Physikalisch-basierte Simulation in der Computergraphik

Schriftliche Ausarbeitung

Titel des Projektes

Liste der Studierenden

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

1 Einleitung

Detailierte Instruktionen sind in der Englischen Vorlage zu finden.

2 Theorie

Bei der Geglaetteten Teilchen-Hydrodynamik (SPH) wird die zu simulierende Fluessigkeit durch eine endliche Anzahl an Partikeln diskretisiert. Die Partikel interagieren innerhalb eines bestimmten Radius h miteinander und besitzen eigene physikalische Eigenschaften wie Dichte, Masse oder Druck.

Die Nachbarpartikel eines Partikels sind die Menge der Partikel, welche sich innerhalb von h befinden und somit Einfluss auf den Partikel besitzen. Die Suche dieser Nachbarpartikel stellt einen sehr aufwaendigen Schritt des SPH-Algorithmus dar. Um diese effizient zu finden, teilen wir den Raum in ein uniformes Gitter auf. Eine Zelle besitzt die Groesse des Radius h, sodass die Nachbarpartikel nur in den anliegenden 26 Zellen und in der Zelle des Partikels gesucht werden muessen. Durch ihre Positionen im Raum werden die Partikel je einer Zelle zugewiesen. Dies erfolgt durch Spatial Hashing, damit eine endliche Anzahl an Zellen fuer einen unendlich grossen Raum ausreicht. Dann koennen die Partikel anhand ihrer zugewiesenen Zelle sortiert werden. Fuer jede Zelle wird der Abstand zu dem ersten Partikel der Zelle gespeichert. Diese Idee wurde 2008 von Nvidia vorgestellt. Nvi

Die Dichte wird dabei wie in der Gleichung 1 mithilfe eines Poly6 Smoothing Kernels W_{ij} aus Gleichung 2 berechnet.

$$\rho_i = \sum_i m_j W_{ij} \tag{1}$$

$$W_{ij} = \frac{315}{64\pi h^9}(h^2 - r^2)^3 \tag{2}$$

Der Druck p_i wird mit der Gaskonstanten K und dem Referenzdruck ρ_0 in Gleichung 3 berechnet.

$$p_i = K(\rho - \rho_0) \tag{3}$$

Die Force f aus Gleichung 4 kann als Addition der drei einzelnen Forces pressure, viscosity und external berechnet werden.

$$f = f^{pressure} + f^{viscosity} + f^{external} \tag{4}$$

TODO Force

Der Raum der Partikel wird durch eine quadratische Box begrenzt. Kollisionen mit den Seiten der Box werden durch einfache Positionsabfragen behandelt. Diese verhindern, dass die Partikel die Begrenzungen ueberschreiten. Ausserdem werden die Geschwindigkeiten der entsprechenden Richtungen umgekehrt. Eine Daempfungsvariable wird eingefuehrt, die die Geschwindigkeit dabei zusaetzlich abmildern kann.

3 Implementierung

Das Projekt wird in der Laufzeit und Entwicklungsumgebung Unity implementiert. Dazu wird ein sogenanntes GameObject erstellt, welches ein eigens geschriebenes C#-Skript zugewiesen wird. Dieses Skript wird verwendet, um die Simulation zu initialisieren und aufzurufen. Ein Zeitschritt wird mithilfe der Update-Funktion von Unity einmal pro Frame durchgefuehrt. Der eigentliche SPH-Algorithmus wird auf sieben Shader aufgeteilt, welche von der GPU ausgefuehrt werden. Es ist je ein Shader dafuer zustaendig, die Buffer zu initialisieren, die Partikel in Zellen einzuteilen, die Partikel anhand ihrer Zelle zu sortieren, die Dichte und im Anschluss die Force eines Partikels zu berechnen und zum Schluss den Integrationsschritt durchzufuehren.

Die Positionen in x-,y- und z-Richtung werden gehasht, um daraus den Zellindex zu bestimmen. Um die Anzahl der Hash-Kollisionen moeglichst gering zu halten, werden 8-stellige Primzahlen und eine grosse Anzahl an Partikeln - und damit auch Zellen - benutzt.

Fuer das Sortieren der Partikel anhand ihrer zugewiesenen Zelle wird eine fertige Implementierung des Algorithmus Bitonisches Sortieren verwendet. Dieser Sortieralgorithmus kann parallel auf der GPU ausgefuehrt werden.

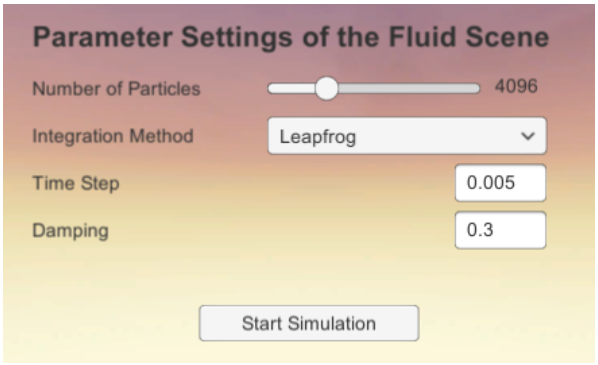

Fig. 1: Oeberflaeche vor dem Starten der Simulation, um die entsprechenden Parameter der Simulation einzustellen.

Das Einbeziehen der Nachbarpartikel bei der Berechnung der Dichte und im Anschluss der Force wird im Dichte- und im Force-Shader auf aehnliche Weise implementiert. Hierbei muss ueber die 27 moeglichen Nachbarzellen iteriert werden. Mithilfe des zuvor gespeicherten Abstands kann dann effizient auf alle Partikel einer bestimmten Nachbarzelle zugegriffen werden. Zusaetzlich wird ueberprueft, ob der Abstand zwischen den Partikeln kleiner als der Radius h ist.

Im Integrations-Shader wird neben der Berechnung der Position und der Geschwindigkeit auch die Kollision mit der Box behandelt.

Die Partikel werden in der Update-Funktion in jedem Frame gerendert. Dazu wird eine von Unity vorgegebene Zeichen-Funktion aufgerufen, welche das gleiche Kugel-Mesh parallel auf der GPU zeichnet. Dadurch verhindern wir den unnoetigen Overhead, eigenstaendige GameObjects pro Partikel zu erstellen. In diesem Schritt werden die Partikel anhand ihrer Dichte in einen blassen bis kraeftigen Blauton gefaerbt.

Neben dem GameObject, welches die Simulation behandelt, werden auch weitere Objekte benoetigt. Der Raum, in welchem sich die Partikel bewegen koennen, wird durch eine Box begrenzt, welche zur Laufzeit erzeugt wird. Die Box wird dabei durch ein eigenes Skript kontrolliert, welches die Positionen der Seitenplatten anhand einer vorgegebenen Bodenplatte berechnet. Ausserdem werden zwei Oberflaechen benoetigt, um Parameter einerseits vor dem Beginn der Simulation wie in Bild 1 und andererseits waehrend der laufenden Simulation wie in Bild 2 einstellen und anpassen zu koennen. Dazu werden fertige UI-Elemente von Unity verwendet. Beim Starten der Simulation findet ein Szenenwechsel statt. Dadurch werden nicht mehr benoetigte Objekte entfernt und neue Objekte eingeblendet.

4 Ergebnisse und Evaluierung

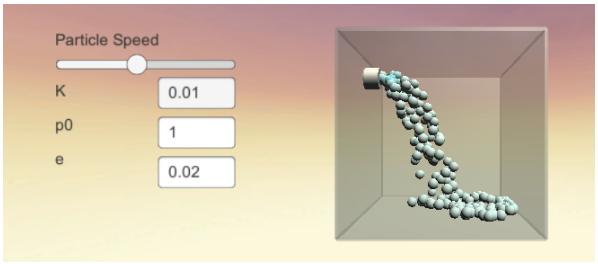

Fig. 2: Oeberflaeche waehrend der Simulation, um die Parameter des SPH-Algorithmus anzupassen.

5 Beiträge

Kirill Menke hat ueber den SPH-Algorithmus recherchiert und die Methode gefunden, wie effizient auf die Nachbarpartikel zugegriffen werden kann. Er hat das Grundgeruest der Simulation in Unity aufgesetzt, sodass die einzelnen Schritte des SPH-Algorithmus auf der GPU ausgefuehrt werden koennen. Zusaetzlich hat er das Rendering der Partikel implementiert und bei der Umsetzung des SPH-Algorithmus mitgewirkt.

Linda Stadter hat ebenfalls bei der Umsetzung des SPH-Algorithmus mitgewirkt. Zuesatzlich hat sie sich um das Erzeugen der Box und die Kollisionen der Kugeln mit den Seiten gekuemmert. Ausserdem hat sie die beiden Oberflaechen umgesetzt.

Peter Wichert

6 Diskussion

Durch das Verwendung von Spatial Hashing und der Sortierung nach Zellen laesst sich die Laufzeit der aufwaendigen Suche der Nachbarpartikel von $O(n^2)$ auf $O(n)$ reduzieren. In Kombination mit der parallelen Ausfuehrung des SPH-Algorithmus auf der GPU lassen sich grosse Anzahlen an Partikeln in Echtzeit simulieren.

Hash-Collisions lassen sich leider nicht vermeiden und fallen bei einer sehr geringen Anzahl an Partikeln auf. Daher werden diese geringen Partikelanzahlen von der Simulation ausgeschlossen.

7 Zusammenfassung

Literatur

Particle-based fluid simulation. `http://developer.download.nvidia.com/presentations/2008/GDC/GDC08_ParticleFluids.pdf`. Accessed: 2021-02-24.