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:

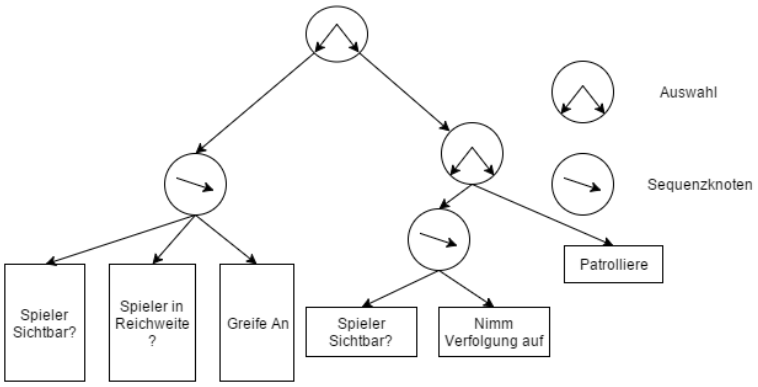

Abb. 5.1. Verhalten des Gegners als Behaviour Tree

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

6

Endboss

7

Grafiken und Animationen

8

Sounds

9

Spielobjekte

10

Events

11

Demolevel

12

Menüführung

13

Head-Up-Display & Tooltips

14

Zielsetzung

15

Abschlussanalyse