Dokumentation: Kollisionsdetektion, WPF-Projekt in SSDS

Ivan Kurilin Dimitrij Pivovar

18. Januar 2025

Technische Hochschule Köln
Prof. Herr Dr. Konen

1 Einführung

1.1 Projektziel

Das Ziel dieses WPF-Projekts war es, eine Kollisionsdetektion in einer 3D-Umgebung zu implementieren. Die Objekte sollten eine realistische Reaktion auf Kollisionen mit den Bällen zeigen und dem Benutzer verschiedene Methoden zur Veranschaulichung bieten, um die Kollisionen näher zu betrachten und besser zu verstehen.

1.2 Überblick

Für die Entwicklung des Projekts stand uns eine einfachere Version in Processing zur Verfügung. In dieser wurden gleichmäßig bewegte Bälle im Raum simuliert, die nur mit den Wänden kollidieren konnten. Auf dieser Grundlage haben wir ein Projekt entwickelt, bei dem nicht nur die Kollisionen sichtbar sind, sondern dem Benutzer auch eine benutzerfreundliche Oberfläche, Funktionen zur Visualisierung und Optimierungsalgorithmen bereitgestellt werden.

2 Benutzeroberfläche

2.1 Design

Dem User wird eine benutzerfreundliche Oberfläche angeboten, die man ganz einfach durch die Menüpunkte navigieren kann.

2.2 Funktionen

- **Normalmodus**:
  Dies ist die Grundlage des Projekts. Hier kann der Benutzer sehen, wie das Programm zu Beginn war.

- **Kollisionsdetektion (Bruteforce)**:
  Eine einfache Überprüfung auf Kollisionen mittels Bruteforce mit einer Laufzeit von $O(n^2)$.

- **Impuls Demonstration**:
  In diesem Modus hat der Benutzer die Möglichkeit, per Mausklick auf die Bälle zu klicken. Das Ziel dieses Modus ist es, zu verdeutlichen, welchen Einfluss die Masse auf die Kollision mit anderen Bällen hat, die weniger Masse besitzen.

- **Tunneling Demonstration**:
  In diesem Modus wird der quantenmechanische Tunneleffekt simuliert. Dies veranschaulicht das Phänomen, bei dem Teilchen eine Barriere durchqueren können, die sie gemäß klassischer Physik nicht passieren könnten.

- **Vektorenrichtung Drawer**:
  Dieser Effekt zeichnet Pfeile, die in die Bewegungsrichtung der Bälle zeigen.

- **Effet**:
  Dieser Modus ermöglicht es, die Rotation des Balls zu betrachten.

- **Quadtree Algorithmus**:
  Der Quadtree Algorithmus verbessert die Performance der Kollisionsdetektion. Durch die hierarchische Unterteilung des Raumes in vier Teilbereiche pro Knoten können Kollisionen effizienter erkannt werden. Die Laufzeit für das Einfügen eines Elements in einen Quadtree beträgt $O(\log(n))$.

3 Physikalische Grundlagen

3.1 Elastische Kollision

Wir bezeichnen einen Stoß dabei als elastisch, wenn die Summe der kinetischen Energien der Stoßpartner nach dem Stoß genau so groß ist wie vor dem Stoß. [1]
Das bedeutet für uns, dass keine kinetische Energie in andere Energieformen wie Wärme oder Verformungsenergie umgewandelt wird. Ein klassisches Beispiel für eine nahezu elastische Kollision ist der Zusammenstoß von Billardkugeln: Nach dem Stoß bewegen sich die Kugeln mit nahezu unveränderter kinetischer Energie weiter.
Im Gegensatz dazu steht die unelastische Kollision, bei der ein Teil der kinetischen Energie in andere Energieformen umgewandelt wird, was oft zu einer dauerhaften Verformung der Objekte führt. Ein typisches Beispiel hierfür ist ein Autounfall, bei dem die kinetische Energie in Verformungsenergie umgewandelt wird, was zu Schäden am Fahrzeug führt.

Um eine Kollision festzustellen, ist dies bei einer Ballkollision relativ einfach. Dafür benötigen wir die Distanz zwischen den beiden Bällen. Wenn diese Distanz kleiner ist als die Summe der Radien der Bälle, findet eine Kollision statt.

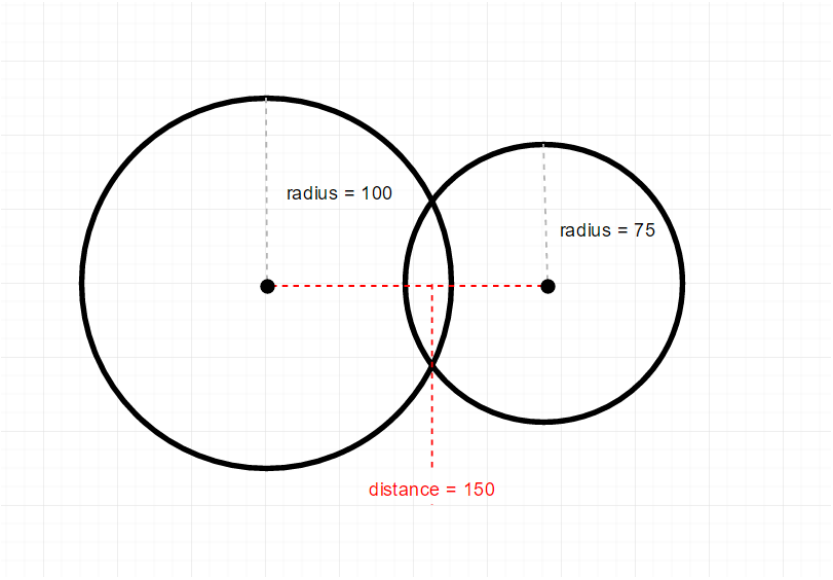

Abbildung 1: Kollision

3.2 Impuls und Impulserhaltungssatz

Die folgende Gleichung, bekannt als Impulserhaltungsgesetz stellt die Erhaltung der kinetischen Energie bei einem elastischen Stoß zwischen zwei Körpern dar.

$$m_1 v_1^2 + m_2 v_2^2 = m_1 v_1'^2 + m_2 v_2'^2$$

- m_1 und m_2 sind die Massen der Körper.
- v_1 und v_2 sind die Geschwindigkeiten der Körper vor dem Stoß.
- v_1' und v_2' sind die Geschwindigkeiten der Körper nach dem Stoß.

3.3 Effet

4 Algorithmen zur Kollisionsdetektion

4.1 Bruteforce

Vor- und Nachteile

4.2 Quad-Tree

Vorteile gegenüber Bruteforce

5 Visualisierungen und Demonstrationen

5.1 Vektorenrichtung Drawer

5.2 Tunneling-Demonstration

5.3 Impuls-Demonstration

5.4 Effet

6 Fazit

7 Quellen

[1]: https://www.leifiphysik.de/mechanik/impulserhaltung-und-stoesse/grundwissen/zentraler-elastischer-stoss [2]: https://happycoding.io/tutorials/processing/collision-detection