# Python 3 - Game of Pong Challenge

*(moet nog uitgewerkt worden; samenwerkingsopdracht?)*

## Samenwerken

Deze Challenge heeft 5 opdrachten die allemaal moeten worden gemaakt.

- Iedereen maakt ten minste één opdracht en je helpt elkaar.
- Iedereen moet de code snappen en kunnen uitleggen.
- Jullie presenteren samen de complete code en laten zien hoe het werkt.

De opdrachten zijn deze keer in het Engels, maar je kan deze natuurlijk laten vertalen door AI.

# *1 - Advanced Ball Movement*

Now let's make the movement of the ball more interesting. You should make the ball go faster when it hits the paddle, and randomly change the direction of the ball within a small range.

## Instructions

1. Ensure that the speed of the ball increases by 5% each time it hits a paddle.
2. Make the ball have a small random variation in angle when it hits a paddle to make the game more unpredictable.
3. Test your code to check if the ball responds correctly.

## Deliverables

1. The code (.py)
2. A short explanation, in a.txt file, about how you implemented the implementation of the variations in speed and angle.

In this .txt file, you specify the members of the team and the author(s) of the modified code.

# *2 - Score Multiplier and Bonus Points*

## Introduction

In this assignment, we'll introduce a score multiplier feature. If a player hits the ball multiple times without missing, their score will increase faster. This concept is similar to a combo multiplier in many games, encouraging players to keep the ball in play to maximize their score.

## Task

1. **Multiplier Logic:**
   - Create a multiplier variable starting at 1.
   - Every time the ball hits the paddle without the player missing, increase the multiplier by 0.5 (up to a maximum of 3x).
   - When the player misses, reset the multiplier to 1.

2. **Update Score Function:**
   - Modify the `update_score()` function to display the current multiplier alongside the score.

### Example Code Snippet:

```
# Initialize the multiplier
multiplier = 1.0

# Update the score function to include the multiplier
def update_score():
    pen.clear()
    pen.write(f"Score: {score} Levens: {lives} Multiplier: {multiplier}x", align="center",
font=("Courier", 24, "normal"))
```

```
# Update multiplier in the game loop
if hit_paddle:
    multiplier = min(multiplier + 0.5, 3)
    score += int(1 * multiplier)
else:
    multiplier = 1
```

# Deliverable

1. A .txt document
2. A short description of a few lines in a.txt file, about how your code works.
   In this .txt file you specify the members of the Team and the author(s) of the modified code.

# *3 - Handling Game State and Pausing*

## Introduction

Managing game state is crucial in game development. In this assignment, you'll implement a pause functionality, allowing players to pause and resume the game.

## Task

1. ### Game State Variable:
   - Introduce a `game_state` variable that can be either "playing" or "paused".

2. ### Pause and Resume Functions:
   - Create functions to pause and resume the game. When paused, the game loop should not update the ball or paddle positions.
   - Bind a keyboard key (e.g., "p") to toggle between "playing" and "paused".

### Example Code Snippet:

```
game_state = "playing"

def toggle_pause():
    global game_state
    if game_state == "playing":
        game_state = "paused"
    else:
        game_state = "playing"

wn.listen()
wn.onkeypress(toggle_pause, "p")

while True:
    if game_state == "playing":
        # Game logic updates
        wn.update()
    else:
        # Game is paused; do not update game logic
        pass
```

## Deliverable

1. A `.py` file with the pause functionality implemented. Include a comment explaining how the pause feature is beneficial in games.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

# *4 - Power-Ups and Advanced Features*

## Introduction

Adding power-ups can make games more engaging. In this assignment, you'll introduce a new feature: power-ups that appear randomly and affect the game when collected.

## Task

1. ## Power-Up Implementation:

   - Create a new turtle object representing a power-up. Randomly place it on the screen every 20 seconds.
   - If the ball hits the power-up, apply a random effect such as increasing the paddle size, slowing down the ball, or adding an extra life.

2. ## Effect Duration:

   - Ensure that power-up effects last only for a certain period (e.g., 10 seconds), after which the game returns to normal.

## Example Code Snippet

```
import random

def spawn_power_up():
    power_up = turtle.Turtle()
    power_up.shape("circle")
    power_up.color("blue")
    power_up.penup()
    x = random.randint(-350, 350)
    y = random.randint(-250, 250)
    power_up.goto(x, y)
    return power_up

def apply_power_up(effect):
    global paddle, ball
    if effect == "increase_paddle":
        paddle.shapesize(stretch_wid=8)
        # Reset after 10 seconds
        wn.ontimer(lambda: paddle.shapesize(stretch_wid=6), 10000)
    elif effect == "slow_ball":
        ball.dx *= 0.5
        ball.dy *= 0.5
        wn.ontimer(lambda: reset_ball_speed(), 10000)

# Example effect application in the game loop
```

```
if ball.distance(power_up) < 20:
    apply_power_up(random.choice(["increase_paddle", "slow_ball"]))
```

# Deliverable

1. A `.py` file with the power-up feature implemented. Include comments explaining each power-up and its impact on the game.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

# *5 - Collision Detection and Physics*

## Introduction

Improving collision detection can make a game feel more realistic. In this assignment, we'll refine the collision detection to account for the ball's velocity and angle, providing a more physics-based gameplay experience.

## Task

1. ### Advanced Collision Detection:
   - Adjust the ball’s direction based on where it hits the paddle (top, middle, or bottom). If it hits the top, it should bounce off at a sharper angle; if it hits the middle, it should bounce back more vertically.

2. ### Implement Angled Bounces:
   - Modify the game so that when the ball hits the paddle near its edges, it alters its `dx` and `dy` to create an angled bounce.

### Example Code Snippet

```
def check_collision():
    global ball, paddle
    if ball.distance(paddle) < 50 and ball.xcor() < -340:
```

```
        # Calculate where the ball hit the paddle
        hit_pos = ball.ycor() - paddle.ycor()
        ball.dy = hit_pos * 0.05
        ball.dx *= -1
```

# Deliverable

1. A `.py` file with advanced collision detection and physics implemented. Include a brief explanation of how physics enhances gameplay realism.
2. In this .py file you specify the members of the team and the author(s) in comments. Place this at the top of your code.

--

---

Revision #15
Created 2024-08-25 12:49:40 UTC by Max
Updated 2024-08-25 18:48:45 UTC by Max