#17.2 Simple KI

Um unsere KI jetzt in SFML zu programmieren brauchen wir natürlich eine Entity. Dazu erstelle man eine neue Entity oder nimmt eine bereits erstellte.

class Entity

{

// Graphic Variable sollte in jeder Entity drin sein

// RectangleShape,CircleShape oder Sprite

public RectangleShape graphic;

Screen screen;

// Konstruktor

public Entity(Screen parentScreen)

{

screen = parentScreen;

graphic = new RectangleShape();

graphic.Size = new Vector2f(50, 50);

graphic.FillColor = Color.Blue;

graphic.Position = new Vector2f(300, 300);

}

// Loop Methode

public void loop()

{

}

public void draw()

{

screen.draw(this.graphic);

}

}

Jetzt müssen wir dieser Entity eine **state** Variable geben. Die State Variable **speichert den Zustand** unseres Zustandsautomaten.

// Graphic Variable sollte in jeder Entity drin sein

// RectangleShape,CircleShape oder Sprite

public RectangleShape graphic;

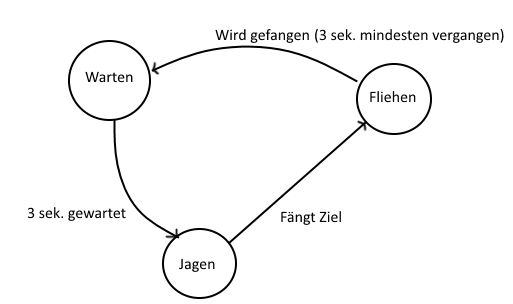
string state;

Screen screen;

Den state kann man als int, enum, string oder auch vielen anderen Typen speichern. Hier nehmen wir einen String.

Als nächstes machen wir uns Gedanken, wie unsere Zustände sind. Für dieses Beispiel nehmen wir folgenden Zustände:

Warten, Fliehen, Jagen

Der Sinn dieser Zustände ist: Das Objekt spielt "fangen". Es läuft weg bis man es berührt("fängt"), dann wartet es eine Zeit damit man selbst weglaufen kann und dann verfolgt es einen bis man es fängt. Danach läuft es wieder weg(wobei es ein paar Sekunden "Unfangbar" bleibt.) Der Zugehörige Zustandsautomat sieht wie folgt aus:

Der Startzustand ist hierbei das "Fliehen". Dazu setzen wir unsere State Variable entsprechend im Konstruktor auf "Fliehen". Außerdem speichern wir ein Transformable in der Klasse ab, die das Ziel darstellt.

public RectangleShape graphic;

Screen screen;

string state;

Transformable target;

// Konstruktor

public Entity(Screen parentScreen, Transformable target)

{

screen = parentScreen;

graphic = new RectangleShape();

graphic.Size = new Vector2f(50, 50);

graphic.FillColor = Color.Blue;

graphic.Position = new Vector2f(300, 300);

this.target = target;

state = "FLIEHEN";

}

Jetzt wollen wir für jeden Zustand eine Methode erstellen. Also drei Methoden: fliehen, jagen und warten.

public void fliehen()

{

}

public void jagen()

{

}

public void warten()

{

}

Diese Methoden beinhalten das Verhalten in den jeweiligen Zuständen.

In der Loop überprüfen wir den Zustand und rufen immer die entsprechende Methode auf:

// Loop Methode

public void loop()

{

if (state == "FLIEHEN")

{

fliehen();

}

else if (state == "JAGEN")

{

jagen();

}

else if (state == "WARTEN")

{

warten();

}

}

In den Methoden müssen wir jetzt die Übergangsregeln definieren. Dafür brauchen wir noch zusätzlich eine Counter-Variable:

public RectangleShape graphic;

string state;

Screen screen;

Transformable target;

int counter;

Für Fliehen zählen wir den Counter jetzt hoch und sobald er über 180 ist(180 Frames = 3 sek.) überprüfen wir ob das target zu Nahe ist(hier kann man auch eine Kollisionsabfrage machen, wenn man als Target ein RectangleShape definiert), und wenn das der Fall ist, wechseln wir zu Warten und setzen den Counter zurück.

public void fliehen()

{

if (counter >= 180)

{

// Ist die distanz kleiner als 50 pixel?

if (utility.Functions.distance(this.graphic, target) < 50)

{

state = "WARTEN";

counter = 0;

}

}

else

{

counter++;

}

}

In jagen überprüfen wir auch die Distanz und wechseln zu fliehen, sobald wir nahe genug sind.

public void jagen()

{

// Ist die distanz kleiner als 50 pixel?

if (utility.Functions.distance(this.graphic, target) < 50)

{

state = "FLIEHEN";

}

}

In Warten lassen wir einfach den Counter bis 180 zählen.

public void warten()

{

if (counter >= 180)

{

state = "JAGEN";

counter = 0;

}

else

{

counter++;

}

}

Jetzt haben wir die Übergangsregelungen und die States eingebaut. Jetzt fehlt noch das Verhalten.

Dazu lassen wir das Objekt in fliehen immer vom Target wegbewegen(aber nie über die Bildschirmgrenzen hinaus).

public void fliehen()

{

//Der Einheitsvektor in Flieh-Richtung

Vector2f dir = utility.Functions.getUnitVector(this.graphic.Position - target.Position);

this.graphic.Position += dir \* 2; //Mit Geschwindigkeit 2 weg vom Target

//Gehe nie über Bildschirm hinaus

if (this.graphic.Position.X > 800)

{

this.graphic.Position = new Vector2f(800, this.graphic.Position.Y);

}

if (this.graphic.Position.X < 0)

{

this.graphic.Position = new Vector2f(0, this.graphic.Position.Y);

}

if (this.graphic.Position.Y > 600)

{

this.graphic.Position = new Vector2f(this.graphic.Position.X,600);

}

if (this.graphic.Position.Y < 0)

{

this.graphic.Position = new Vector2f(this.graphic.Position.X,0);

}

if (counter >= 180)

{

// Ist die distanz kleiner als 50 pixel?

if (utility.Functions.distance(this.graphic, target) < 50)

{

state = "WARTEN";

counter = 0;

}

}

else

{

counter++;

}

}

Für das Jagen machen wir dasselbe, nur dass wir uns auf das Target zubewegen(ohne bildschirmgrenzen)

public void jagen()

{

//Der Einheitsvektor in Jag-Richtung

Vector2f dir = utility.Functions.getUnitVector(target.Position - this.graphic.Position);

this.graphic.Position += dir \* 2; //Mit Geschwindigkeit 2 das target jagen

// Ist die distanz kleiner als 50 pixel?

if (utility.Functions.distance(this.graphic, target) < 50)

{

state = "FLIEHEN";

}

}

Die Warten Funktion kann man so lassen.

Jetzt die Entity wie immer in Mainscreen hinzufügen und als zweiten Parameter den Player angeben.

class MainScreen : Screen

{

// Deklariere hier Objekte oder Klassenvariablen!

Entity entity;

RectangleShape player;

// Setup, wird immer einmal zu Beginn eines Screens aufgerufen

// Hier Startwerte setzen!

public override void setup()

{

player = new RectangleShape();

player.Size = new Vector2f(50, 50);

player.Position = new Vector2f(400, 400);

player.FillColor = Color.Green;

player.Origin = player.Size \* 0.5f;

entity = new Entity(this, player);

}

// Loop, wird jeden Frame (60 mal die Sekunde) aufgerufen

public override void loop()

{

if (Keyboard.IsKeyPressed(Keyboard.Key.W))

{

player.Position += new Vector2f(0, -3);

}

if (Keyboard.IsKeyPressed(Keyboard.Key.A))

{

player.Position += new Vector2f(-3, 0);

}

if (Keyboard.IsKeyPressed(Keyboard.Key.S))

{

player.Position += new Vector2f(0, 3);

}

if (Keyboard.IsKeyPressed(Keyboard.Key.D))

{

player.Position += new Vector2f(3, 0);

}

draw(player);

entity.loop();

entity.draw();

}

}