PlayerBot

Raissa e Igor

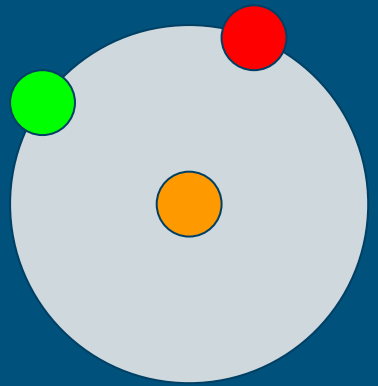
I.A. 2021/02

Objetivos

- Movimentação automata
- Ir até comida mais próxima (Seek)
- Fugir do inimigo (Flee)
- Implementar máquina de estados

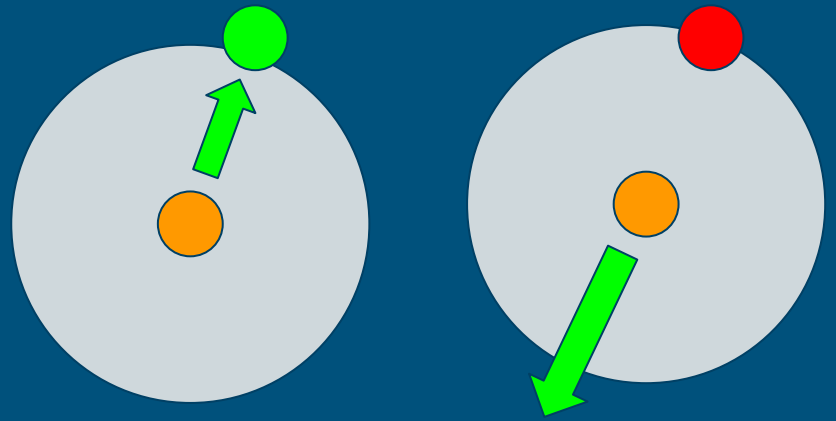
Seek and Flee

- Área circular ao redor da cabeça, detectando colisão com objetos
- Se objeto for comida
  - adiciona-o numa lista
  - verifica qual está mais próximo
  - vai até mais próximo
- Se inimigo
  - cancela-se a busca por comida e esvazia-se a lista
  - foge na direção oposta à do inimigo
  - volta a buscar comida após um delay



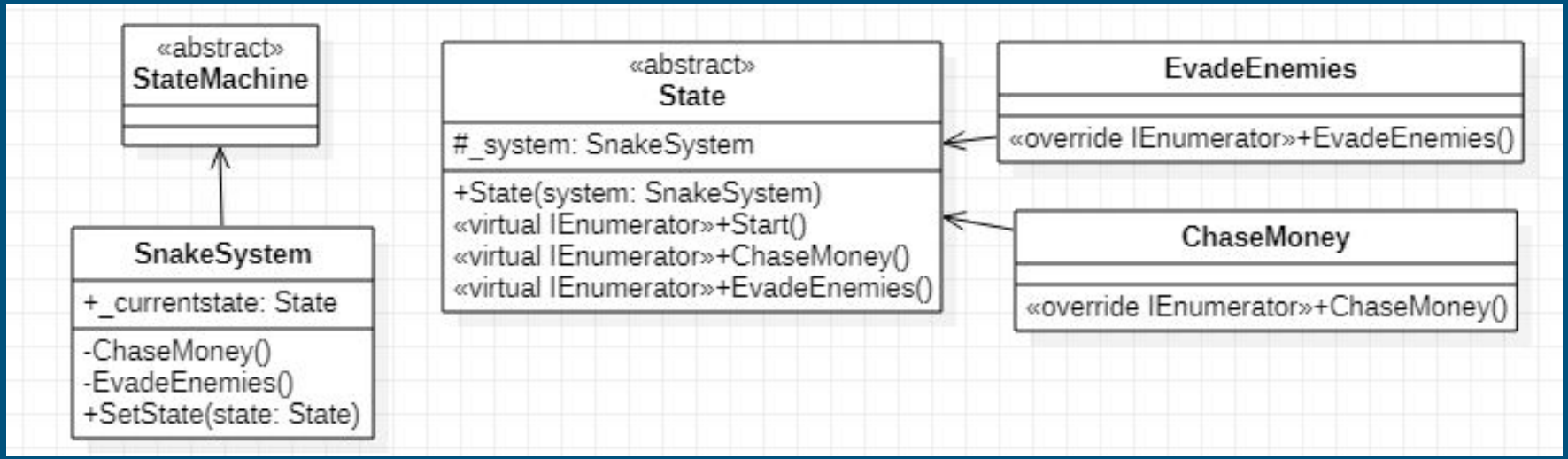
Movimentação autônoma

- Inicialmente usando apenas x,y
- Implementada movimentação do bot
- Ir até a posição da comida
- Com base na posição do bot, anda na direção oposta



Máquina de estados

Inicialmente um conjunto de scripts



Métodos

- Movimentação simples usando x,y
- mudou-se para método usado pelos Dummies para se mover

```
Unity Message | 0 references
void Update()
{
    if(ativo == true && findF == false && flee == false)
    {
        if (choose == 0)
        {
            transform.position = transform.position + (transform.right * -1 * speed * Time.deltaTime);
        }
        else if (choose == 1)
        {
            transform.position = transform.position + (transform.right * speed * Time.deltaTime);
        }
        else if (choose == 2)
        {
            transform.position = transform.position + (transform.up * speed * Time.deltaTime);
        }
        else if (choose == 3)
        {
            transform.position = transform.position + (transform.up * -1 * speed * Time.deltaTime);
        }

    }
```

Métodos

- Verificar comida mais próxima

```
void MoveFood()
{
    float currentDist = 5000;
    int currentIndex = 0;

    overlap();

    if (orbs.Count > 0)
    {
        for (int i = 0; i < orbs.Count; i++)
        {
            if (orbs[i] == null)
            {
                orbs.RemoveAt(i);
            }
            else
            {
                float d = Vector2.Distance(orbs[i].transform.position, owner.transform.position);
                if (d < currentDist)
                {
                    currentDist = d;
                    currentIndex = i;
                }
            }
        }
        owner.transform.position = Vector2.MoveTowards(owner.transform.position,
            orbs[currentIndex].transform.position, ownerMovement.speed * Time.deltaTime);
    }
}
```

Métodos

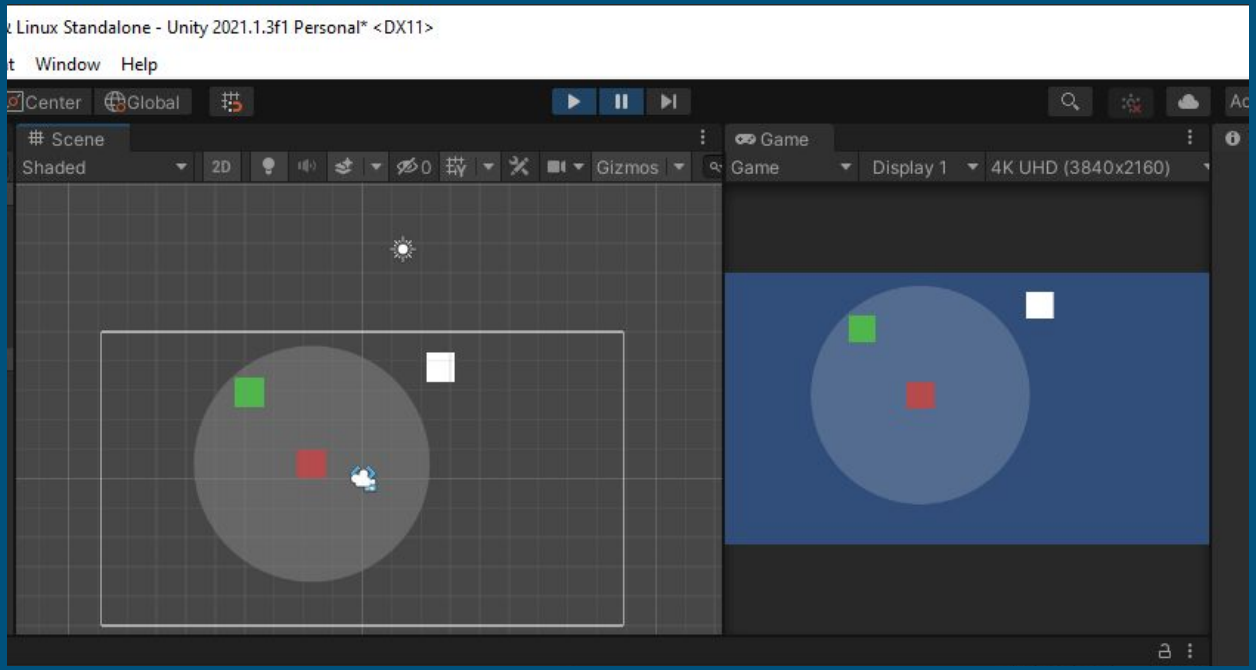
```
void MoveFlee()
{
    remove(bots);
    remove(bodys);
    if (pos.x < owner.transform.position.x)
        owner.transform.position = owner.transform.position + (owner.transform.right * -1 *
                                                               ownerMovement.speed * Time.deltaTime);
    if (pos.x > owner.transform.position.x)
        owner.transform.position = owner.transform.position + (owner.transform.right *
                                                               ownerMovement.speed * Time.deltaTime);
    if (pos.y < owner.transform.position.y)
        owner.transform.position = owner.transform.position + (owner.transform.up *
                                                               ownerMovement.speed * Time.deltaTime);
    if (pos.y > owner.transform.position.y)
        owner.transform.position = owner.transform.position + (owner.transform.up * -1 *
                                                               ownerMovement.speed * Time.deltaTime);
}
```

Métodos

```
1 reference
void CreateChild()
{
    GameObject loadedObject = Resources.Load<GameObject>("Circle");
    GameObject myNewSmoke = Instantiate(loadedObject, this.transform.position,
        Quaternion.identity, this.gameObject.transform);
    myNewSmoke.name = "area-1";
    myNewSmoke.transform.localScale = new Vector3(8.094784f, 8.094784f, 8.094784f);
}
```

```
Unity Message | 0 references
void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Orb")
    {
        //this.collider2D.attachedRigidbody.SendMessage("OnCollisionEnter2D", collision);
        this.transform.parent.GetComponent<move>().GetRoute(collision.gameObject);
    }
    else if (collision.gameObject.tag == "Bot")
    {
        //this.collider2D.attachedRigidbody.SendMessage("OnCollisionEnter2D", collision);
        if(this.transform.parent.transform.position != collision.gameObject.transform.position)
        {
            this.transform.parent.GetComponent<move>().Flee(collision.gameObject);
            Debug.Log(collision.gameObject.name);
        }
    }
}
```

Protótipo


Linux Standalone - Unity 2021.1.3f1 Personal* <DX11>
Window
Help
Center
Global
Scene
Shaded
2D
Gizmos
Game
Game
Display 1
4K UHD (3840x2160)

Problemas

- Área circular dando erro
  - Protegendo player
  - Não detectar comida
- Implementação de raycast dando erro
  - Detectando outros objetos
  - Não detectando comida e bots
- Eixo z dos objetos na cena
- Problemas em implementar máquina de estados

Soluções

- Implementação de switch case
  - int para cada estado
  - Cada valor aciona um método diferente

```
2 references
public override void Execute()
{
    //MoveForward();
    //Debug.Log("loop");
    Debug.Log("state " + state);

    switch (state)
    {
        case 0:
            //MoveRandom();
            MoveForward();
            break;
        case 1:
            MoveFood();
            break;
        case 2:
            MoveFlee();
            break;
    }
}
```

Soluções

- Implementação OverlapCircleAll
  - Retorna array de colliders
  - Separam-se colliders por tag em listas separadas
  - Tratamento e mudança de estados com base na quantidade de elementos na lista

Soluções

```
public void overlap()
{
    Collider2D[] hitC = Physics2D.OverlapCircleAll(owner.transform.position, radius);
    //OnDrawGizmosSelected();

    remove(orbs);
    remove(bots);
    remove(bodys);

    foreach (var hit in hitC)
    {
        if (hit.CompareTag("Orb"))
        {
            if (orbs.Contains(hit.gameObject) == false)
            {
                orbs.Add(hit.gameObject);
                //Debug.Log("entrou orb, " + orbs.Count);
            }
        }
        if (hit.CompareTag("Bot") && hit.gameObject != owner.gameObject)
        {
            bots.Add(hit.gameObject);
            //Debug.Log("entrou bot, " + hit.gameObject.name
            //    + ", " + hit.gameObject.transform.position + ", " + hit.transform.parent.name);
        }
        if (hit.CompareTag("Body"))
        {
            {...
            }

            if (hit.transform.parent.name != player.name)
            {
                bodys.Add(hit.gameObject);
                //Debug.Log("entrou body, " + hit.gameObject.name
                //    + ", " + hit.gameObject.transform.position + ", " + hit.transform.parent.name);
            }
        }
```

Soluções

```
    if (orbs.Count > 0)
    {
        state = 1;
    }
    if (bodys.Count > 0 || bots.Count > 0)
    {
        state = 2;
        pos = new Vector2(bots[0].transform.position.x, bots[0].transform.position.y);
        orbs.Clear();
        ownerMovement.StartCoroutine(RunningAway(delayFuga));
    }
    if (orbs.Count <= 0 && (bodys.Count <= 0 || bots.Count <= 0))
    {
        state = 0;
    }
}
```

Resultados

- IgorGSoares/Prot-tipo-IA (github.com)

- RaissaSche/AISnakeBot2020: Um framework didático para o desenvolvimento de agentes (bots) em um jogo "fortemente" inspirado no Slither.io. #madewithunity (github.com) (pasta “comida e fuga”)

Referências

- https://www.youtube.com/watch?v=5PTd0WdKB-4

- https://www.youtube.com/watch?v=G1bd75R10m4