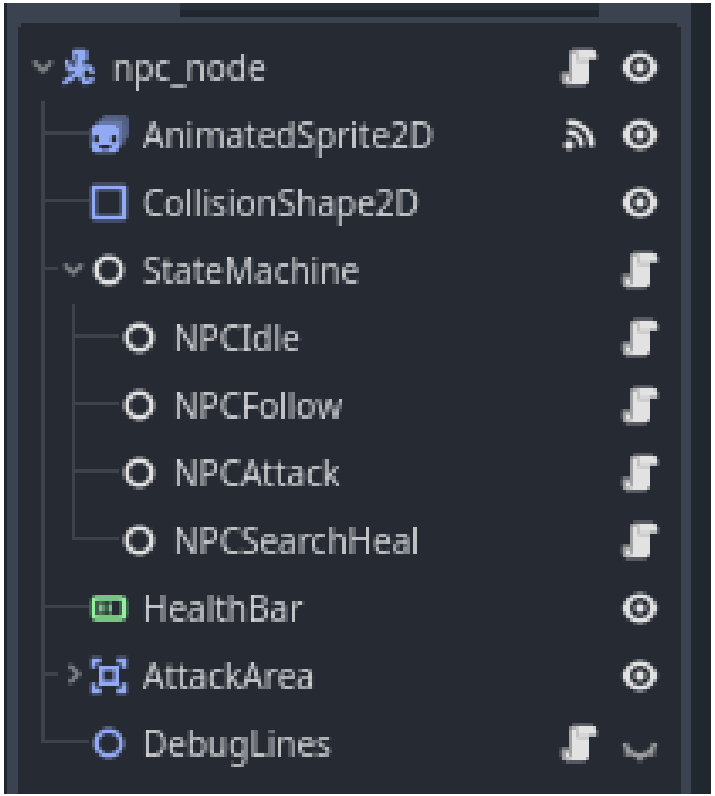
Projeto de IA para jogos – Fase 1

Ricardo Alexandre Rodrigues do Amaral Junior

Máquina de Estados Finitos

- No momento apenas os NPCs, que serão os inimigos do jogo possuem uma máquina de estados finitos.
- A FSM é um node dentro do objeto NPC que possui um script anexado a ele. Esse node possui outros nodes filhos que agem como os estados que compõem a FSM.



Máquina de Estados Finitos

- O node StateMachine inicia com um dicionário vazio e assim que o NPC é instanciado ele verifica todos os estados que são filhos do node StateMachine e os adiciona ao dicionário. Além disso ele também conecta os sinais de transição de todos os objetos do tipo “State”.
- Após isso ele ativa a função de entrada do estado marcado como inicial.

```
1   extends Node
2
3   @export var initial_state : State
4
5   var current_state : State
6   var states : Dictionary = {}
7
8   func _ready():
9       for child in get_children():
10          if child is State:
11              states[child.name.to_lower()] = child
12              child.Transitioned.connect(on_child_transition)
13      if initial_state:
14          initial_state.Enter()
15          current_state = initial_state
```

Máquina de Estados Finitos

- Os outros métodos executam os metodos de “_process” do estado atual.
- O método “_process” é executado uma vez por frame, sempre que possivel. Já o método “_physics_process” é executado a uma taxa fixa de 60 vezes por segundo, independente do framerate.
- O método “on_child_transition” realiza a transição entre estados.

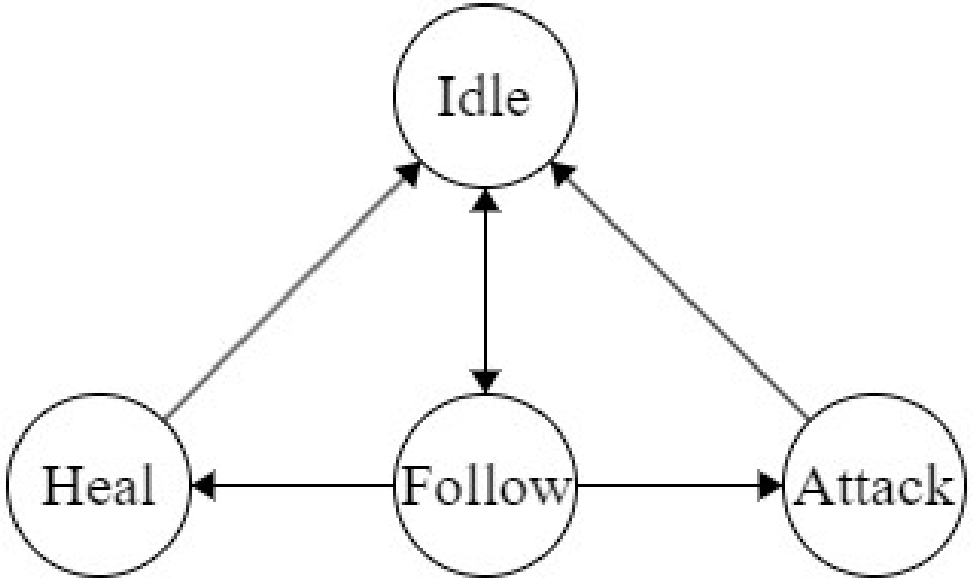
```
17 func _process(delta):
18     if current_state:
19         current_state.Update(delta)
20
21 func _physics_process(delta):
22     if current_state:
23         current_state.PhysicsUpdate(delta)
24
25 func on_child_transition(state, new_state_name):
26     if state != current_state:
27         return
28
29     var new_state = states.get(new_state_name.to_lower())
30     if !new_state:
31         return
32
33     if current_state:
34         current_state.Exit()
35
36     new_state.Enter()
```

Máquina de Estados Finitos

- Cada node de estado herda do código ao lado. Ele possui os métodos necessários para executar códigos quando o NPC entrar e sair do estado e também a cada update.

```
1   extends Node
2   class_name State
3
4   signal Transitioned
5
6   func Enter():
7       pass
8
9   func Exit():
10      pass
11
12  func Update(_delta: float):
13      pass
14
15  func PhysicsUpdate(_delta: float):
16      pass
17
```

- **A máquina atual é bem simples e possui apenas 4 estados para o NPC.**



Máquina de Estados Finitos

- O estado Idle toca uma animação quando o NPC entra nesse estado. A cada update ele calcula a distância entre si e o jogador e quando essa distância for menor que 100 ele transiciona para o estado de Follow.

```gdscript
extends State
class_name NPCIdle

@export var npc: CharacterBody2D
@export var move_speed := 10.0
var player : CharacterBody2D

var move_direction : Vector2

func Enter():
    get_node("../../AnimatedSprite2D").play("idle")
    player = get_tree().get_first_node_in_group(("Player"))

func PhysicsUpdate(delta: float):
    if npc:
        npc.velocity = move_direction * move_speed

    var direction = player.global_position - npc.global_position

    if direction.length() < 100:
        Transitioned.emit(self, "NPCFollow")
```

Máquina de Estados Finitos

- O estado Follow move o NPC em direção ao jogador. Quando essa distância se torna maior que 100 o NPC retorna ao estado Idle. Quando a distância for menor que 20 ele irá passar para o estado de Attack.
- Além disso, nesse estado, caso a saúde do NPC se torne menor que 30, ele irá para para um estado em que irá procurar um item de cura.

```
9  func Enter():
10     get_node("../../AnimatedSprite2D").play("walking")
11     player = get_tree().get_first_node_in_group(("Player"))
12
13 func PhysicsUpdate(delta:float):
14     var direction = player.global_position - npc.global_position
15
16     if direction.length() > 15:
17         npc.velocity = direction.normalized() * move_speed
18     else:
19         npc.velocity = Vector2()
20
21     if direction.length() > 100:
22         Transitioned.emit(self, "NPCIdle")
23     if direction.length() < 20:
24         Transitioned.emit(self, "NPCAttack")
25     if get_node("../..").health < 30:
26         Transitioned.emit(self, "NPCSearchHeal")
```

Máquina de Estados Finitos

- O estado Attack apenas executa a animação de ataque e retorna para Idle quando a animação finaliza.

```
5  func Enter():
6      get_node("../../AnimatedSprite2D").play("attack")
7      pass
8
9  func PhysicsUpdate(delta:float):
10     pass
11
12 func _on_animated_sprite_2d_animation_finished():
13     Transitioned.emit(self, "NPCIdle")
14     pass # Replace with function body.
```

Máquina de Estados Finitos

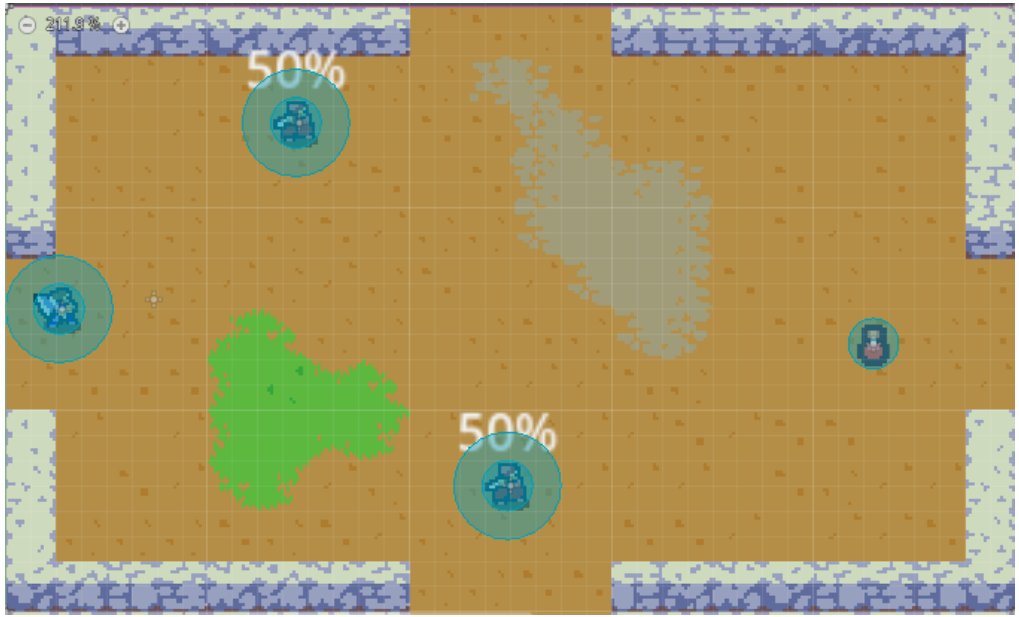
- O estado SearchHeal faz o NPC procurar itens de cura até sua saúde ficar acima de 30.

```gdscript
func PhysicsUpdate(delta:float):
    if is_instance_valid(heal_item):
        var direction = heal_item.global_position - npc.global_position
        if direction.length() > 15:
            npc.velocity = direction.normalized() * move_speed
        else:
            npc.velocity = Vector2()
    else:
        Transitioned.emit(self, "NPCIdle")
        pass

    if $"../..".health > 30:
        Transitioned.emit(self, "NPCIdle")
        pass
```

Outras implementações

- A malha 9x9 não foi implementada até o momento.
- O jogo possui então uma fase com vários inimigos e itens de cura espalhados. O jogador pode pressionar a tecla espaço quando estiver perto dos inimigos para causar dano a eles. Cada ataque causa 25 de dano e inimigos possuem 100 de sáude ao serem instanciados.



Link para o Github

- https://github.com/ricalex42/Projeto-DIM0126---2023.2