HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

Entwicklung eines 2D Plattformers

Dokumentationn

Fabio Gimmillaro, Daniel Schreiber, Tobias Rühl und Joscha Wülk

Medienprojekt

Betreuer: Prof. Dr. Christof Rezk-Salama

Trier, Abgabedatum

Inhaltsverzeichnis

1

Vorwort

2

Konzeptionierung

3

Die Kamera

4

GameObject “Player“ & Komponenten

5

Enemies

In unserem Spiel waren zwei sich ähnliche Gegner angedacht, wovon einer als Fernkämpfer und der andere als Nahkämpfer fungierte. Auf Grund mangelnder Animationen hat lediglich der Fernkämpfer seinen Weg ins Spiel gefunden. Von der Implementierung hätten sich die beiden Gegnertypen aber kaum unterschieden, da die KI möglichst modular und austauschbar gehalten werden sollte. Somit unterscheiden sich die zwei Typen lediglich durch eine andere Reichweite für den Angriff, und eine andere Angriffsfunktion, die aufgerufen wird.

Die Gegner wurde eine simple Entscheidungskaskade in Form von verschiedenen If-Anweisungen gegeben, nach denen die Gegner dann ihr Verhalten auswählen. Das Verhalten der Gegner ist stark davon abhängig ob Sichtkontakt zum Spieler besteht, was durch Raycasts zwischen dem Gegner und dem Spieler überprüft wird. Hierbei kann auch ein Blickwinkel definiert werden, in welchem der Spieler wahrgenommen wird.

```
1  [...]
2  //Blickrichtung entweder x=1 oder -1
3  if (actions.facingRight)
4      viewVector += new Vector2(1, 0);
5  else
6      viewVector += new Vector2(-1, 0);
7
8  //Winkel zwischen Blickrichtung und Verbindungsvektor
9  float currentAngle = Vector2.Angle(viewVector, difference);
10 RaycastHit2D hit = Physics2D.Raycast(visionCheck.position,viewVector,
       noticeDistance);
11
12 //Ueberpruefen ob der Winkel maximal dem angegebenen Field of View Winkel
       ist
13 if (currentAngle <= fovAngle)
14 {
15     //Raycast der von Augen zum Ziel geschossen wird, wenn das erste
           getroffene Objekt der Spieler ist, ist die sicht nicht blockiert
16     if (hit.collider != null)
17         playerVisible = hit.collider.gameObject == rigplayer.gameObject;
18    else
19       playerVisible = false;
20 }
21 else
22    playerVisible = false;
```

Mit Hilfe dieses Sichtbarkeitschecks, kann die KI verschiedene Aktionen abhängig von verschiedenen Bedingungen wählen. Diese Entscheidungskaskade könnte man als Behaviour Tree folgendermaßen visualisieren:

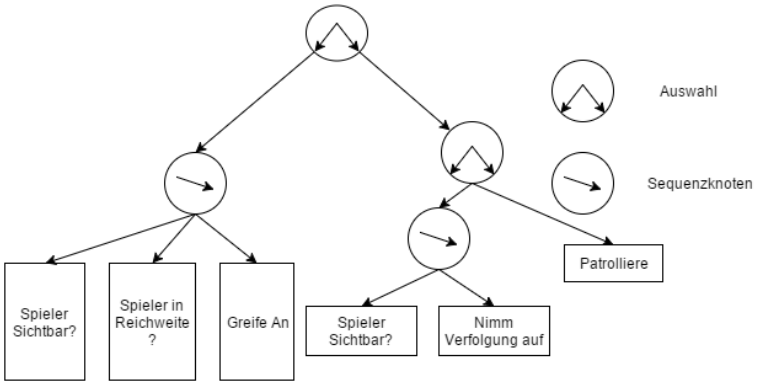

Abb. 5.1. Verhalten des Gegners als Behaviour Tree

Implementiert wurde der Entscheidungsalgorithmus jedoch nur mit if-Bedingungen und nicht mit einem Behaviour Tree oder einer State Machine.

```
1  //Wenn in Angriffsreichweite, und der Spieler sichtbar ist, bleibe stehen
       beginne anzugreifen
2  //Ruft die Angriffsmethode des RangedSystems auf
3  if ((inAttackRangex && inAttackRangey) && vision.playerVisible)
4  {
5     movement.move(0.0f);
6     anim.SetBool("AttackInProgress", true);
7     StartCoroutine(rangeSys.shoot(true));
8  }
9
10 //Wenn sich Spieler nicht in Reichweite zum Angreifen befindet, aber
       Sichtkontakt besteht, nimm Verfolgung auf
11 //Ruft die Bewegungsmethode des Movementssystem auf.
12 if (!inAttackRangex && inAttackRangey && vision.playerVisible)
13 {
14    if (movement.grounded)
15    {
16       if (walkingRight)
17          movement.move(1.0f);
18       else
19          movement.move(-1.0f);
20    }
21    else
22       movement.move(0.0f);
23 }
24
25
26 //Wenn das Ziel nicht sichtbar ist, soll patrolliert werden
27 if (!vision.playerVisible)
28 {
29    //Wenn wir auf dem Boden befinden, beweg dich, ansonsten bleib stehen
30    if (movement.grounded)
31    {
32       //Befinden wir uns vor einer Wand oder einem Abgrund, drehe um.
33       if (hittingWall || onAnEdge)
34          walkingRight = !walkingRight;
35       if (walkingRight)
36       {
37          movement.move(1.0f);
```

```
38            }
39            else
40            {
41                movement.move(-1.0f);
42            }
43        }
44        else
45            movement.move(0.0f);
46    }
```

6

Endboss

Der Kampf mit dem Endboss, welcher auf der obersten Plattform des Levels zu finden ist, wurde speziell für den Bosskampf gescriptet. Hierzu wurden ähnliche Logikabfragen wie bei der normalen Gegner KI implementiert, jedoch in einer spezielleren Form, da sich durch gewünschte Bossfähigkeiten diese nicht mit dem normalen KI Script umsetzen lassen.

Der Boss hat 3 Angriffe die von bestimmte Bedingungen haben:

Schlag mit linker Hand Wenn sich der Spieler in der Trefferzone der linken Hand befindet

Schlag mit rechter Hand Wenn sich der Spieler in der Trefferzone der rechten befindet

Schreien Keine Bedingung. Lässt den Spieler für eine gewisse Zeit bewegungsunfähig werden.

Damit diese drei Angriffe nicht etwa in statischer Reihenfolge ausgeführt werden, und etwas mehr Dynamik in den Kampfablauf bringen, wurde hier ein Zufallsalgorithmus implementiert. Dieser entscheidet zufällig welcher der möglichen Angriffe gewählt werden soll, unter Berücksichtigung der letzten ausgeführten Aktionen. Eine der möglichen Aktionen wird proportional zu ihren vergangenen Aktionsphasen, seit der letzten Verwendung immer wachsen. Beim ersten Angriff besteht also für jede Funktion eine Chance von $\frac{1}{3}$. Wird dann beispielsweise der Schlag mit der linken Hand ausgeführt, ist die Chance für nochmals diese Aktion $\frac{1}{5}$, für den Schrei und den Schlag mit der rechten Hand jeweils $\frac{2}{5}$.

Besonderheit hierbei ist, dass die Schläge zweimal hintereinander ausgeführt werden, der Zähler wird lediglich auf 1 zurückgesetzt, während es ausgeschlossen ist, dass zweimal ein Schrei ausgeführt wird.

7

Grafiken und Animationen

8

Sounds

9

Spielobjekte

10

Events

11

Demolevel

12

Menüführung

13

Head-Up-Display & Tooltips

14

Zielsetzung

15

Abschlussanalyse