GPU를 이용한 대용량 3D 메쉬 모델에 대한 병렬 자체 충돌검사

박성훈[1], 김양은[2], 최유주[2]
[1]한국전자기술연구원
[2]서울미디어대학원대학교 인공지능 응용소프트웨어학과
buff22@naver.com, yang7319@naver.com, yjchoi@smit.ac.kr

Parallel Self-Collision Detection for Large 3D Mesh Model using GPU

Sung-Hun Park[1] Yangen Kim[2] Yoo-Joo Choi[2,*]
[1]Korea Electronics Technology Institute
[2]AI Software Engineering, Seoul Media Institute of Technology
* Corresponding Author

요 약

본 논문은 3D 프린팅 출력 성공률을 높이기 위해 GPU를 이용한 대용량 3D 메쉬 모델에 대한 병렬 자체충돌 검사 방법을 제안한다. 강인하고 견고한 자체 충돌 검사를 위해 분리축 검사, 삼각형-삼각형 교차 검사, 메쉬 연결성 검사, 대용량 메쉬를 위한 분할 처리 기법의 절차를 제안한다. 이러한 자체 충돌 검사를 빠르게 수행하기 위하여 GPU 기반 병렬처리 구현 방법을 제시한다.

1. 서론

3D 프린팅은 보다 정교하고 복잡한 형상을 만들 수 있으며 다품종 소량 생산 등 특화 및 맞춤형 제작이 가능한 장점이 있어 산업 전반으로 영향력을 넓히고 있다[1,2]. 또한, 3D 프린팅은 여러 부품이 연결된 상태로 제조가 가능하고 내부 형상이 복잡해도 제조가 가능한 강점을 기반으로 건축·조선·발전에 사용되는 대형 부품 생산에 적용되고 있다.

3D 프린팅의 출력 성공률을 높이기 위해 출력 모델을 회전시키거나 출력 모델을 편집한다. 하지만 포맷 변환은 때때로 메쉬 오류를 불러온다. 또한, 출력 모델을 편집하면서도 메쉬 오류가 발생하기도 한다. 출력 실패를 줄이기 위해서는 메쉬 오류 수정이 필요하다. 메쉬 오류를 수정하는 알고리즘은 많이 연구되고 있으며 주로 연산량 최적화에 초점이 맞춰져있다. 반면, 대형 3D 프린팅 추세에 맞춰 생성되는 3D 출력 모델 크기도 같이 증가하고 있다. 이러한 요구에 따라 본 논문에서는 많은 연산량을 빨리 처리하기 위해서 병렬처리에 강점이 있는 GPU를 사용해 메쉬 오류를 검출하는 방법을 제안한다.

이러한 GPU 병렬처리와 손쉬운 STL 메쉬 표현을 위해 유니티를 기반으로 실험환경을 구축하였다. 3D 프린팅 작업 단계에서 발생할 수 있는 메쉬 오류 중 하나인 자체 충돌(Self-Collision)을 대용량 3D 메쉬 파일(STL)에서 GPU 병렬처리를 이용해 검출하는 방법을 고안하였다.

2. 병렬 자체 충돌검사

2.1 제안 방법의 처리 절차

자체 충돌을 견고하게 검출하기 위하여 메쉬를 구성하는 모든 삼각형에 대하여 삼각형-삼각형 교차 검사 (Triangle-Triangle Intersection Test)[3]를 수행한다.

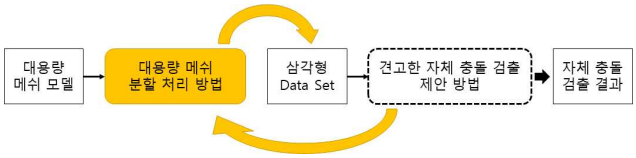
본 제안 방법에서는 단순화된 분리 축 검사 (SAT : Separating Axis Test)[4,5]를 사용해 삼각형 간 비교 쌍을 줄여 연산 효율을 높였다. 삼각형-삼각형 교차 검사는 삼각형간의 충돌을 검사하기

때문에 메쉬 모델과 같이 삼각형간 연결성(Connectivity)이 있는 상황이 고려되지 않았다. 맞닿아있는 이웃한 삼각형인 경우에도 알고리즘은 충돌로 판정한다. 따라서 삼각형간 충돌에 대한 잘못된 판정을 줄이기 위해 메쉬 연결성 검사를 추가로 진행한다.

앞서 논의한 세부 절차들을 합쳐 구성한 견고한 자체 충돌 검출을 위한 제안방법은 (그림 1)과 같다. 해당 방법은 효율적인 연산을 위해 병렬처리가 가능한 GPU의 계산 셰이더를 사용해 구현한다.

(그림 1) 견고한 자체 충돌 검출 제안 방법 세부 절차

대용량 메쉬 모델에 대해 분할하여 자체 충돌검사를 수행하기 위해서, GPU 계산 셰이더에서 이용되는 삼각형 데이터 집합을 재구성하여 GPU 병렬처리가 가능한 최대치 만큼 대용량 메쉬 모델을 분할해서 자체 충돌을 검출하도록 하였다. 대용량 메쉬는 분할된 만큼 반복하여 삼각형 데이터 집합을 GPU 에 갱신하여 견고한 자체 충돌 검출을 진행한다. [그림 2]는 대용량 메쉬 모델에 대한 분할 처리 절차를 보여주고 있다.

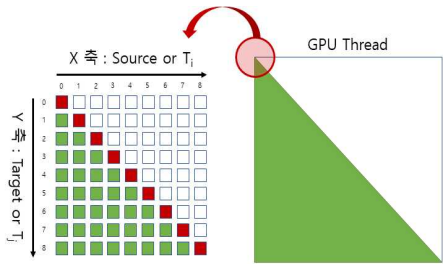


(그림 2) 대용량 메쉬 모델 처리를 위한 분할 처리 방법

본 논문에서는 (그림 2)의 절차에서 메쉬 모델 분할 처리에 대한 세부 사항만을 설명하도록 하겠다.

2.2 대용량 메쉬 모델 분할 처리

삼각형-삼각형 교차 알고리즘을 GPU에서 병렬처리하기 위해서는 계산 셰이더(Compute Shader)를 GPU 스레드(Thread)로 처리할 수 있도록 설계해야 한다. 가장 쉬운 방법으로는 (그림 3)와 같이 계산 셰이더에서 GPU 스레드 X와 Y축을 비교해야하는 삼각형에 맵핑하는 방법이 있다. X축을 비교해야하는 원본 삼각형으로, Y축을 비교해야하는 삼각형으로 설정하면 계산 셰이더에서는 n^2의 수행비용으로 전체 삼각형 검사를 할 수 있다.



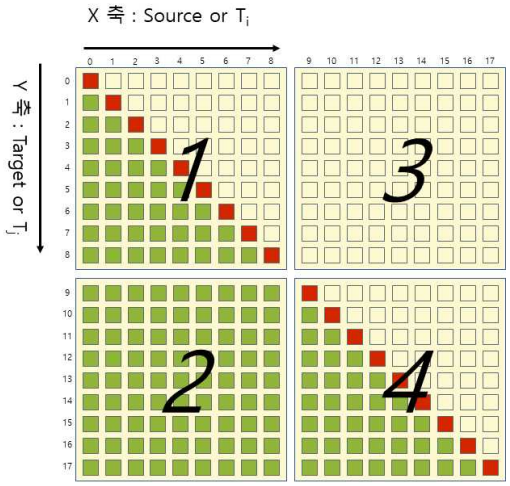
(그림 3) 병렬처리를 위한 GPU Thread 모습

본 논문에서는 단일 강체에 대한 자체 충돌 검사를 진행하기 때문에 X와 Y축을 동일한 삼각형으로 맵핑하면 된다. 따라서 X와 Y의 동일한 번호는 동일한 삼각형이므로 자체 충돌 검사를 진행할 필요가 없다. 즉, (그림 3)에서 빨간색 사각형으로 표현된 스레드는 자체 충돌검사를 진행할 필요가 없는 스레드를 표현하고 있다. . 또한 X와 Y 번호가 서로 바뀌어도 동일한 쌍이기에 검사가 중복되게 된다. 따라서 X가 Y보다 큰 경우 자체 충돌 검사를 진행하지 않고 종료하면 검사 수행비용이 n^2에서 $\frac{n(n-1)}{2}$으로 줄어들게 된다. 최종적으로 검사하는 자체 충돌 검사 쌍은 (그림 3) 초록색으로 나타난 부분이 된다.

다음은 대용량 메쉬를 분할 처리하는 알고리즘이다.

Algorithm 1 : Big Mesh Partition
Input : numFace, Max_Thread_X, Max_Thread_Y
Output : Count of Collision Detection

(1) maxTi = numFace / Max_Thread_X
(2) maxTj = numFace / Max_Thread_X
(3) **Loop** maxTi
(4) **Loop** maxTj
(5) idxTi = i * Max_Thread_X
(6) idxTj = j * Max_Thread_Y
(7) lengthTi = min(numFace - idxTi, Max_Thread_X)
(8) lengthTj = min(numFace - idxTj, Max_Thread_Y)
(9) **Update GPU Buffer**(idxTi, idxTj, lengthTi, lengthTj)
(10) **Dispacth**
(11) **Download Count of Collision Detection**
(12) **Endloop**
(13) **Endloop**



(그림 4) 대용량 메쉬 분할 처리 GPU 스레드 예시

(그림 4)는 GPU 스레드 최대 개수가 X축 9개, Y축 9개일 때 메쉬 크기가 18개인 경우를 가정해 설명한다. 계산 셰이더를 이용한 계산은 총 4번이어야 하지만, 앞서 살펴본 알고리즘 1 (4)에서 반복 loop의 시작 번호를 (3)의 반복 숫자로 지정 할 수 있다. 그러면 3번 영역을 계산하지 않고 1, 2, 4번 영역만 계산 셰이더로 연산해 효율을 높일 수 있다. 이러한 계산 셰이더는 알고리즘 1의 (7)과 (8)의 삼각형 정보 길이가 GPU에 갱신되어 보다 효율적으로 계산할 수 있도록 한다.

4.실험 결과 분석

4.1 실험데이터 및 실험 환경

3D 프린팅 관련 알고리즘을 검증하기 위해서는 실험 데이터 집합이 필요하다. 이상적인 실험 데이터 집합은 그 특성 분포가 실제 사용되는 3D 프린팅 출력물의 특성 분포와 비슷한 데이터 집합인데 이러한 실험 데이터를 Thingi10K[16]를 통해서 확보하였다. <표 1>은 실험 모델을 보여주고 있다.

<표 1> 실험군 3 (삼각형 개수 1,000,000 이상)

이름	Car	Skull	Venus
형태			
Face 개수	1,039,452	2,714,120	3,154,110

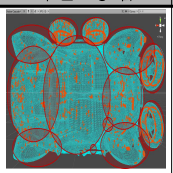
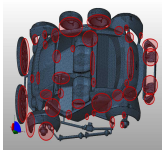
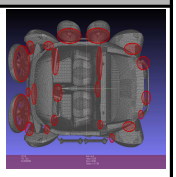
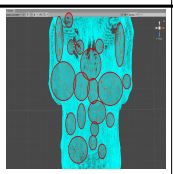
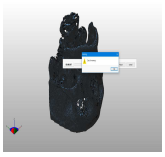
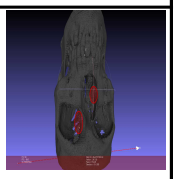
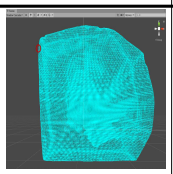
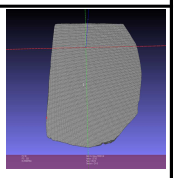
본 논문의 실험 환경은 유니티 2020.3.18f1 버전에서 구현되었으며, 실험은 Intel i7-9750HF 2.60GHz와 메모리(RAM) 16GB, 그래픽 처리 장치(GPU)는 메모리 6GB의 NVIDIA GeForce GTX 1660 Ti가 탑재된 시스템에서 진행했다. 실험은 GPU Thread Group의 수를 4500으로 설정하고 실험한 결과이다.

4.2 대용량 메쉬 분할 처리 알고리즘 성능 실험

netfabb은 현재 3D프린팅 세계적 강자 중 하나인 Autodesk사(社)에 인수합병 되었으며, Basic이외 버전은 연간 라이선스를 통해 사용되는 3D프린팅 SW중 하나이다. Mesh Repair에 강점이 있는 SW로 유명하다.

MeshLab은 ISTI-CNR이라하는 이탈리아 연구기관이 이탈리아 피사 대학교(University of Pisa) 학생들과 함께 개발한 SW이다. GPL라이선스를 가지고 있어 3D프린팅 SW 개발에 많이 사용되고 있다. <표 2>는 본 논문에서 제안한 방법과 두 상용 SW에서 자체 충돌 감지를 실행한 비교 결과이다.

<표 2> 실험군 3 실험 결과 비교

	제안 방법	Netfabb	MeshLab
Car (1,039,452)	 38,486ms (3438개)	 29,000ms	 3,346ms
Skull (2,714,120)	 250,342ms (442개)	 메모리 오류	 10,222ms
Venus (3,154,110)	 337,608ms (3개)	모델 Load 안 됨	 20,624ms

상용 SW의 경우 자체 충돌 검출 결과를 시각적으로만 보여줄 뿐 다른 정보로 제공하지 않았다. 제안 방법의 경우 시각적인 표현뿐만 아니라 자체 충돌이 발생한 삼각형 쌍의 번호를 보여주고, 그 번호들을 텍스트 파일로 저장하여 사용자가 확인할 수 있도록 하였다. 상용 SW의 경우 삼각형 2백만 개 이상 모델부터 정상적인 자체 충돌 검사가 어려웠으나 제안 방법의 경우 3백만 개 이상의 모델에서도 정상적으로 자체 충돌 오류를 검출하는 것을 확인 하였다.

다만, 자체 충돌 검사를 견고하게 진행함으로 인해 상대적으로 상용 SW보다 메쉬 크기가 커질수록 수행 속도가 많이 느려지는 단점 확인 되었다.

5. 결론

본 연구는 3D 프린팅 전처리 단계에서 발생 할 수 있는 메쉬 오류 중 하나인 자체 충돌을 검출하기 위해 GPU 기반 대용량 메쉬 병렬처리 방법을 제시했다. 상용 SW의 경우 2백만 개 이상의 모델에서 자체 충돌 검사가 제대로 수행되지 않음을 실험을 통해 확인하였다. 제안 방법은 3백만 개 이상 모델에서도 자체 충돌을 검출함을 확인하였다.

추후, 보다 효율적인 삼각형 도태(Culling)방법을 도입하면 상용 SW에 가까운 수행 속도 성능을 보여줄 것으로 예측된다.

참고문헌

[1] 정보통신산업진흥원, *2020년 3D 프린팅 산업 실태조사*, 2021

[2] REDWOOD, Ben; SCHÖFFER, Filemon; GARRET, Brian. *The 3D printing handbook: technologies, design and applications*. 3D Hubs, 2017.

[3] MÖLLER, Tomas. A fast triangle-triangle intersection test. *Journal of graphics tools*, 1997, 2.2: 25-30.

[4] ERICSON, Christer. Real-time collision detection. Crc Press, 2004.

[5] https://dyn4j.org/2010/01/sat/