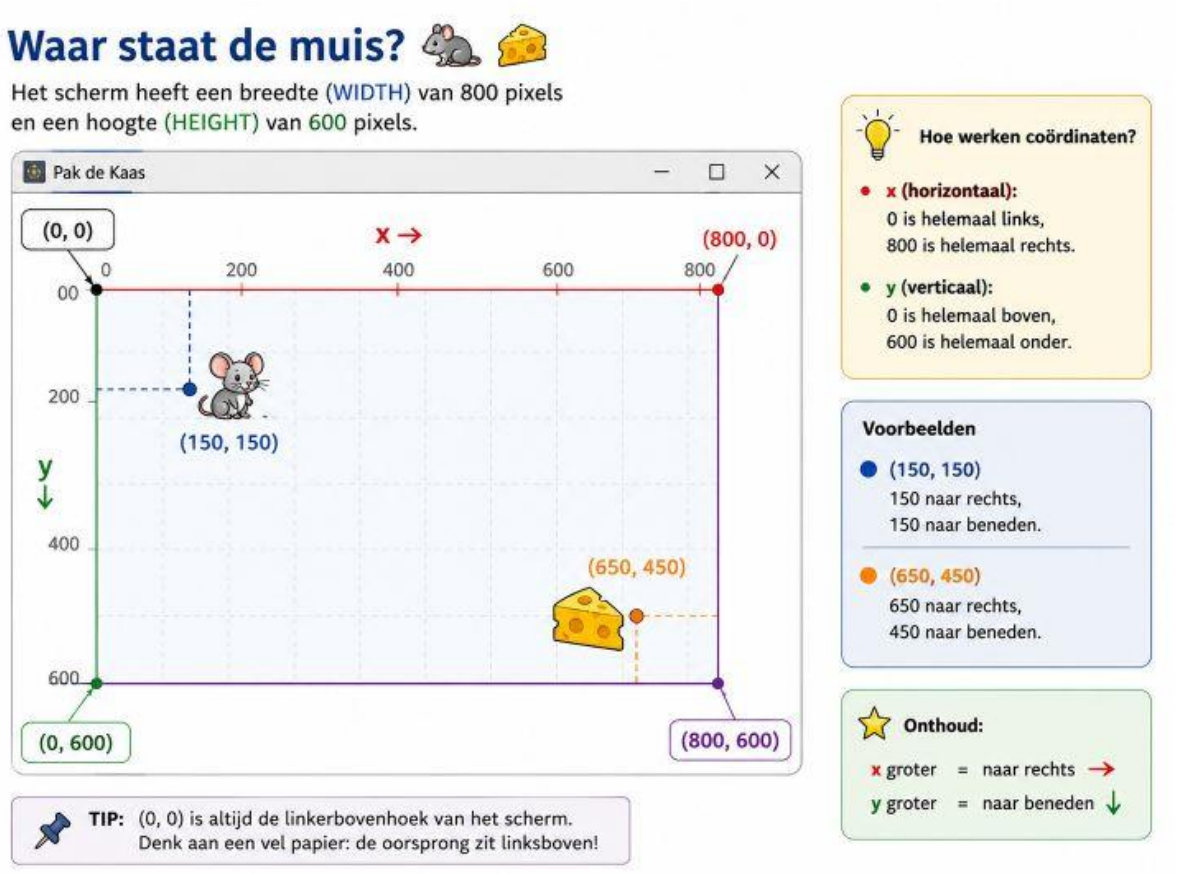
def draw():
    screen.clear()
    screen.blit("mouse", (150, 150))
    screen.blit("cheese", (50, 50))

#start programma
```

```
pgzrun.go()
```

?? Uitleg

- De code wordt van **boven** naar **beneden**, regel voor regel uitgevoerd.
- `import pgzrun` hiermee vertellen we dat de library pgrun gaan gebruiken. Sommige computer-comando's 'sdie straks gaan gebruiken staat beschreven in deze library.
- `WIDTH` en `HEIGHT` bepalen hoe groot het venster is
- `draw()` is een speciale functie die automatisch wordt aangeroepen om het scherm te tekenen
- `screen.clear()` is een library functie die het scherm bij elke frame wist
- `screen.blit("mouse", (150, 150))` tekent het plaatje `mouse.png` op positie (150, 150). Dit zijn ook beide library functies.



?? Opdracht

Pas de coördinaten van de muis en de kaas aan en kijk wat er gebeurt.

- Zet de muis linksboven in beeld
- Zet de kaas rechtonder in beeld

Wat gebeurt er als deze posities gebruikt?

```
    screen.blit("mouse", (50, 50))
    screen.blit("cheese", (30, 30))
```

> Denk na over de volgorde waarin de commando's worden uitgevoerd.

? Inleveren

Maak een screenshot de code met de coordinaten waarbij de muis linksboven in beeld in beeld staat en de kaas rechtsonder.

datasource

2 Beweeg de muis

In deze les gaan we de muis laten bewegen met de pijltjestoetsen.

We doen dat door de positie van de muis aan te passen telkens als een toets wordt ingedrukt.

Wat gaan we doen?

We maken twee variabelen `mouse_x` en `mouse_y` om de positie van de muis bij te houden.

In de functie `update()` passen we deze coördinaten aan als je een pijl indrukt.

? Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

# Startpositie van de muis
```

```
mouse_x = 150
mouse_y = 150

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (50, 50))

def update():
    global mouse_x, mouse_y
    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

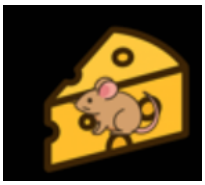
#start programma
pgzrun.go()
```

?? Uitleg

- `mouse_x` en `mouse_y` zijn variabelen die bijhouden waar de muis staat
- `update()` wordt meerdere keren per seconde uitgevoerd
- Met `keyboard.left` controleer je of de linkerpijl wordt ingedrukt
- Bij elke toetsdruk wordt de positie een klein stukje aangepast

?? Opdracht

- Voer de code uit en beweeg de muis met de pijltjestoetsen
- Pas de waarde `5` aan naar een groter of kleiner getal. Wat merk je?
- Probeer ervoor te zorgen dat de muis niet buiten het scherm kan verdwijnen (zie extrat uitleg).
- Als de muis en de kaas op dezelfde positie staan dan verdwijnt de muis onder de kaas. Zorg ervoor dat de muis boven de kaas komt. Zoals hier is weegegven:



Extra uitleg

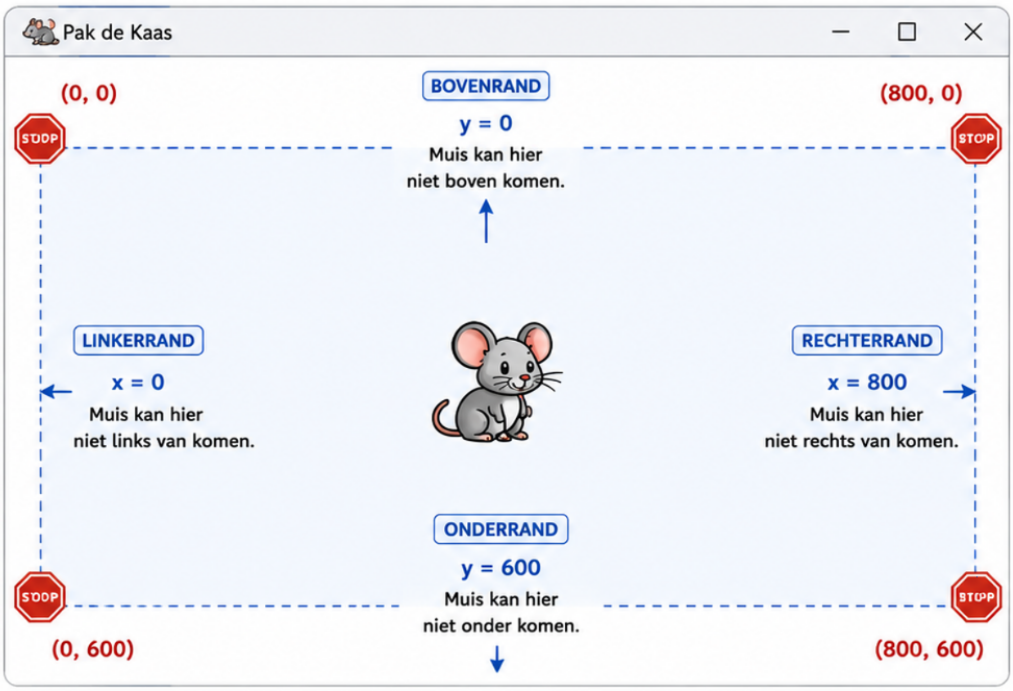
Laat de muis **niet** verder bewegen als hij het scherm uit dreigt te gaan. Voeg hiervoor `if`-statements toe zoals:

```
if keyboard.left and mouse_x > 10:
    mouse_x -= 5
```

Dus hier staat: las de linker pijtjes toets is ingedrukt én de x coördinaat is > 10 zet de postie van de muis dan op de huidige positie - 5. Dus trek 5 van de huisdige posite af.

Doe dit in vier stappen en test **elke** stap.

1. Doe dit eerst voor keyboard.left en test of het werkt.
2. Doe dit ook voor de keyboard.right en test of het werkt.
3. Doe dit daarna ook voor de keyboard.up en test of het werkt.
4. Doe dit tenslotte voor de keyboard.down en test of het werkt.



? Inleveren

Maak een screenshot van je aangepaste code waarbij de **muis boven de kaas** beweegt én waarbij de muis **niet uit het scherm** kan bewegen.

3 Botsing met de kaas

In deze les gaan we kijken of de muis de kaas aanraakt.

Daarvoor gebruiken we een `if`-statement en controleren we of de muis en de kaas elkaar overlappen.

Wat gaan we doen?

We maken een rechthoek rondom de muis en rondom de kaas, en gebruiken `colliderect()` om te kijken of ze elkaar raken.

Als de muis de kaas raakt, tonen we een bericht in de console met `print()`.

? Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

# Startpositie van de muis en kaas
mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))

def update():
    global mouse_x, mouse_y
```

```
    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    # Botsing controleren
    mouse_rect = Rect((mouse_x, mouse_y), (50, 50))
    cheese_rect = Rect((cheese_x, cheese_y), (100, 100))

    if mouse_rect.colliderect(cheese_rect):
        print("Gevonden!")
    else:
      print('-')

#start programma
pgzrun.go()
```

?? Uitleg

- Een `Rect` is een rechthoek: (x, y, breedte, hoogte)
- `colliderect()` geeft `True` als twee rechthoeken elkaar raken
- In dit voorbeeld zijn muis 50x50 pixels en de kaas beide 100x100 pixels groot.
- Als er een botsing is, toont Python het woord `Gevonden!`

?? Opdracht

- Probeer de muis met de pijltjestoetsen naar de kaas te bewegen
- Als je `Gevonden!` ziet in de console, werkt de botsing
- De kaas is niet helemaal 100x100 (in alle richtingen). Pas de grootte van de `Rect` aan zodat je alleen een melding "Gevonden!" krijgt als de muis duidelijk de kaas raakt en duidelijk 'op de kaas zit'.
- Zorg er ook voor dat de muis **op** de kaas komt en niet eronder!

⬜⬜ Extra uitdaging

Toon een boodschap op het scherm als de muis de kaas heeft gevonden:

```
gevonden = False


def update():
    ...
    if mouse_rect.colliderect(cheese_rect):
        gevonden = True


def draw():
    ...
    if gevonden:
        screen.draw.text("Gevonden!", (350, 10), fontsize=60, color="red")
```

? Inleveren

Maak een screenshot van je console waarin je ziet dat "Gevonden!" verschijnt als de muis de kaas aanraakt.

4 Kaas verspringt

In deze les gaan we ervoor zorgen dat de kaas naar een nieuwe plek springt als de muis hem aanraakt.

We doen dat met de functie `random.randint()` om een willekeurige positie te kiezen.

Wat gaan we doen?

We importeren de `random`-bibliotheek en maken een functie die een nieuwe plek kiest voor de kaas.

Als er een botsing is, roepen we die functie aan en verplaatsen we de kaas.

? Startercode

```
import pgzrun
import random
```

```
WIDTH = 800
HEIGHT = 600

mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()

#start programma
pgzrun.go()
```

?? Uitleg

- `import random` zorgt ervoor dat we willekeurige getallen kunnen gebruiken
- `random.randint(a, b)` geeft een willekeurig getal tussen `a` en `b`
- `WIDTH - 64` zorgt ervoor dat het plaatje niet buiten beeld komt
- Als de muis de kaas raakt, wordt `cheese_x` en `cheese_y` aangepast

?? Opdracht

- Laat de muis de kaas aanraken en kijk of deze naar een nieuwe plek verspringt
- Probeer het een paar keer en kijk of het altijd binnen het scherm blijft
- Pas de afmetingen van de sprite aan als jouw plaatjes groter of kleiner zijn dan 64x64

◫ Extra uitdaging

Kies een leuk geluidseffect op : https://www.wavsource.com/sfx/sfx.htm

◫ Zet dan ook een bestand, bijvoorbeeld `bloop_x.wav` in de (nieuwe) map `sounds/`

Voeg een geluid toe dat afspeelt als de muis de kaas raakt:

```
if mouse_rect.colliderect(cheese_rect):
    sounds.blook_x.play()
    cheese_x, cheese_y = nieuwe_kaas_plek()
```

? Inleveren

Maak een screenshot van je werkende code waar de kaas verspringt.

Leg kort uit wat `random.randint()` doet en waarom je `WIDTH - 64` gebruikt.

5 Score bijhouden

In deze les gaan we een score bijhouden: elke keer als de muis de kaas raakt, telt de score één punt op.

Deze score tonen we ook op het scherm.

Wat gaan we doen?

We maken een variabele `score` die begint op 0 en steeds verhoogd wordt bij een botsing.

In de `draw()`-functie tekenen we de score linksboven in het scherm.

? Startercode

```
# importeer library
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50
score = 0

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))
    screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color="white")

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y, score

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
```

```
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()
        score += 1

#start programma
pgzrun.go()
```

?? Uitleg

- `score = 0` zet de score aan het begin op nul
- Elke keer als de muis de kaas aanraakt, wordt de score verhoogd met `score += 1`
- `screen.draw.text()` toont tekst op het scherm

?? Opdracht

- Speel het spel een paar keer en kijk of de score steeds verder oploopt
- Pas de tekstkleur of positie aan van de score
- Maak de tekst groter of kleiner door `fontsize` aan te passen

□□ Extra uitdaging

Laat de kleur van de tekst veranderen bij een bepaalde score:

```
kleur = "black"
if score >= 5:
    kleur = "red"
screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color=kleur)
```

? Inleveren

Maak een screenshot van het spel waarbij de score zichtbaar is (minimaal 3 punten).

Leg in een zinnetje uit wat `score += 1` betekent.

6 Tijdslimiet

In deze les gaan we een tijdslimiet toevoegen. De speler heeft bijvoorbeeld 30 seconden om zoveel mogelijk kaas te pakken.

Aan het einde tonen we “Game Over” en stoppen we het spel.

Wat gaan we doen?

We maken een teller `tijd_over` die elke seconde met 1 omlaag gaat. Als de tijd op is, stopt het spel.

We tonen de tijd linksboven in beeld naast de score.

? Startercode

```
# importeer library
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50
score = 0
tijd_over = 30
game_over = False

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y
```

```
def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))
    screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color="white")
    screen.draw.text(f"Tijd: {tijd_over}", (10, 50), fontsize=40, color="blue")
    if game_over:
        screen.draw.text("Game Over", center=(WIDTH//2, HEIGHT//2), fontsize=60, color="red")

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y, score

    if game_over:
        return

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()
        score += 1

def verlaag_tijd():
    global tijd_over, game_over
    if tijd_over > 0:
        tijd_over -= 1
    if tijd_over == 0:
        game_over = True
```

```
clock.schedule_interval(verlaag_tijd, 1.0)


#start programma
pgzrun.go()
```

?? Uitleg

- `tijd_over` begint op 30 (seconden)
- `clock.schedule_interval(verlaag_tijd, 1.0)` zorgt ervoor dat elke seconde de functie `verlaag_tijd()` wordt aangeroepen
- Als `tijd_over` op 0 staat, verandert `game_over` in `True` en stopt het spel
- In de `draw()`-functie tonen we de resterende tijd en, als het spel voorbij is, de tekst “Game Over”

?? Opdracht deel 1

- Laat het spel lopen en probeer zoveel mogelijk punten te halen binnen de tijd
- Pas de tijd aan naar 10 of 60 seconden – wat vind je leuker?
- Laat bij “Game Over” ook de eindscore groter in beeld zien.

??Opdracht deel 2 (opnieuw uitvoeren)

- Zorg ervoor dat de **muis niet uit het beeld** kan worden bewogen (zoals we bij opgave 2 hebben gedaan).
- Zorg ervoor dat de muis **op de kaas** en niet onder de kaas verdwijnt.

□□ Extra uitdaging

Voeg een herstart-mogelijkheid toe met de `R`-toets:

```
def on_key_down(key):
    global score, tijd_over, game_over, mouse_x, mouse_y, cheese_x, cheese_y
    if key == keys.R:
        score = 0
        tijd_over = 30
        game_over = False
```

```
mouse_x, mouse_y = 150, 150
cheese_x, cheese_y = nieuwe_kaas_plek()
```

? Inleveren

1. Maak een screenshot van het spel als de tijd op is en je “Game Over” ziet en zet een mooie score neer!
2. Bewaar je code als een bestand (in Thonny, file - save as...) en lever het .py bestand met de code in.

7 Eindopdracht

? Eindopdracht

Kies één (of meerdere) van de volgende uitbreidingen en voeg die toe aan je spel:

- □□ Zet meerdere stukjes kaas tegelijk op het scherm (gebruik een `for`-loop)
- □□ Voeg een “giftige” kaas toe: als je die pakt, verlies je punten
- □□ Voeg geluiden toe voor eten, botsing of game over
- □□ Verander het uiterlijk van de muis of de achtergrond
- □□ Laat het spel moeilijker worden naarmate de tijd verstrijkt (bijv. snellere muis of bewegende kaas)

□□ Bedenk zelf ook een uitbreiding? Schrijf het plan op, laat het goedkeuren door je docent en priobeer het te maken!

> Bij deze opgave mag je AI gebruiken!

? Inleveren

Lever de volgende dingen in:

1. Een werkend Python-bestand (.py) waarin jouw uitbreiding is verwerkt
2. Een screenshot van je spel én;

3. Uitleg in tekst (.txt) wat je hebt gedaan.

8 Reflectie

In deze les kijk je terug op wat je hebt gemaakt, én krijg je de kans om je spel op een leuke manier uit te breiden of aan te passen.

? Reflectie

Als je terugkijkt naar deze opgave.

Vraag je zelf de volgende dingen af:

- Wat vond je het leukste om te doen?
- Wat vond je moeilijk of lastig om te begrijpen?
- Welke onderdelen van Python begrijp je nu beter?
- Waar ben je trots op.

? Inleveren

1. Lever je reflectie in met de antwoorden op de (reflectie) vragen in een PDF bestand.

> Let op: gebruik je eigen woorden en wees specifiek!

! Docentenhandleiding

? Overzicht

- **Doelgroep:** Leerlingen van 12-15 jaar (instapniveau Python)
- **Duur:** 6 lessen van ±45-60 minuten
- **Software:** Thonny + Pygame Zero

- **Spelconcept:** Een muis beweegt met de pijltjestoetsen en pakt steeds opnieuw verschijnende kaasjes

? Leerdoelen

- Begrijpen hoe coördinaten werken in een 2D-scherm
- Gebruik van `draw()` en `update()` in een animatie
- Werken met variabelen voor positie en beweging
- Detecteren van botsingen met `Rect` en `colliderect()`
- Gebruik van `random` en herhaalde logica
- Score bijhouden en tonen
- Reflecteren op eigen code en uitbreidingen bedenken

? Lesoverzicht

Les	Onderwerp	Nieuwe concepten
1	Muis en kaas tekenen	`draw()`, `screen.blit()`, coördinaten
2	Muis beweegt met toetsen	`keyboard.left`, `update()`, grenzen
3	Botsing detecteren	`Rect()`, `colliderect()`
4	Kaas verspringt op random plek	`random.randint()`, logica
5	Score bijhouden	`score += 1`, `screen.draw.text()`
6	Reflectie & uitbreiden	Reflecteren, creatief uitbreiden

?? Valkuilen

- `colliderect()` werkt niet → verkeerde grootte/positie van Rect
- Sprites bewegen niet vloeiend → `update()` mist `force_redraw()` (indien nodig)
- Kaas verschijnt buiten het scherm → randomwaarden buiten bereik
- Score telt verkeerd → `score += 1` buiten juiste `if`-blok

? Differentiatie

Voor snelle leerlingen

- Voeg meerdere stukjes kaas toe tegelijk
- Laat een “slechte” kaas verschijnen die de score verlaagt
- Gebruik afbeeldingen en laat muis draaien in richting

Voor langzamere leerlingen

- Werk eerst zonder `Rect`, laat botsing handmatig triggeren
- Gebruik alleen horizontale beweging
- Geef startscripts per les met al werkende onderdelen

? Beoordeling (optioneel)

Criterium	Omschrijving	Score (1-5)
Spel werkt	Muis beweegt, kaas wordt gepakt, score telt	☐☐
Codekwaliteit	Variabelen zijn logisch, overzichtelijk	☐☐
Creativiteit	Leerling heeft iets extra’s toegevoegd (geluid, extra levels, design)	☐☐
Reflectie	Leerling beantwoordt de reflectievragen met inzicht	☐☐

? Lesaanpak & tips

- Laat leerlingen direct runnen en testen na elke wijziging
- Gebruik klassikale codebesprekingen met testvoorbeelden
- Moedig leerlingen aan om “stukjes code” zelf te veranderen
- Laat ze uitleggen wat ze doen: peer programming werkt goed bij deze opdracht

? Benodigdheden

- Thonny + Pygame Zero geïnstalleerd
- Afbeeldingen: muis.png, cheese.png (optioneel)
- Toetsenbord met pijltjestoetsen
- Basiskennis Python (variabelen, `if`, `def`)

Revision #39
Created 2025-05-21 20:15:35 UTC by Max
Updated 2026-07-05 08:41:26 UTC by Max