Chirurgische Simulationen in VR mit Position Based Dynamics und Game Engines

Maximilian Legnar

Master-Thesis

Studiengang Angewandte Informatik
Fakultät für Mathematik und Informatik

Institut für Wissenschaftliches Rechnen (IWR)
Ruprecht-Karls-Universität Heidelberg

01.11.2020 - 30.04.2021

Betreuer
Prof. Dr. Jürgen Hesser
Prof. Dr. XXX

Legnar, Maximilian:
Chirurgische Simulationen in VR mit Position Based Dynamics und Game Engines / Maximilian Legnar. –
Master-Thesis, Heidelberg: Ruprecht-Karls-Universität Heidelberg, 2021. 9 Seiten.

Legnar, Maximilian:
Surgery simulations in VR with Position Based Dynamics and game engines / Maximilian Legnar. –
Master-Thesis, Heidelberg: Ruprecht Karl University of Heidelberg, 2021. 9 pages.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit ist in gleicher oder vergleichbarer Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Universität Heidelberg öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Heidelberg, 01.11.2020 - 30.04.2021

Maximilian Legnar

Abstract

Chirurgische Simulationen in VR mit Position Based Dynamics und Game Engines

Ziel: Potential von PBD für serious games für chirurgischen Bereich untersuchen. Kann mit handelsüblichen VR Geräten realistische Operations-Szenarien nachgebildet werden?

NVIDIA Flex wird mit Spiele Engine (Unreal Engine) um Funktionalitäten erweitert, die für Serious Games benötigt werden, ohne Flex-Bibliothek zu verändern.

(Brücke zwischen PnysX und Flex hergestellt...)

Use Case in dieser Arbeit: Werkzeuge können in VR mit virtuellen Händen aufgehoben und benutzt werden. Mit Werkzeugen können soft bodies manipuliert werden und virtuelle Operationen durchgeführt werden. Werkzeuge: Nadel und Pinzette.

Nadel-Gewebe-Interaktionen untersucht. Wie kann man das mit dem gegebenen Modell gut erreichen?

Surgery simulations in VR with Position Based Dynamics and game engines

... Abstract in English kommt hier hin...

Inhaltsverzeichnis

Abkürzungsverzeichnis

PBD Position Based Dynamics

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

teste quellen ref: [Sur, 2021]

1.1 Motivation und Problemstellung

1.2 Einordnung und Abgrenzung

1.3 Aufbau dieser Arbeit

Kapitel 2

Grundlagen

zuerst Anforderungen klären, dann theoretische Grundlagen und benutzte Software Werkzeuge erläutern...

2.1 Anforderungen

Folgende Anforderungen werden an das zu entwickelnde System und dieser Arbeit gestellt: ...

Interaktion und Echtzeit

...

Performance

Simulationsgenauigkeit

Muss nicht so gut sein.

2.2 Chirurgische Simulatoren

Was gibts da? was wird da benutzt? meist partikel basiert, technischen stand klären...

2.3 Simulationstechniken für chirurgische Simulatoren

2.3.1 Position Based Dynamics

spätestens hier Position Based Dynamics (PBD) einführen.

2.3.2 Erweiterungen von PBD

HPBD

Unified Particle Dynamics und NVIDIA Flex

zu erwähnen über [Macklin et al., 2014] und Flex:

Partikel sind über constraints miteinander verbunden.

Unified solver gut weil es einfacher zu handeln ist. Nur ein solver muss optimiert werden und vor allem weil dann objekte unterschiedlicher Art (soft, rigid, fluid) miteinander interagieren (collision constraint gillt für alle) können, aut of the box, in Echtzeit.

single particle radius sorgt für effizentere kollisionserkennen (wieso? nochmal nachlesen!)

Constaints: Distance(cloth) shape-matching(rigid, plastic/elastic deformation, siehe paper fig 5) density(fluid) volume(inflatables), friction(wichtiges constraint für nadel, flex is stolz auf friction constraint, siehe sand-teepot-demos), different collisions...

solver: PBD-solver, [Macklin et al., 2014] constraint-projektion als optimization problem, siehe kap 4.2, Gleichung (7). Bei jedem sim-step PBD versucht die Distanz zur Constraint-Verteilung (constraint manifold) zu minimieren unter berücksichtigung der partikel massen. Das tolle an dieser optimization sichtweise ist, das wir den PBD solver mit anderen solvern veergleichen können. So erkennt man das implizit euler recht ähnlich zu PBD ist. Unterschied: euler macht die minimization with respect zur initialen condition. bei BPD with respect to last iteration.

Parallelisierung mit gauss-jacobi solver (statt gaus-seidel aus PBD, der sequenziel arbeitet), der convergiert dann allerdings häufig nicht, z.b. wenn 2 partikel über identisches constraint verbunden sind. lösung: constraints paralell lösen, constraint

deltas für jedes partikel summieren und durch n=anzahl constraints teilen. Convergiert gut.

Kapitel 3

NVIDIA FleX und Unreal Engine 4 für Serius Games

Hier kurz erläutern was das is, was das kann usw...

3.1 Evaluation

Die verwendete Unreal Engine 4 mit FleX-integration wird zunächst ausführlich evaluiert. Die Evaluation dient dazu, einen Eindruck davon zu bekommen, wie gut die Software für die Entwicklung von Serious Games im chirurgischen Bereich geeignet ist und ob noch fehlende Funktionalitäten hinzugefügt werden sollten.

3.1.1 Performance

NVIDIA FleX zeichnet sich laut diversen Quellen ([Fle, 2021], [Macklin et al., 2014], HIER NOCH MEHR REIN!) durch eine besonders gute Performance aus, wobei in [Fle, 2021] genauer darauf eingegangen wird, welchen Einfluss unterschiedliche Größen auf die Geschwindigkeit des Flex-Solvers haben.

So wirkt sich die Partikelanzahl weniger stark auf die Performance aus, wie der Partikelradius oder die Anzahl der verwendeten Constraints [Fle, 2021]. Befinden sich beispielsweise nur Soft Bodies in einer Szene, muss der Solver deutlich weniger Constraints lösen, als in einer Szene, in der sich Soft Bodies, Flüssigkeiten und Stoffe befindet.

Außerdem kommt es auch auf die Komplexität der verwendeten Constraints an. Die Constraints **X, X und X** sind beispielsweise deutlich rechenaufwändiger, als ein einfaches **X-Constraint** (siehe [Fle, 2021]).

Zunächst wollen wir untersuchen, ob die Performance von Flex auch zusammen in einer Game Engine zufriedenstellend ist.

Für chirurgische Simulationen sind vor allem Soft Bodies interessant. Daher wurde eine Szene simuliert, in der sich ausschließlich Soft Bodies befinden. Dabei haben sich die Messdaten aus Abbildung 3.1 ergeben.

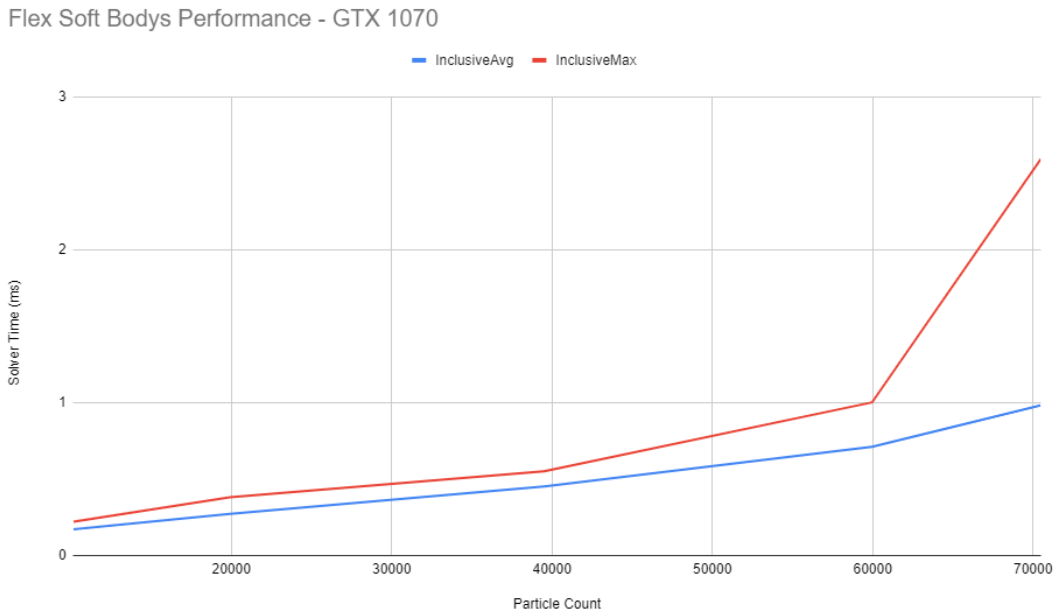

Abbildung 3.1: TEMP!!! Performance Messungen bei unterschiedlichen Partikelanzahlen mit einem Partikelradius von 10cm. Die durchschnittliche Solver Zeit steigt flach und linear mit der Partikelanzahl an.

LESEZEICHEN -> Hinweis, dass Messungen mit deaktiviertem sleep enthoben wurden!

Dann Messungen präsentieren:

Dann nochmal mit kleineren Partikeln und mehr substeps simulieren, weil das besser auf chirurges-szenario passt. In einigen anderen Arbeiten wurden ja auch recht kleine radi verwendet und mehr substeps usw (könnte man auch kurz drauf verweisen, auf diese paper und argumentieren das wir daher besser auf solche szenarien achten sollten)

3.1.2 Fehlende Funktionalitäten

Im Abschnitt *Introduction* aus [UE4, 2021] wird bereits erwähnt, dass die Unreal 4 Integration noch nicht weit genug entwickelt ist, um Gameplay-beeinflussende

Physikszenarios zu erstellen. Dafür fehlen noch Funktionalitäten [UE4, 2021], die für

3.2 Erweiterung

Um welche Funktionalitäten habe ich (und UE4) Flex erweitert und wie?

- Einfachere Benutzung von Flex mit BP-Script schnittstellen.

- Kollisionserkennung.

- FlexSynchronizeComponent

- Attachment.

- Kräfte/Impulse zwischen Flex und PhysX. (mit hinweis, das reaktionsktraft besser auf gpu berechnet werden sollte. cpu war halt einfacher)

Kräfte und Momente auf Flex-Objekte.

Transformationen von FlexComponents.

In [UE4, 2021] wird empfohlen, fehlende Kollisionserkennungen mit einer traditionellen Rigid Body Physics Engine zu realisieren. Weil die Unreal Engine 4 bereits solch eine Physic Engine besitzt (zitieren! verlinken!!!), bietet es sich an, die fehlende Kollisionserkennung mit dieser zu realisieren.

Kapitel 4

Implementation von Chirurgischen Szenarien

Welche Use Cases habe ich implementiert und wie? Und wie klappte das?

4.1 Soft Body Interaktion mit Motion Controller

Idee und Umsetzung von “VR-Physics2.0”, wie ich es genannt habe

4.2 Nadel-Gewebe Interaktion

Wie umgesetzt, welche Tricks angewandt? Constraints? Geeignete Simulationsparameter?

4.3 Weitere Chirurgische Szenarien

Häute, schneiden und Reißen, Pinzette SPH und Körperflüssigkeiten?

Kapitel 5

Fazit und Ausblick

Literaturverzeichnis

[Fle, 2021] (2021). D3d async compute for physics: Bullets, bandages, and blood. gdcvault.com/play/1024344/. aufgerufen: 15:00, 5 März 2021.

[Sur, 2021] (2021). Surgical simulators. wiki.tum.de/display/btt/Group+4%3A+Surgical+Simulators. aufgerufen: 14:30, 4 März 2021.

[UE4, 2021] (2021). Unreal 4 flex integration - documentation. gameworksdocs.nvidia.com/FleX/1.2/ue4_docs/FLEXUe4_Intro.html. aufgerufen: 13:30, 5 März 2021.

[Macklin et al., 2014] Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4).