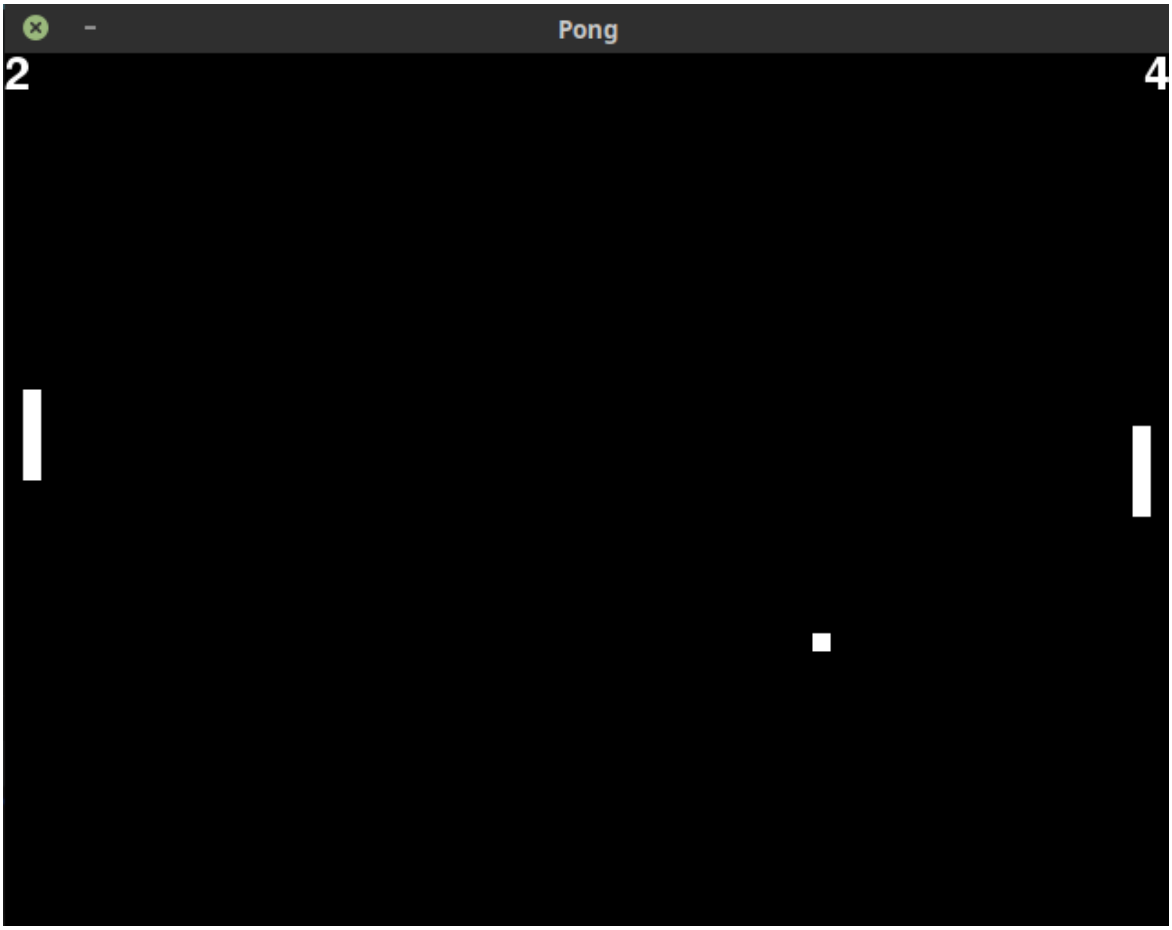
Pong

Mattias Salo

10 juli 2018



Innehåll

1 Introduktion

Pong kom 1972 och var det första kommersiellt framgångsrika arkadspelet, där en fyrkant föreställande en boll ska bollas över nätet mellan två spelare som är utrustade med varsitt racket. Pong blev en succé och Atari, som tillverkade spelet, blev USA:s då snabbast växande företag.

2 Sätt upp projektet

Det första som ska göras är att bestämma alla de värden som vi ska använda som en bas att bygga vidare på. Detta är värden som inte ändras, såkallade konstanter (i Python skriver vi konstanter med stora bokstäver och understreck). Vi börjar med storleken på fönstret som spelet visas i, en bra storlek är 640 pixlar bred och 480 pixlar hög. Vi använder oftast engelska namn på de värdena, eftersom Python är gjort för att användas på engelska. Window betyder fönster, width betyder bredd, och height betyder höjd.

```
WINDOW_WIDTH = 640
WINDOW_HEIGHT = 480
```

Vi vill även sätta ett värde på hur många gånger i sekunden som bildskärmen ska uppdateras (eftersom en rörlig bild egentligen är många bilder ändras väldigt snabbt). För att göra att det ser ut som att bollen rör på sig väldigt smidigt väljer vi ett värde på 60, den hastighet som många vanliga bildskärmar uppdaterar sina bilder med.

```
FPS = 60
```

Där FPS står för ”Frames per Second”, eller bilder per sekund på svenska.

Vi vill även lägga till biblioteket som gör det möjligt för oss att göra spelet och faktiskt visa upp det på skärmen, PyGame. Detta görs enklast genom att lägga till de här raderna högst upp i filen.

```
import pygame
import sys
from pygame.locals import *
```

Vilka färger använder spelet? Eftersom spelet är så gammalt så är det bara två färger som används, svart och vit. Färger i PyGame skrivs med tre värden, som beskriver de tre färger som nästan alla skärmar är uppbyggda av, röd, grön och blå. De brukar förkortas RGB efter just de tre färgerna som används. Och hur får man då svart och vit genom att bara använda röd, grön och blå? Jo, det är så enkelt att svart är när ingen av de tre färgerna finns, och eftersom svart på engelska heter black blir konstanten för svart.

```
BLACK = (0, 0, 0)
```

Men hur är det med vit? Vit är motsatsen till svart, det är när alla färger är ställda till max, vilket i det här fallet är 255. Därför ser konstanten för vit ut så här.

```
WHITE = (255, 255, 255)
```

De här konstanterna lägger vi efter alla andra konstanter.

Nu är det dags att skriva lite kod som faktiskt gör någonting. Vi ska skriva den kodsnutten som vi vill ska köras först i hela programmet. För detta vill vi skapa en funktion som heter main, där main i princip betyder huvud, eftersom det är därifrån programmet styrs. I main lägger vi till lite kod för att starta pygame och klockan som håller reda på hur många bilder per sekund som spelet ska visa.

```
def main():
    pygame.init()
    fps_clock = pygame.time.Clock()
    display_surf = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
    pygame.display.set_caption("Pong")

    while True:
        run_game(display_surf, fps_clock)
```

display_surf är här det som vi ska använda för att måla vår spelplan på. pygame.display.set_caption("Pong") är det som gör att det kommer stå Pong på vårt fönster. Där det står while True: är där vår loop för att styra spelet är, run_game(display_surf, fps_clock) (som vi kommer börja skriva på snart) kommer köra ett helt spel av Pong. När den är klar kommer ett nytt spel att börja (eftersom den här loopen är en evighetsloop).

Nu kan vi börja skriva på funktionen run_game(display_surf, fps_clock) som faktiskt ska köra en hel omgång av Pong. Till en början ska vi se till att visa en bakgrund, och i Pong är den ju svart. Vi skriver alltså en funktion som ser ut så här.

```
def run_game(display_surf, fps_clock):
    while True:  # main game loop
        display_surf.fill(BLACK)
        fps_clock.tick(FPS)
```

Vi kan då se att vi även här använder en evighetsloop, men där varje omgång i loopen inte beskriver ett helt spel men istället en bild av spelet. fps_clock.tick(FPS) är den raden kod som gör att precis 60 bilder per sekund ska visas, utan den skulle spelet gå jättesnabbt. display_surf.fill(BLACK) är den raden kod som gör att vi faktiskt ritar upp en svart bakgrund, eller rättare sagt fyller hela skärmen med färgen svart.

Innan vi kan starta spelet är det endast en till kodsnutt som behöver läggas till.

```
if __name__ == "__main__":
    main()
```

Den här kodsnutten ska alltid ligga längst ner i Python-filen, den bestämmer vilken funktion som ska köras när programmet startar. Eftersom vi har sagt att vår main() funktion ska startas först skriver vi in den.

Om du startar filen nu kommer du kunna se svart fönster med titeln Pong, coolt eller hur!

```
"""
A Pong game written in Python using pygame.
Written by Mattias Salo salo.mattias@gmail.com
"""

import pygame
import sys
import math
import time
import random
from pygame.locals import *

FPS = 60
WINDOW_WIDTH = 640
WINDOW_HEIGHT = 480
BALL_SIDE = 10
BALL_SPEED = 5
PADDLE_LENGTH = 50
PADDLE_WIDTH = 10
PADDLE_SPEED = 5
LEFT_PADDLE_X = 10
RIGHT_PADDLE_X = WINDOW_WIDTH - LEFT_PADDLE_X - PADDLE_WIDTH
WINNING_SCORE = 7
SPEED_INC_COUNTER = 4

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
BG_COLOR = BLACK

LEFT = "left"
RIGHT = "right"


def main():
    pygame.init()
    fps_clock = pygame.time.Clock()
    display_surf = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
    pygame.display.set_caption("Pong")

    while True:
        run_game(display_surf, fps_clock)
        show_game_over_screen(display_surf)


def run_game(display_surf, fps_clock):
    """ Runs an entire instance of the game, returns when someone wins. """
    left_score = 0
    right_score = 0
    collision_counter = 0

    ball = random_ball_start()

    paddles = {'left': WINDOW_HEIGHT / 2 - PADDLE_LENGTH / 2,
               'right': WINDOW_HEIGHT / 2 - PADDLE_LENGTH / 2}

    while True:  # main game loop
        display_surf.fill(BG_COLOR)

        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()

        paddles = move_paddles(paddles)
        ball = move_ball(ball)

        # create rectangles for easier collision detection
        ball_rect = pygame.Rect(ball['x'], ball['y'], BALL_SIDE, BALL_SIDE)
        l_paddle_rect = pygame.Rect(LEFT_PADDLE_X, paddles[LEFT], PADDLE_WIDTH,
    PADDLE_LENGTH)
        r_paddle_rect = pygame.Rect(RIGHT_PADDLE_X, paddles[RIGHT], PADDLE_WIDTH,
    PADDLE_LENGTH)
```

```
        collision = collision_detect(ball_rect, l_paddle_rect, r_paddle_rect)

        if collision:
            ball['x_speed'], ball['y_speed'] = handle_paddle_collision(ball, paddles,
    collision)
            collision_counter += 1

        ball['speed'] = BALL_SPEED + int(collision_counter / SPEED_INC_COUNTER)

        if is_point(ball_rect):  # update score if needed
            ball = random_ball_start()
            collision_counter = 0
            if is_point(ball_rect) == LEFT:
                left_score += 1
            else:
                right_score += 1
            time.sleep(2)

        draw_game(display_surf, left_score, right_score, ball_rect, l_paddle_rect,
    r_paddle_rect)

        if left_score == WINNING_SCORE or right_score == WINNING_SCORE:  # game over
            return

        fps_clock.tick(FPS)


def draw_game(display_surf, left_score, right_score, ball_rect, l_paddle_rect,
    r_paddle_rect):
    """ Draws the entirety of the game. """
    font = pygame.font.Font("freesansbold.ttf", 25)
    left_score_surf = font.render("%s" % left_score, True, WHITE)
    left_score_rect = left_score_surf.get_rect()
    left_score_rect.topleft = (0, 0)

    right_score_surf = font.render("%s" % right_score, True, WHITE)
    right_score_rect = right_score_surf.get_rect()
    right_score_rect.topright = (WINDOW_WIDTH, 0)

    display_surf.blit(left_score_surf, left_score_rect)
    display_surf.blit(right_score_surf, right_score_rect)
    draw_ball(display_surf, ball_rect)
    draw_paddle(display_surf, l_paddle_rect)
    draw_paddle(display_surf, r_paddle_rect)
    pygame.display.update()


def move_ball(ball):
    """ Moves the ball one step. """
    new_y = ball['y'] + ball['y_speed']
    if new_y > WINDOW_HEIGHT - BALL_SIDE or new_y < 0:
        ball['y_speed'] *= -1
    ball['y'] += ball['y_speed']
    ball['x'] += ball['x_speed']
    return ball


def draw_ball(display_surf, ball_rect):
    pygame.draw.rect(display_surf, WHITE, ball_rect)


def draw_paddle(display_surf, paddle_rect):
    pygame.draw.rect(display_surf, WHITE, paddle_rect)


def move_paddles(paddles):
    """
    Moves the paddles according to the buttons pressed.
    A and D moves the left paddle, and UP and DOWN moves
    the right.
    """
```

```
    keys = pygame.key.get_pressed()
    if keys[K_w]:
        new_paddle_y = paddles[LEFT] - PADDLE_SPEED
        paddles[LEFT] = new_paddle_y if new_paddle_y > -PADDLE_LENGTH else -
    PADDLE_LENGTH
    elif keys[K_s]:
        new_paddle_y = paddles[LEFT] + PADDLE_SPEED
        paddles[LEFT] = new_paddle_y if new_paddle_y < WINDOW_HEIGHT else
    WINDOW_HEIGHT
    if keys[K_UP]:
        new_paddle_y = paddles[RIGHT] - PADDLE_SPEED
        paddles[RIGHT] = new_paddle_y if new_paddle_y > -PADDLE_LENGTH else -
    PADDLE_LENGTH
    elif keys[K_DOWN]:
        new_paddle_y = paddles[RIGHT] + PADDLE_SPEED
        paddles[RIGHT] = new_paddle_y if new_paddle_y < WINDOW_HEIGHT else
    WINDOW_HEIGHT
    return paddles


def random_ball_start():
    """
    Sets the ball in the middle of the screen and sets a random direction towards
    either
    one of the players.
    """
    angle = random.randint(0, 360)
    while (45 < angle < 135) or (225 < angle < 315):
        angle = random.randint(0, 360)
    x_speed = math.cos(math.radians(angle)) * BALL_SPEED
    y_speed = math.sin(math.radians(angle)) * BALL_SPEED
    return {'x': WINDOW_WIDTH / 2 - BALL_SIDE / 2,
            'y': WINDOW_HEIGHT / 2 - BALL_SIDE / 2,
            'x_speed': x_speed,
            'y_speed': y_speed,
            'speed': BALL_SPEED}


def collision_detect(ball, left_paddle, right_paddle):
    """
    Detects a collision between the paddles and the ball and returns a value
    corresponding to the collision
    or lack of collision.

    :return: RIGHT, LEFT or False
    """
    if ball.colliderect(right_paddle):
        return RIGHT
    if ball.colliderect(left_paddle):
        return LEFT
    return False


def handle_paddle_collision(ball, paddles, collision):
    """
    Changes the direction of the ball depending on which paddle is hit.
    """
    relative_position = (paddles[collision] + (PADDLE_LENGTH / 2)) - (ball['y'] + (
    BALL_SIDE / 2))
    normalised_relative_position = relative_position / (PADDLE_LENGTH / 2)
    bounce = normalised_relative_position * 50
    direction = -1 if collision == RIGHT else 1
    return (direction * math.cos(math.radians(bounce)) * ball['speed'],
            -math.sin(math.radians(bounce)) * ball['speed'])


def is_point(ball):
    """ Detects if a point should be rewarded to a player. """
    if ball.x < 0:
        return RIGHT
    elif ball.x + BALL_SIDE > WINDOW_WIDTH:
        return LEFT
```

```
    else:
        return False


def show_game_over_screen(display_surf):
    """ Shows the game over screen over the game board. """
    font = pygame.font.Font("freesansbold.ttf", 18)
    game_over_font = pygame.font.Font("freesansbold.ttf", 150)
    game_surf = game_over_font.render('Game', True, WHITE)
    over_surf = game_over_font.render('Over', True, WHITE)
    game_rect = game_surf.get_rect()
    over_rect = over_surf.get_rect()
    game_rect.midtop = (WINDOW_WIDTH / 2, 10)
    over_rect.midtop = (WINDOW_WIDTH / 2, game_rect.height + 10 + 25)

    display_surf.blit(game_surf, game_rect)
    display_surf.blit(over_surf, over_rect)
    draw_press_key_msg(display_surf, font)
    pygame.display.update()
    pygame.time.wait(500)
    check_for_key_press()  # clear out any key presses in the event queue

    while True:
        if check_for_key_press():
            pygame.event.get()  # clear event queue
            return


def draw_press_key_msg(display_surf, basic_font):
    """ Shows a message to the player to press a key. """
    press_key_surf = basic_font.render('Press a key to play', True, WHITE)
    press_key_rect = press_key_surf.get_rect()
    press_key_rect.topleft = (WINDOW_WIDTH - 200, WINDOW_HEIGHT - 30)
    display_surf.blit(press_key_surf, press_key_rect)


def check_for_key_press():
    """ Checks whether a key has been pressed and exits if quit has been pressed. """
    if len(pygame.event.get(QUIT)) > 0:
        pygame.quit()
        sys.exit()

    key_up_events = pygame.event.get(KEYUP)
    if len(key_up_events) == 0:
        return None
    if key_up_events[0].key == K_ESCAPE:
        pygame.quit()
        sys.exit()
    return key_up_events[0].key


if __name__ == "__main__":
    main()
```