

xna

Bestrahlung

Lesson 4: Feuern



Spieleentwicklung mit dem XNA Framework

M! und Games Academy machen ein Spiel – Teil 4

Willkommen zur vierten Folge des XNA-Programmierkurses. Diesmal bringen wir dem Raumschiff das Schießen bei und wagen uns an die Gegner.

Schussrichtung

Diese wollen wir mithilfe des rechten Analogsticks bestimmen. Dazu verändern wir die letztes Mal eingefügte Variable *fPlayerRotation*, die bestimmt, in welche Richtung unsere Grafik dreht. Wir befinden uns immer noch in der 'Update Methode', dort fügen wir die Zeilen aus **Codekasten 1** ein: Sobald wir den rechten Stick bewegen und die Achsen X und Y nicht mehr null sind, errechnen wir mithilfe der Umkehrfunktion des Tangents den Winkel, mit dem wir unsere Player-Grafik rotieren.

Schießen

Wenn wir den rechten Analogstick bewe-

gen, möchten wir ein Geschoss abfeuern, um die herannahenden Feinde zu eliminieren. Wir werden natürlich mehr als ein Projektil abschießen. Deshalb erstellen wir uns dafür ein Objekt, das sie repräsentiert. Das passiert mit den Zeilen aus **Codekasten 2**, und zwar oberhalb der Zeile *public class Game1 : Microsoft.Xna.Framework.Game*.

Wie **Codekasten 2** zu sehen, haben wir in unserer Objekt-Definition folgende Werte: die Grafik; die Startposition des abgefeuerten Geschosses; die Geschwindigkeit; die Rotation des Geschosses abhängig von der Drehung des Raumschiffes; eine Alive-Variable, mit der wir festlegen, ob das Geschoss noch aktiv ist oder schon zerstört wurde beziehungsweise gar außerhalb des Bildschirmes ist. Anschließend definieren wir eine Liste, in welche wir jeden neu generierten Schuss hineinladen. Dafür notieren wir unterhalb der Zeile *float fPlayerRotation*;

die Zeile *List<Shot> lShots*; - somit deklarieren wir eine Liste vom Objekt Shots mit dem Namen lShots. Um sie verwenden zu können, müssen wir sie noch aktivieren. In der 'Initialize Methode' schreiben wir dafür *lShots = new List<Shot>()*.

Nun befassen wir uns mit dem Einbau der Schüsse. Da wir diese mit dem rechten Stick abfeuern, schreiben wir den **Codekasten 3** unterhalb an der Stelle, an der wir die Player Rotation berechnen. Als Erstes erstellen wir ein neues Objekt vom Typ 'Schuss'. Danach laden wir die Textur *t2DShot*, die Ihr zuvor selbst mit Euren erworbenen Kenntnissen initialisiert und deklariert habt.

Für den Schuss verwenden wir die 'Rocket Grafik'. In den nächsten zwei Zeilen wird die Position der Rakete zugewiesen. Ihre Rotation basiert auf der Drehung, die das Raumschiff zu diesem Zeitpunkt vollführt.

SO STEIGT IHR MIT EIN:

Wer an unserem Entwickler-Experiment teilnehmen will, sollte diese Adressen und Links verinnerlichen:

Tools: Die notwendigen Programme für die XNA-Entwicklung könnt Ihr Euch auf <http://creators.xna.com> herunterladen.

Stand der Dinge: Die aktuellste Fassung unseres Projektes findet Ihr auf der Homepage www.games-academy.de

Ratgeber: Falls Probleme oder Fragen aufkommen, könnt Ihr mit dem Autor der Artikelreihe unter der folgenden E-Mail-Adresse in Kontakt treten: alfred.bigler@serafan.ch.

Gegner

Für die Feinde legen wir ebenfalls ein eigenes Objekt an. Diesen **Codekasten 4** schreiben wir oberhalb des Geschoss-Objekts.

Wie bei Letzterem sind alle Werte in der Liste identisch, mit der einen Ausnahme: Der Gegner erhält die zusätzliche Variable *life* für die Lebensanzeige.

```
2 public class Shot
{
    public Texture2D t2dShot;
    public Vector2 vStartPosition;
    public Vector2 vVelocity;
    public float fStartRotation;
    public bool bAlive;
}
```

```
4 public class Enemy
{
    public Texture2D t2dEnemy;
    public int iLive;
    public bool bAlive;
    public Vector2 vPosition;
    public Vector2 vVelocity;
    float fAbschlussRotation;
}
```

```
1 if ((GamePad.GetState(PlayerIndex.One).ThumbSticks.Right.X != 0.0f) |
    (GamePad.GetState(PlayerIndex.One).ThumbSticks.Right.Y != 0.0f))
{
    fPlayerRotation =
    (float)Math.Atan2(GamePad.GetState(PlayerIndex.One).ThumbSticks.Right.X,
    GamePad.GetState(PlayerIndex.One).ThumbSticks.Right.Y);
}
```

```
3 Shot sShot = new Shot();
sShot.bAlive = true;
sShot.t2dShot = t2DShot;
sShot.vAnfangsPosition = new Vector2(iPlayerPositionX, iPlayerPositionY);
sShot.fAbschlussRotation = fPlayerRotation;
sShot.vVelocity = new Vector2(
    (float)Math.Cos(fPlayerRotation - 1.5f),
    (float)Math.Sin(fPlayerRotation - 1.5f)) * 10.0f;
lShots.Add(sShot);
```

ALFRED BIGLER



Alfred Bigler ist Student an der Games Academy in Berlin im 3. Semester und gleichzeitig CEO der frisch gegründeten Firma Twisted Bytes.

