PE FS25 Semesterarbeit, Teil 1

Im Folgenden beschreiben wir das Szenario und die Aufgaben für Teil 1 der Semesterarbeit für das Modul PE. In diesem Szenario soll ein Dartpfeil auf eine Dartscheibe zufliegen. Pfeil und Scheibe werden dabei von der Gewichtskraft in Richtung des Bodens beschleunigt, während die Geschwindigkeit des Pfeils in Richtung der Scheibe konstant bleibt.

Wir haben hierfür eine Vorlage vorbereitet, auf deren Grundlage die Simulation in Unity (die Engine, in welcher Sie im Verlauf des Semesters die Semesterarbeit implementieren werden) umgesetzt werden kann. In dieser Vorlage sind für Dartpfeil und -Scheibe jeweils ein Controller hinterlegt, in welchem Sie die entsprechende Physik implementieren müssen. Es handelt sich dabei um die C#-Skripts `MoveDart.cs` sowie `MoveDartBoard.cs`. In der Vorlage ebenfalls vorhanden ist Code, welcher die Position von Pfeil und Scheibe im Verlauf der Zeit festhält. Diese Funktionalität ist in `Controller.cs` implementiert und schreibt in eine Datei `time_series.csv`.

Sie können die Simulation mit der Leertaste («Space») starten, die Szene mit der Taste «r» zurücksetzen und das Schreiben der csv-Datei mit der Taste «o» auslösen. Dabei wird der Inhalt dieser Datei mit den Positionen aus dem aktuellen Lauf überschrieben. Ausserdem ist es möglich, die Kamera mit den Pfeiltasten bzw. mit «w» und «s», «a» und «d» sowie «q» und «e» einzustellen.

1. Vorbereitung

Befolgen Sie zunächst die Anweisungen in Moodle [1] zur Installation und Einführung in Unity.

Beschaffen Sie anschliessend unsere Vorlage, indem Sie den Code aus dem github-Repository [2] als zip-Datei herunterladen und extrahieren, oder indem Sie das Repository klonen. Laden Sie dieses Projekt in Unity, indem Sie in Unity Hub unter «Projects» mit der Schaltfläche «Add» und dann «Add project from disk» das Verzeichnis anwählen, in welchem das Projekt enthalten ist. Ein Doppelklick auf den neuen Eintrag in der Projektliste öffnet die Vorlage dann in Unity. Unter Umständen müssen Sie noch die Szene «Dart» aus den Assets auswählen, dies geschieht womöglich nicht automatisch.

2. OrbitCamera

Das Skript `OrbitCamera` ist derart implementiert, dass Sie einem Target (der Dartscheibe, wie im Inspektor festgelegt) folgt. Erklären Sie diese Implementierung: Zeichnen Sie je ein Koordinatensystem, in welchem man pitch und yaw erkennen kann. Erklären Sie anhand einer weiteren Skizze, wie die neue Position der Kamera zustande kommt, d.h. die dritte Skizze sollte die relevanten Vektoren zwischen Koordinatensystem, Dartscheibe und Kamera zeigen. Hier der relevante Codeabschnitt

```
Quaternion rotation = Quaternion.Euler(pitch, yaw, 0);
Vector3 offset = rotation * new Vector3(0, 0, distance);
transform.position = targetInitialPosition + offset;
```

3. Pfeil und Scheibe fallen zu Boden

Da in der Vorlage noch keine Physik implementiert ist, stehen Pfeil und Scheibe auch nach dem Betätigen der Leertaste noch still im Raum und bewegen sich nicht. Dies möchten wir nun ändern.

Zuerst möchten wir erreichen, dass sowohl die Scheibe wie auch der Pfeil in Richtung Boden fallen. Implementieren Sie dazu in `FixedUpdate()` beider Controller eine Gewichtskraft, welche dies verursacht. Nutzen Sie hierfür `rb.AddForce()` [4]. Definieren Sie dazu den $\vec{g}$-Vektor einmal selbst und benutzen Sie den vordefinierten `Physics.gravity`-Vektor.

Beschreiben Sie im Bericht auch den Unterschied zwischen `ForceMode.Force` und `ForceMode.Acceleration`.

Erklären Sie im Bericht, weshalb die beiden Game Objekte `Dart` und `DartBoard` vor dem Betätigen der Leertaste still stehen. Hinweise: Was ist die Bedeutung von `isKinemtik`? Beachten Sie auch die Einstellung `useGravity` zu den beiden Game Objects.

4. Dartpfeil fliegt in Richtung der Dartscheibe

In einem weiteren Schritt möchten wir erreichen, dass sich der Dart in Richtung der Dartscheibe bewegt. Orientieren Sie sich zunächst in den Weltkoordinaten der Unity-Simulation. Achten Sie dabei darauf, dass Unity ein linkshändiges Koordinatensystem nutzt. In welche Richtungen zeigen dann die x-, y- bzw. z-Achsen? In welche dieser Richtungen soll der Dartpfeil fliegen?

Zu Beginn der Simulation soll der Pfeil mit einer Geschwindigkeit von `initialVelocity` in Richtung der Scheibe fliegen. Setzen Sie dazu in `MoveDart.LaunchDart()` die `rb.linearVelocity`. Tipp: Konsultieren Sie die API-Dokumentation zu `Rigidbody.linearVelocity` [3].

5. Export der Bewegungsdaten

In der bisherigen Implementation von `Controller` werden nur die Positionsdaten des Pfeils und der Dartscheibe exportiert. Erweitern Sie dieses Skript, sodass auch die Geschwindigkeit und die Beschleunigung beider Objekte exportiert werden.

Im Skript `Controller` ist die Datenstruktur `TimeSeriesData` definiert. Fügen Sie dort Felder für die Geschwindigkeit und Beschleunigung hinzu, weisen Sie die entsprechenden Werte in `FixedUpdate()` zu und passen Sie den Export entsprechend an.

6. Bericht

Verfassen Sie einen kurzen Bericht über Ihre Implementation. Achten Sie dabei darauf, dass Sie dabei aufzeigen, wie Sie die einzelnen Schritte implementiert haben und dass Sie alle Fragen und Aufforderungen in dieser Aufgabe nachgekommen sind. Der Bericht soll auch einen Screenshot des

Endzustandes enthalten. Plotten Sie ausserdem anhand der Daten aus `time_series.csv` die Position, Geschwindigkeit und Beschleunigung von Scheibe und Pfeil auf eine geeignete Art und Weise (Python, Excel, gnuplot, …). Gehen Sie im Bericht bei der Diskussion dieser Plots auf die folgenden Punkte ein:

1. Achten Sie darauf, dass beide Achsen dieser Plots korrekt beschriftet sind (was wird auf dieser Achse aufgetragen und welche physikalische Einheit hat diese Grösse?).
2. Wie ändert sich die Position von Pfeil bzw. Scheibe in den beiden dargestellten Raumrichtungen und was ist der Grund hierfür?
3. Vergleichen Sie die Höhe (über dem Boden) von Pfeil und Scheibe im Verlauf der Zeit. Welche Unterschiede oder Gemeinsamkeiten stellen Sie fest? Erklären Sie diese Unterschiede/Gemeinsamkeiten unter Berücksichtigung der Tatsache, dass Pfeil und Scheibe unterschiedliche Massen haben.

[1] https://moodle.zhaw.ch/mod/page/view.php?id=1560824

[2] https://github.zhaw.ch/physicsenginesmodule/Dart-609-template

[3] https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Rigidbody-linearVelocity.html

[4] https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Rigidbody.AddForce.html