Implementação dos algoritmos de Steering Behaviors propostos por Craig Reynolds

Nome: Augusto C. Setti
Matrícula: 119994

Steering Behaviors

- Segundo passo do comportamento de um Agente Autônomo
- Determina a direção, ou Força, que deve ser aplicada ao Agente para que ele siga um caminho

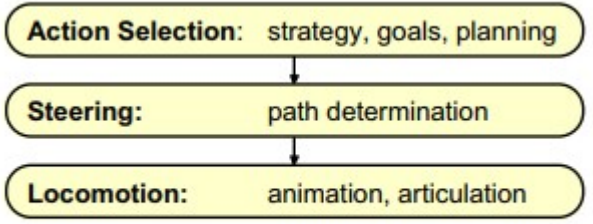


Figure 1: A hierarchy of motion behaviors

Implementação

- Linguagem Python
- Bibliotecas:
    - Pygame para interface
    - Numpy e Math para manipular vetores

A Simple Vehicle Mode

- Atributos do Agente

Simple Vehicle Model:

mass	scalar
position	vector
velocity	vector
max_force	scalar
max_speed	scalar
orientation	*N* basis vectors

```python
class Agent:
    def __init__(self, position) -> None:

        self.origin = np.array(position)
        self.size = 15
        self.color = (random.random() * 128, random.random() * 128, random.random() * 128)

        self.visionratio = 100
        self.position = np.array(position, dtype=float)
        self.velocity = np.array((random.random()*2 -1,random.random()*2 -1))
        self.acceleration = np.zeros(2)
        self.maxforce = .2
        self.maxvelocity= 3
```

A Simple Vehicle Mode

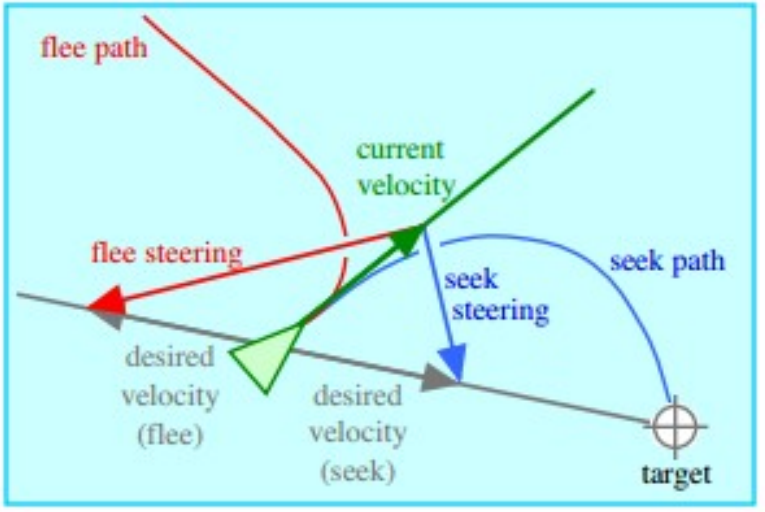
- Física básica aplicada ao Agente

```
steering_force = truncate (steering_direction, max_force)
acceleration = steering_force / mass
velocity = truncate (velocity + acceleration, max_speed)
position = position + velocity
```

```python
def applyForce(self, force):
    '''
    function to apply a force to an agent
    '''
    self.acceleration += force
    self.velocity += self.acceleration
    # limit velocity
    if np.linalg.norm(self.velocity) > self.maxvelocity:
        self.velocity = self.normalizeto(self.velocity,self.maxvelocity)
    self.position += self.velocity
    self.acceleration = (0, 0)
```

Behavior

- Seek and flee



Behavior

- Seek
  - Perseguir um alvo estático

```
desired_velocity = normalize (position -
target) * max_speed
steering = desired_velocity - velocity
```

```python
def seek(self, target_pos=None):
    if not np.any(target_pos != None):
        target_pos = pygame.mouse.get_pos()
    steer = np.zeros(2)
    #difference to target
    desired = np.array(target_pos) - self.position
    #normalization
    if np.linalg.norm(desired) > 0:
        desired = desired/np.linalg.norm(desired)
        #magnitude is maxvelocity
        desired = desired * self.maxvelocity
        #calculate steering force
        steer = desired - self.velocity
        # limit force
        if np.linalg.norm(steer) > self.maxforce:
            steer = self.normalizeto(steer, self.maxforce)
    return steer
```

Behavior

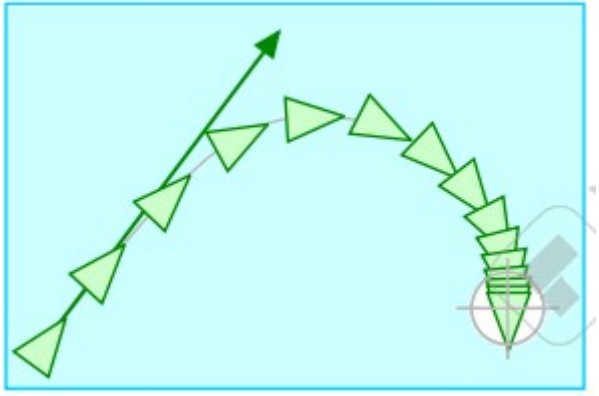
- Flee
  - Conuzir na direção oposto do alvo (Seek^(-1))

```
target_offset = target - position
distance = length (target_offset)
ramped_speed = max_speed * (distance / slowing_distance)
clipped_speed = minimum (ramped_speed, max_speed)
desired_velocity = (clipped_speed / distance) * target_offset
steering = desired_velocity - velocity
```

```python
def flee(self, target_pos=None):
    if not np.any(target_pos != None):
        target_pos = pygame.mouse.get_pos()
    steer = np.zeros(2)
    # difference to target
    difference = np.array(target_pos) - self.position
    #distance
    distance = np.linalg.norm(difference)
    # normalization
    if distance < self.visionratio and distance > 0:
        desired = difference / np.linalg.norm(difference)
        # magnitude is maxvelocity
        desired_speed = self.maxvelocity
        desired = desired * desired_speed
        # calculate steering force
        steer = desired - self.velocity
    return steer*-1
```

Behavior

- Arrival
    - Seek com redução de velocidade ao aproximar do alvo



Behavior

- Arrival

```
target_offset = target - position
distance = length (target_offset)
ramped_speed = max_speed * (distance / slowing_distance)
clipped_speed = minimum (ramped_speed, max_speed)
desired_velocity = (clipped_speed / distance) * target_offset
steering = desired_velocity - velocity
```

```python
def arrive(self, target_pos=None):
    if not np.any(target_pos != None):
        target_pos = pygame.mouse.get_pos()
    steer = np.zeros(2)
    #difference to target
    difference = np.array(target_pos) - self.position
    #normalization
    distance = np.linalg.norm(difference)
    if distance < self.visionratio:
        # normalize
        desired = difference / np.linalg.norm(difference)
        #magnitude is dependant on distance to target
        desired_speed = np.interp(distance,[0,self.visionratio],[0,self.maxvelocity])
        # set magnitude
        desired = desired * desired_speed
    else:
        desired = difference/np.linalg.norm(difference)
        #magnitude is maxvelocity
        desired = desired*self.maxvelocity

    #calculate steering force
    steer = desired - self.velocity
    return steer
```

Behavior

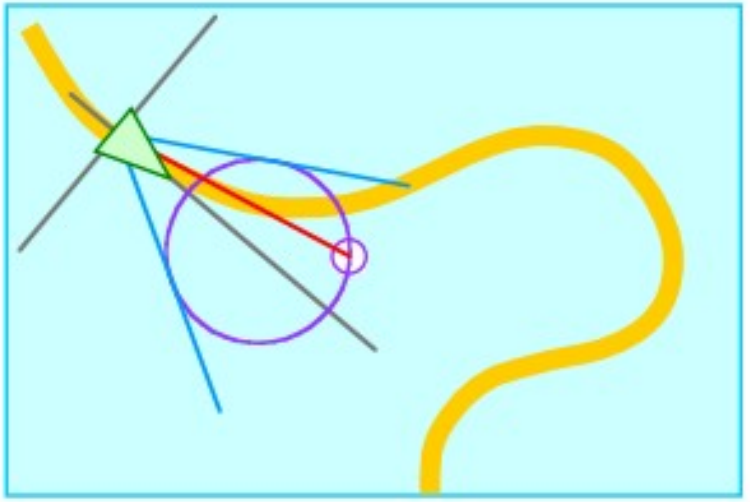
- Wander Simples
    - Agente vaga de forma aleatória podendo mudar de direção bruscamente

```
# for wander
self.target = np.array((random.randint(0, SCREENWIDTH), random.randint(0, SCREENHEIGHT)))
self.last_target = 0
```

```
def wander_simple(self):
    now = pygame.time.get_ticks()
    if now - self.last_target > 1000:
        self.last_target = now
        self.target = self.target = np.array((random.randint(0, SCREENWIDTH), random.randint(0, SCREENHEIGHT)))
    return self.seek(self.target)
```

Behavior

- Wander - Reynolds
    - Agente vaga de forma aleatória, mas não muda bruscamente a direção
    - Direção varia de acordo com a esfera

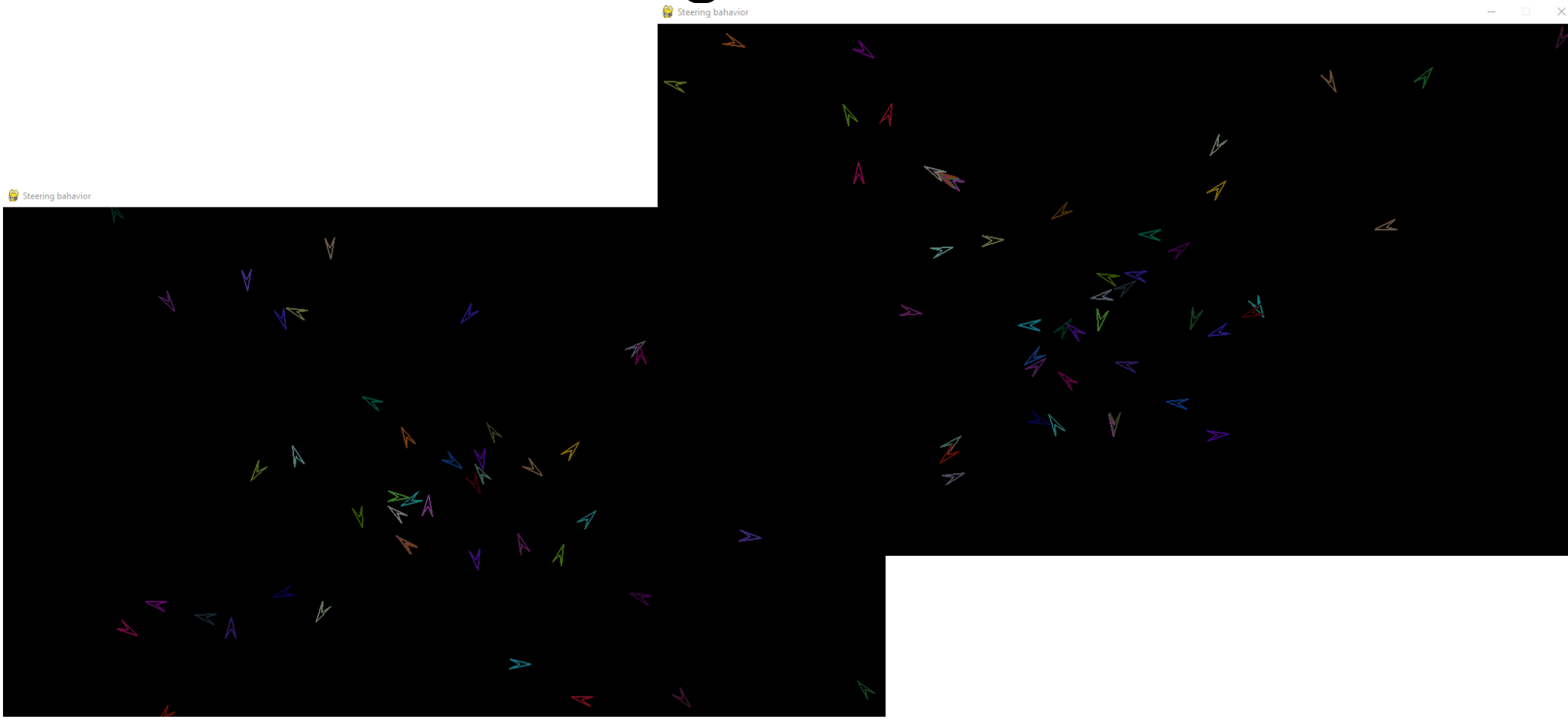


Behavior

- Wander - Reynolds
    - Agente vaga de forma aleatória, mas não muda bruscamente a direção
    - Direção varia de acordo com a esfera

```
def wander(self):
    orientation = self.velocity/np.linalg.norm(self.velocity)
    circle_position = self.position + self.normalizeto(orientation, WANDER_RING_DISTANCE)
    target = circle_position + np.array((random.randint(-WANDER_RING_RADIUS, WANDER_RING_RADIUS),
                                         random.randint(-WANDER_RING_RADIUS, WANDER_RING_RADIUS)))
    return self.seek(target)
```

Steering Behaviors



Steering Behaviors

- Repositório:
  - https://github.com/augustocsetti/steering-behaviors
- Referências:
  - Craig Reynolds website: http://www.red3d.com/cwr/steer/
  - Autonomous Agents and Steering - The Nature of Code: https://www.youtube.com/watch?v=JIz2L4tn5kM
  - Gamedev In-depth: Steering Behaviors (Wander): https://www.youtube.com/watch?v=jz-YNNVlVrQ

Implementação dos algoritmos de Steering Behaviors propostos por Craig Reynolds

Nome: Augusto C. Setti
Matrícula: 119994