

DOCUMENTATION POULE RENARD VIPERE

MATTHIEU PELISSIER

Constantes

Les constantes définissent les paramètres principaux du jeu, tels que la taille de la fenêtre, la vitesse des entités, et les dimensions des obstacles.

```
WINDOW_WIDTH = 1200
WINDOW_HEIGHT = 700
ENTITY_RADIUS = 5
SPEED = 3
DETECTION_RADIUS = 100
CONVERSION_RADIUS = 20
EDGE_AVOID_RADIUS = 50
REPULSION_RADIUS = 20
ICON_WIDTH = 30
ICON_HEIGHT = 30
OBSTACLE_WIDTH = 50
OBSTACLE_HEIGHT = 50
```

Entity

La classe Entity représente une entité dans le jeu (serpent, poule ou renard).

Attributs

- **x** : Position x de l'entité.
- **y** : Position y de l'entité.
- **tribe** : Type de l'entité ('snake', 'chicken', 'fox').

- **jail** : Indicateur si l'entité est capturée.
- **frozen** : Indicateur si l'entité est gelée.

Méthodes

- **check_prison_reach()**: Vérifie si une entité a atteint sa zone de prison.
- **move_towards(target_x, target_y)**: Déplace l'entité vers une cible.
- **move_away_from(target_x, target_y)**: Éloigne l'entité d'une cible.
- **distance_to(other)**: Calcule la distance à une autre entité.
- **repel_from(other)**: Repousse l'entité si elle est trop proche d'une autre entité.
- **draw()**: Dessine l'entité à l'écran.
- **avoid_edges_and_restricted_areas()**: Évite les bords de la fenêtre et les zones restreintes.
- **avoid_restricted_area()**: Évite les zones restreintes spécifiques.
- **avoid_obstacles()**: Évite les obstacles.

Obstacle

La classe Obstacle représente un obstacle dans le jeu.

Attributs

x : Position x de l'obstacle.

y : Position y de l'obstacle.

Méthodes

draw(): Dessine l'obstacle à l'écran.

Fonctions

is_within_rect(x, y, rect_left, rect_right, rect_top, rect_bottom): Vérifie si une position est à l'intérieur d'un rectangle.


generate_valid_position(): Génère une position valide pour une nouvelle entité, en évitant les zones restreintes.

Détail du code

Initialisation et chargement des ressources

Importation des bibliothèques


python

 Copier le code

```
import pygame
import random
import math
```

Initialisation de `pygame`

python


 Copier le code

```
pygame.init()
```

Définition des constantes

Les constantes définissent les paramètres principaux du jeu.

python

 Copier le code

```
WINDOW_WIDTH = 1200
WINDOW_HEIGHT = 700
ENTITY_RADIUS = 5
SPEED = 3
DETECTION_RADIUS = 100
CONVERSION_RADIUS = 20
EDGE_AVOID_RADIUS = 50
REPULSION_RADIUS = 20
ICON_WIDTH = 30
ICON_HEIGHT = 30
OBSTACLE_WIDTH = 50
OBSTACLE_HEIGHT = 50
```

Chargement et mise à l'échelle des images

Chargement des images de fond, des icônes des entités et des obstacles.

```
python Copier le code

background_image = pygame.image.load('field.jpg')
background_image = pygame.transform.scale(background_image, (WINDOW_WIDTH, WINDOW_HEIGHT))

snake_icon = pygame.image.load('snake.png')
snake_icon = pygame.transform.scale(snake_icon, (ICON_WIDTH, ICON_HEIGHT))

chicken_icon = pygame.image.load('chicken.png')
chicken_icon = pygame.transform.scale(chicken_icon, (ICON_WIDTH, ICON_HEIGHT))

fox_icon = pygame.image.load('fox.png')
fox_icon = pygame.transform.scale(fox_icon, (ICON_WIDTH, ICON_HEIGHT))

obstacle_image = pygame.image.load('roche.png')
obstacle_image = pygame.transform.scale(obstacle_image, (OBSTACLE_WIDTH, OBSTACLE_HEIGHT))
```

Définition des classes

Classe Entity

La classe `Entity` représente une entité dans le jeu (serpent, poule ou renard).

Constructeur

```
python Copier le code


class Entity:
    def __init__(self, x, y, tribe):
        self.x = x
        self.y = y
        self.tribe = tribe
        self.jail = False
        self.frozen = False
```

Méthodes

`check_prison_reach`

Vérifie si une entité a atteint sa zone de prison.

python


 Copier le code

```
def check_prison_reach(self):
    if self.tribe == 'chicken' and self.frozen == False and is_within_rect(self.x, self.y,
        return True
    if self.tribe == 'fox' and self.frozen == False and is_within_rect(self.x, self.y, rec
        return True
    if self.tribe == 'snake' and self.frozen == False and is_within_rect(self.x, self.y, r
        return True
    return False
```

`move_towards`

Déplace l'entité vers une cible.

python


 Copier le code

```
def move_towards(self, target_x, target_y):
    if not self.frozen:
        angle = math.atan2(target_y - self.y, target_x - self.x)
        self.x += (SPEED + random.uniform(-0.3, 0.3)) * math.cos(angle)
        self.y += (SPEED + random.uniform(-0.3, 0.3)) * math.sin(angle)
        self.avoid_edges_and_restricted_areas()
```

`move_away_from`

Éloigne l'entité d'une cible.

python


 Copier le code

```
def move_away_from(self, target_x, target_y):
    if not self.frozen:
        angle = math.atan2(target_y - self.y, target_x - self.x)
        self.x -= (SPEED + random.uniform(-0.3, 0.3)) * math.cos(angle)
        self.y -= (SPEED + random.uniform(-0.3, 0.3)) * math.sin(angle)
        self.avoid_edges_and_restricted_areas()
```

`distance_to`

Calcule la distance à une autre entité.

python


 Copier le code

```
def distance_to(self, other):  
    return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)
```

`repel_from`

Repousse l'entité si elle est trop proche d'une autre entité.

python


 Copier le code

```
def repell_from(self, other):  
    if self.distance_to(other) < REPULSION_RADIUS:  
        self.move_away_from(other.x, other.y)
```

`draw`

Dessine l'entité à l'écran.

python


 Copier le code

```
def draw(self):  
    if self.tribe == 'snake':  
        screen.blit(snake_icon, (self.x - ICON_WIDTH // 2, self.y - ICON_HEIGHT // 2))  
    elif self.tribe == 'chicken':  
        screen.blit(chicken_icon, (self.x - ICON_WIDTH // 2, self.y - ICON_HEIGHT // 2))  
    else:  
        screen.blit(fox_icon, (self.x - ICON_WIDTH // 2, self.y - ICON_HEIGHT // 2))
```


`avoid_edges_and_restricted_areas`

Évite les bords de la fenêtre et les zones restreintes.

python

 Copier le code


```
def avoid_edges_and_restricted_areas(self):
    if self.x < EDGE_AVOID_RADIUS:
        self.x = EDGE_AVOID_RADIUS
    elif self.x > WINDOW_WIDTH - EDGE_AVOID_RADIUS:
        self.x = WINDOW_WIDTH - EDGE_AVOID_RADIUS
    if self.y < EDGE_AVOID_RADIUS:
        self.y = EDGE_AVOID_RADIUS
    elif self.y > WINDOW_HEIGHT - EDGE_AVOID_RADIUS:
        self.y = WINDOW_HEIGHT - EDGE_AVOID_RADIUS

    self.avoid_restricted_area()
```

`avoid_restricted_area`

Évite les zones restreintes spécifiques.

python

 Copier le code

```
def avoid_restricted_area(self):
    if self.tribe in ['chicken', 'fox']:
        rect1_left = 500
        rect1_right = 700
        rect1_top = 0
        rect1_bottom = 200


        if rect1_left < self.x < rect1_right and rect1_top < self.y < rect1_bottom:
            if self.x < (rect1_left + rect1_right) / 2:
                self.x = rect1_left - EDGE_AVOID_RADIUS
            else:
                self.x = rect1_right + EDGE_AVOID_RADIUS

            if self.y < (rect1_top + rect1_bottom) / 2:
                self.y = rect1_top - EDGE_AVOID_RADIUS
            else:
                self.y = rect1_bottom + EDGE_AVOID_RADIUS
```

``avoid_obstacles``

Évite les obstacles.

python

 Copier le code


```
def avoid_obstacles(self):
    for obstacle in obstacles:
        if self.distance_to(obstacle) < OBSTACLE_WIDTH:
            self.move_away_from(obstacle.x, obstacle.y)
```

Classe Obstacle

La classe ``Obstacle`` représente un obstacle dans le jeu.

Constructeur

python

 Copier le code


```
class Obstacle:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Méthodes

``draw``

Dessine l'obstacle à l'écran.

python

 Copier le code


```
def draw(self):
    screen.blit(obstacle_image, (self.x - OBSTACLE_WIDTH // 2, self.y - OBSTACLE_HEIGHT //
```

Fonctions utilitaires

`is_within_rect`

Vérifie si une position est à l'intérieur d'un rectangle.

python


 Copier le code

```
def is_within_rect(x, y, rect_left, rect_right, rect_top, rect_bottom):  
    return rect_left <= x <= rect_right and rect_top <= y <= rect_bottom
```


`generate_valid_position`

Génère une position valide pour une nouvelle entité, en évitant les zones restreintes.

python

 Copier le code


```
def generate_valid_position():  
    while True:  
        x = random.randint(EDGE_AVOID_RADIUS, WINDOW_WIDTH - EDGE_AVOID_RADIUS)  
        y = random.randint(EDGE_AVOID_RADIUS, WINDOW_HEIGHT - EDGE_AVOID_RADIUS)  
        if (is_within_rect(x, y, 500, 700, 0, 200) or  
            is_within_rect(x, y, 0, 200, 500, 700) or  
            is_within_rect(x, y, 1000, 1200, 500, 700)):  
            continue  
    return x, y
```



Création des entités et des obstacles

Création des instances pour les serpents, poules, renards et obstacles.

python

 Copier le code

```
snakes = [Entity(*generate_valid_position(), 'snake') for _ in range(5)]  
chickens = [Entity(*generate_valid_position(), 'chicken') for _ in range(5)]  
foxes = [Entity(*generate_valid_position(), 'fox') for _ in range(5)]  
obstacles = [Obstacle(*generate_valid_position()) for _ in range(5)]
```