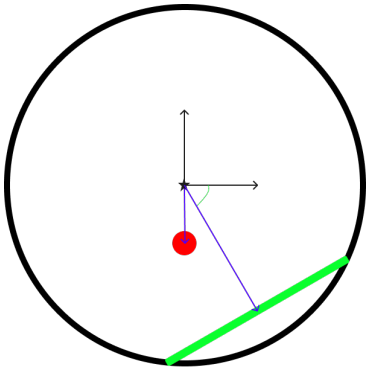
CircleGame

Développement d'un Jeu avec Python, Pygame et Apprentissage d'une IA



Auteur : Fono Colince
Étudiant à l'EILCO
Version : 1.0
Date : 31 octobre 2024

Résumé
Ce tutoriel guide le développement de CircleGame, un jeu utilisant Python et Pygame, intégrant une version jouable par l'humain et une version contrôlée par une intelligence artificielle. L'objectif final est de mettre en œuvre un agent d'apprentissage par renforcement pour automatiser le jeu et explorer les techniques d'IA dans le cadre d'un projet ludique et interactif.

Table des matières

1 Introduction

Ce document présente les étapes pour créer **CircleGame**, un jeu en Python avec Pygame. Le jeu inclura deux versions : une version jouable par l'utilisateur et une version contrôlée par une intelligence artificielle utilisant l'apprentissage par renforcement.

2 Objectif du Jeu

L'objectif de **CircleGame** est de contrôler une barre rotative afin de renvoyer une balle qui rebondit à l'intérieur d'un cercle. Le joueur doit éviter que la balle sorte du cercle tout en marquant des points à chaque rebond réussi contre la barre.

3 Prérequis et Installation

Pour commencer, assurez-vous d'avoir :
— Connaissances de base en Python
— installer Python
— Pygame installé : `pip install pygame`
— Compréhension de base en IA

4 Structure de la Classe

La classe principale du jeu est `CircleGame`. Elle gère l'ensemble de la logique du jeu, y compris le mouvement de la balle, la rotation de la barre, la détection des collisions, et l'affichage à l'écran.

4.1 Code de la Classe CircleGame

Listing 1 – class CircleGame

```
class CircleGame:
    def __init__(self):
        #screen
        # Initialisation des positions et angles

    def rotate_bar(self, direction):
        pass
    def move_ball(self):
        pass
    def calculate_bar_position(self):
        pass
    def check_collision(self):
        # Code de v rification des collisions ici...
        pass
    def is_game_over(self):
        # Code de v rification de la fin du jeu ici...
        pass
```

```
    def play_step(self):
        # Code pour g rer un cycle de jeu ici...
        pass
```

5 Développement de la Version Jouable par l'Humain

5.1 Initialisation et Constantes

Listing 2 – Code pour créer la fenêtre du jeu

```
import pygame
import math
import random
from collections import namedtuple

pygame.init()
font = pygame.font.Font(None, 25)

# Couleurs
WHITE = (255, 255, 255)
RED = (200, 0, 0)
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)

# Dimensions et param tres
SCREEN_WIDTH, SCREEN_HEIGHT = 640, 480
RADIUS = 250  # Rayon du cercle
BAR_LENGTH = 100  # Longueur de la barre
BALL_SPEED = 5  # Vitesse de la balle
ROTATION_SPEED = 5  # Vitesse de rotation de la barre
BARRE_ANGLE = 45  # Angle entre les extr mit s de la barre
GRAVITY= 0.1

class CircleGame:
    def __init__(self):
        self.screen = pygame.display.set_mode((SCREEN_WIDTH,
            ↪ SCREEN_HEIGHT))
        pygame.display.set_caption('Circle Game')
        self.clock = pygame.time.Clock()

        # Initialisation des positions et angles
        self.center = Point(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
        self.bar_angle = 65
        self.ball_position = Point(self.center.x, self.center.y)
        self.ball_velocity = [0, BALL_SPEED * math.sin(math.
            ↪ radians(45))]
        self.score = 0
```

5.2 Mouvement du joueur

Le joueur contrôle la barre rotative en utilisant les touches fléchées gauche et droite du clavier. La barre peut pivoter dans les deux directions, permettant au joueur de mieux positionner la barre pour intercepter la balle.

Listing 3 – Code pour le Mouvement du joueur

```
class CircleGame:

    def rotate_bar(self, direction):
        if self.bar_angle >= 360:
            self.bar_angle = 0
        # Ajustement de  l angle  de la barre
        if direction == 'LEFT':
            self.bar_angle += ROTATION_SPEED
        elif direction == 'RIGHT':
            self.bar_angle -= ROTATION_SPEED
```

5.3 Mécanique de la balle

La balle commence à une position centrée dans le cercle et se déplace avec une vitesse initiale définie. La vitesse verticale de la balle est influencée par la gravité, ce qui lui permet de descendre progressivement. Si la balle atteint le bord du cercle, elle rebondit, inversant ainsi sa direction.

Listing 4 – Code pour créer la fenêtre du jeu

```
class CircleGame:
    def move_ball(self):
        # D placement de la balle
        self.ball_velocity[1] += GRAVITY
        self.ball_position = Point(
            self.ball_position.x + self.ball_velocity[0],
            self.ball_position.y + self.ball_velocity[1]
        )

        # V rification si la balle sort du cercle
        distance_to_center = math.sqrt((self.ball_position.x -
            ↪ self.center.x) ** 2 +
                                       (self.ball_position.y -
                                          ↪ self.center.y) ** 2)
        if distance_to_center > RADIUS + 5:  # 5 est une marge
            self.ball_velocity[0] = -self.ball_velocity[0]
            self.ball_velocity[1] = -self.ball_velocity[1]
```

5.4 Détection de collision

La collision entre la balle et la barre est déterminée par le calcul de la distance entre la balle et la ligne de la barre. Si la balle est suffisamment proche de la barre, sa direction est modifiée en fonction de l'angle d'impact, et le score du joueur augmente. La logique

de rebond est basée sur le produit scalaire du vecteur vitesse de la balle et du vecteur normal à la barre.

Listing 5 – Code pour créer la fenêtre du jeu

```
class CircleGame:
   def calculate_bar_position(self):
        bar_x1 = int(self.center.x + (RADIUS * math.cos(math.
           ↪ radians(self.bar_angle))))
        bar_y1 = int(self.center.y + (RADIUS * math.sin(math.
           ↪ radians(self.bar_angle))))
        bar_x2 = int(self.center.x + (RADIUS * math.cos(math.
           ↪ radians(self.bar_angle + BARRE_ANGLE))))
        bar_y2 = int(self.center.y + (RADIUS * math.sin(math.
           ↪ radians(self.bar_angle + BARRE_ANGLE))))
        return bar_x1, bar_y1, bar_x2, bar_y2

    def check_collision(self):
        bar_x1, bar_y1, bar_x2, bar_y2 = self.
           ↪ calculate_bar_position()

        # Calcul de la distance entre la balle et la ligne de la
           ↪ barre
        distance_to_bar = abs((bar_y2 - bar_y1) * self.
           ↪ ball_position.x -
                              (bar_x2 - bar_x1) * self.
                                 ↪ ball_position.y +
                              bar_x2 * bar_y1 - bar_y2 * bar_x1) /
                                 ↪  math.sqrt((bar_y2 - bar_y1)
                                 ↪ ** 2 + (bar_x2 - bar_x1) ** 2)

        # Si la balle est assez proche pour  tre  en collision
           ↪ avec la barre
        distance_center_ball = math.sqrt((self.ball_position.x -
           ↪ self.center.x)**2 + (self.ball_position.y  - self.
           ↪ center.y)**2)
        ball_p1 = self.distansto((self.ball_position.x,self.
           ↪ ball_position.y),(bar_x1,bar_y1))
        ball_p2 = self.distansto((self.ball_position.x,self.
           ↪ ball_position.y),(bar_x2,bar_y2))

        if distance_to_bar < 30  and distance_center_ball < RADIUS
           ↪ -30 and ball_p1>=10 and  ball_p2>=10 :  # Seuil de
           ↪ collision, ajustable pour plus de pr cision
            # Calcul du vecteur normal de la barre
            bar_dx = bar_x2 - bar_x1
            bar_dy = bar_y2 - bar_y1
            bar_length = math.sqrt(bar_dx**2 + bar_dy**2)
            normal_x = -bar_dy / bar_length
            normal_y = bar_dx / bar_length

            # Produit scalaire pour d terminer la nouvelle
```

```
                ↪ direction
            dot_product = self.ball_velocity[0] * normal_x + self.
                ↪ ball_velocity[1] * normal_y
            self.ball_velocity[0] -= 2 * dot_product * normal_x
            self.ball_velocity[1] -= 2.1* dot_product * normal_y


            self.score += 1
            return True
        return False
```

5.5 Score et Game Over

Le score augmente à chaque fois que la balle rebondit sur la barre. Le jeu se termine si la balle sort du cercle. Lorsqu'un *Game Over* se produit, le score final est affiché, et le programme se termine.

Listing 6 – Code pour créer la fenêtre du jeu

```
class CircleGame:
    def is_game_over(self):
        # V rifier si la balle sort du cercle
        distance_to_center = math.sqrt((self.ball_position.x -
            ↪ self.center.x) ** 2 +
                                       (self.ball_position.y -
                                           ↪ self.center.y) ** 2)
        #return False
        return distance_to_center >= RADIUS-5  # 5 est une marge
```

5.6 Affichage et Interface

Le jeu utilise Pygame pour gérer l'affichage graphique. Le cercle, la barre, et la balle sont dessinés à l'écran, et le score actuel est affiché en haut à gauche. La mise à jour de l'affichage se fait à une fréquence de 60 images par seconde.

Listing 7 – Code pour créer la fenêtre du jeu

```
class CircleGame:
    def play_step(self):
        # Gestion des    vnements
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()

        # Contr le de la rotation de la barre
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            self.rotate_bar('LEFT')
        elif keys[pygame.K_RIGHT]:
            self.rotate_bar('RIGHT')
```

```
        # Mise    jour des positions
        self.move_ball()
        self.check_collision()

        # V rifier si la partie est termin e
        if self.is_game_over():
            print(f"Game Over! Score final: {self.score}")
            pygame.quit()
            exit()

        # Rafra chir l'affichage
        self.screen.fill(BLACK)
        pygame.draw.circle(self.screen, WHITE, (self.center.x,
            ↪ self.center.y), RADIUS, 2)

        # Dessiner la barre
        bar_x1, bar_y1, bar_x2, bar_y2 = self.
            ↪ calculate_bar_position()

        pygame.draw.line(self.screen, GREEN, (bar_x1, bar_y1), (
            ↪ bar_x2, bar_y2), 2)

        # Dessiner la balle
        pygame.draw.circle(self.screen, RED, (int(self.
            ↪ ball_position.x), int(self.ball_position.y)), 8)

        # Afficher le score
        score_text = font.render("Score: " + str(self.score), True
            ↪ , WHITE)
        self.screen.blit(score_text, [10, 10])

        pygame.display.flip()
        self.clock.tick(60)

if __name__ == "__main__":
    game = CircleGame()
    while True:
        game.play_step()
```

6 Implémentation de l'IA avec Apprentissage par Renforcement

6.1 Présentation de l'Apprentissage par Renforcement

6.2 Configuration de l'Environnement de Jeu pour l'Agent

6.3 Implémentation de l'Agent IA

7 Conclusion

Ce tutoriel décrit comment créer CircleGame, en explorant le développement de la version jouable par l'humain et la version contrôlée par une IA, offrant ainsi une expérience interactive et instructive.

8 Références

— Documentation Pygame : `https://www.pygame.org/docs/`
— Documentation Python : `https://docs.python.org/3/`