mentoring

Freie Universität Berlin

Spieleentwicklung

Diane Hanke, Okan Danyeli & Toni Wirth

Programm

1. Woche
2. Woche
3. Woche
4. **Woche**

4. Woche

Hilfe, wir werden verfolgt!

Unser Gegner lernt zu denken!

- Implementierung der gegnerischen Spielerbewegung
- Implementierung der gegnerischen KI

Pick_Up Objekte einsammeln

Pick_Up Objekte einsammeln

- Wird im Playerskript implementiert

```
1  public class PlayerBehavoir : MonoBehaviour {
2
3      private Rigidbody rb;
4      public float speed = 8f;
5      public float rotatespeed = 145f;
6      private Vector3 movement;
7      private int count;
8
9
10      // Use this for initialization
11    void Start () {
12
13          rb = GetComponent<Rigidbody>();
14          count = 0;
15
16    }
17
```

Pick_Up Objekte einsammeln

- Wird im Playerskript implementiert
- onTriggerEnter(Collider col) wie Update() per Frame aufgerufen

```
19  void FixedUpdate () {
20          move();
21  }
22
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
```

Pick_Up Objekte einsammeln

- Wird im Playerskript implementiert
- **onTriggerEnter(Collider col)** wie **Update()** per Frame aufgerufen

```
19  void FixedUpdate () {
20          move();
21  }
22
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
```

Pick_Up Objekte einsammeln

- Wird im Playerskript implementiert

- **onTriggerEnter(Collider col)** wie **Update()** per Frame aufgerufen
- **col.gameObject.CompareTag(“Pick_Up“)** überprüft ob das Objekt berührt wurde

```
19  void FixedUpdate () {
20          move();
21  }
22
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
```

Pick_Up Objekte einsammeln

- Wird im Playerskript implementiert

- **onTriggerEnter(Collider col)** wie **Update()** per Frame aufgerufen
- **col.gameObject.CompareTag ("Pick_Up")** überprüft ob das Objekt berührt wurde

- col.gameObject.SetActive (false) setzt das Objekt was wir berührt haben auf inaktiv

```
19  void FixedUpdate () {
20          move();
21  }
22
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
```

Pick_Up Objekte einsammeln

- Erhöhen unseren Counter **count** um 1.

```
19  void FixedUpdate () {
20          move();
21  }
22
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
```

Counter anzeigen

Counter anzeigen

- **OnGui()** wird zu jeder Frame aufgerufen

```
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
32      private void OnGUI()
33      {
34          //erzeugt am oberen linken rand eine countanzeige
35          GUI.Label(new Rect(1, 1, 200, 200), "Counter: " + count.ToString());
36      }
37  }
```

Counter anzeigen

- **OnGui()** wird zu jeder Frame aufgerufen
- Erzeugt ein Label in der oberen linken Ecke

```
23      void OnTriggerEnter(Collider col)
24      {
25          if (col.gameObject.CompareTag("Pick_Up"))
26          {
27              col.gameObject.SetActive(false);
28              count = count + 1;
29          }
30      }
31
32      private void OnGUI()
33      {
34          //erzeugt am oberen linken rand eine countanzeige
35          GUI.Label(new Rect(1, 1, 200, 200), "Counter: " + count.ToString());
36      }
37  }
```

Nach der Pause folgt die Implementierung der gegnerischen Spielerbewegung.

Implementierung der gegnerischen Spielerbewegung

Variablensetzung

- **Rigidbody rb** dient zum Bewegen des enemy-Objektes.

```
6
7       private Rigidbody rb;
8
9
10
11
12
13
14
15
16
17
18
```

Variablensetzung

- **Rigidbody rb** dient zum Bewegen des enemy-Objektes.
- Über **target** holen wir uns die Position unseres Spielers.

```
6
7       private Rigidbody rb;
8       public Transform target;
9
10
11
12
13
14
15
16
17
18
```

Variablensetzung

- **Rigidbody rb** dient zum Bewegen des enemy-Objektes.
- Über **target** holen wir uns die Position unseres Spielers.
- Vector3 **enemyPosition** und **agentPosition** beinhalten beide Positionen.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10
11
12
13
14
15
16
17
18
```

Variablensetzung

- **Rigidbody rb** dient zum Bewegen des enemy-Objektes.
- Über **target** holen wir uns die Position unseres Spielers.
- Vector3 **enemyPosition** und **agentPosition** beinhalten beide Positionen.
- **UnityEngine.AI.NavMeshAgent** ist unsere "künstliche Intelligenz" zur Verfolgung.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12
13
14
15
16
17
18
```

Variablensetzung II

- Die Variablen **time** und **savedTime** werden dazu genutzt, den Skin für n Sekunden zu deaktivieren.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13
14
15
16
17
18
```

Variablensetzung II

- Die Variablen **time** und **savedTime** werden dazu genutzt, den Skin für n Sekunden zu deaktivieren.
- **distance** ist eine Variable vom Typ float, mit der wir die Entfernung zwischen Spieler und enemy speichern.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13      private float distance;
14
15
16
17
18
```

Variablensetzung II

- Die Variablen **time** und **savedTime** werden dazu genutzt, den Skin für n Sekunden zu deaktivieren.
- **distance** ist eine Variable vom Typ float, mit der wir die Entfernung zwischen Spieler und enemy speichern.
- Die Variable **enemy** beinhaltet alle Eigenschaften unseres GameObjects.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13      private float distance;
14
15      private GameObject enemy;
16
17
18
```

Variablensetzung III

- Die Variable **sound** speichert unser Audiofile, welches wir im Unity-Interface übergeben.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13      private float distance;
14
15      private GameObject enemy;
16      public GameObject sound;
17
18
```

Variablensetzung III

- Die Variable **sound** speichert unser Audiofile, welches wir im Unity-Interface übergeben.
- **public Renderer rend** ist eine Variable vom Typ Renderer, mit der wir die Skins vom enemy deaktivieren können.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13      private float distance;
14
15      private GameObject enemy;
16      public GameObject sound;
17      public Renderer rend;
18
```

Variablensetzung III

- Die Variable **sound** speichert unser Audiofile, welches wir im Unity-Interface übergeben.
- **public Renderer rend** ist eine Variable vom Typ Renderer, mit der wir die Skins vom enemy deaktivieren können.
- Mit Hilfe von **time** und **rend** erzeugen wir einen Teleport für unser enemy.

```
6
7       private Rigidbody rb;
8       public Transform target;
9       private Vector3 enemyPosition, agentPosition;
10      private UnityEngine.AI.NavMeshAgent agent;
11
12      private float time, savedTime;
13      private float distance;
14
15      private GameObject enemy;
16      public GameObject sound;
17      public Renderer rend;
18
```

Startfunktion initialisieren

- Wir holen uns das Rigidbody unseres enemys und speichern dies in **rb**.

```
21
22      rb = GetComponent<Rigidbody>();
23                                                                    ;
24
25
26
27
```

Startfunktion initialisieren

- Wir holen uns das Rigidbody unseres enemys und speichern dies in **rb**.
- Unser **agent** bekommt alle Komponenten, die für die KI benötigt werden.

```
21
22      rb = GetComponent<Rigidbody>();
23      agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
24
25
26
27
```

Startfunktion initialisieren

- Wir holen uns das Rigidbody unseres enemys und speichern dies in **rb**.
- Unser **agent** bekommt alle Komponenten, die für die KI benötigt werden.
- Wir deaktivieren das Herzklopfen.

```
21
22      rb = GetComponent<Rigidbody>();
23      agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
24
25      sound.SetActive(false);
26
27
```

Startfunktion initialisieren

- Wir holen uns das Rigidbody unseres enemys und speichern dies in **rb**.
- Unser **agent** bekommt alle Komponenten, die für die KI benötigt werden.
- Wir deaktivieren das Herzklopfen.
- **Time.time** gibt die aktuelle Uhrzeit in Sekunden wieder.

```
21
22      rb = GetComponent<Rigidbody>();
23      agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
24
25      sound.SetActive(false);
26      savedTime = Time.time;
27
```

Nach der Pause folgt die Implementierung der gegnerischen KI.

Implementierung der gegnerischen KI

Updatefunktion initalisieren

- **enemyPosition** bekommt die Koordinaten des Gegners.

```
29
30      enemyPosition = transform.position;
31
32
33
34
35
```

Updatefunktion initalisieren

- **enemyPosition** bekommt die Koordinaten des Gegners.

- **agentPosition** bekommt die Koordinaten des Spielers.

```
29
30      enemyPosition = transform.position;
31      agentPosition = target.position;
32
33
34
35
```

Updatefunktion initalisieren

- Die erste if-Anweisung setzt die Zeit ab über 18 Sekunden zurück.

```
32
33      if (Time.time - savedTime >= 18)
34      {
35        savedTime = Time.time;
36      }
37
38
```

Updatefunktion initalisieren

- Die erste if-Anweisung setzt die Zeit ab über 18 Sekunden zurück.
- Die zweite if-Anweisung deaktivert ab über 10 Sek. die Skin und geht Richtung Spieler.

```
37
38      if (Time.time - savedTime >= 10)
39      {
40        rend.enabled = false;
41        agent.SetDestination(target.position);
42      }
43
```

Updatefunktion initalisieren

- Die erste if-Anweisung setzt die Zeit ab über 18 Sekunden zurück.
- Die zweite if-Anweisung deaktivert ab über 10 Sek. die Skin und geht Richtung Spieler.
- Die else-Anweisung setzt die Skin wieder auf *aktiv*.

```
44      else
45      {
46        if (rend.enabled == false)
47        {
48          rend.enabled = true;
49        }
50      }
```

Updatefunktion initalisieren

- **distance** ermittelt mit Hilfe der Funktion Distance() die Entfernung zwischen **agentPosition** und **enemyPosition**

```
53      distance = Vector3.Distance(agentPosition, enemyPosition);
54
55
56
57
58
59
```

Updatefunktion initalisieren

- **distance** ermittelt mit Hilfe der Funktion Distance() die Entfernung zwischen **agentPosition** und **enemyPosition**

- Ab einer Entfernung von unter 5 Koordinateneinheiten wird der Sound aktiviert .

```
53      distance = Vector3.Distance(agentPosition, enemyPosition);
54
55      if (distance < 5f)
56      {
57        sound.SetActive(true);
58      }
59
```

Updatefunktion initalisieren

- **distance** ermittelt mit Hilfe der Funktion Distance() die Entfernung zwischen **agentPosition** und **enemyPosition**

- Ab einer Entfernung von unter 5 Koordinateneinheiten wird der Sound aktiviert .

- Alles über 5 Koordinateneinheiten deaktiviert den Sound.

```
59
60      else
61      {
62        sound.SetActive(false);
63      }
64
65
```

We finished it!