자바 병렬 환경에서 동작하는 광선 추적기 구현

황정현, 김정훈, 안진호, 황종선
고려대학교 컴퓨터학과 분산시스템 연구실

Implementation of a Ray-Tracer on a Java Parallel Environment

Jeong-Hyon Hwang, Joung-Hoon Kim, Jin-Ho Ahn, Chong-Sun Hwang
Distributed Systems Lab., Dep. of Computer Science, Korea Univerisity

요 약

광선 추적법(Ray-Tracing)은 물체에 직·간접적으로 영향을 미치는 빛의 반사 및 굴절 경로를 역추적함으로써 실제감있는 이미지를 생성하는 렌더링(rendering) 기법이다. 이러한 광선 추적법은 장시간의 계산을 필요로 하는 단점이 있으나, 각각의 광선을 병렬적으로 추적함으로써 속도의 향상을 꾀할 수 있다. 본 논문에서는 자바를 사용하는 메시지 기반 병렬 프로그래밍 시스템인 JPVM 상에서 동작하는 병렬 광선 추적기를 구현하였다. 병렬 광선 추적기는 사용자에 의해 지정된 장면 정의 파일(Scene Definition File)을 읽어 들여 파싱(parsing)한 후, 생성된 장면 객체를 각 worker 프로그램에게 전송한다. 병렬 광선 추적기는 전체 화면 영역을 분할하여 각 worker 프로그램에 할당하며, worker 프로그램들은 자신에게 할당된 영역의 이미지를 병렬적으로 생성한다. 실험 결과, 병렬 광선 추적기는 단일 광선 추적기에 비해 빠르게 렌더링을 수행하였으며, worker 프로그램의 수가 증가함에 따라 수행 속도가 향상되었다.

1. 서 론

컴퓨팅 환경이 발전함에 따라 컴퓨터 그래픽은 다양한 분야에서 그 위력을 발휘하게 되었다. 3차원 공간상에 존재하는 입체적인 대상들을 화면에 출력해주는 3차원 그래픽은 빛과 각 대상들간의 관계를 이용해서 색상을 계산하며, 이를 위해 flat shading, Gauraud shading, Phong shading 등과 같은 여러 가지 shading 기법들이 제안되었다[1]. 그러나 이러한 기법들은 다중 반사 및 굴절된 이미지를 표현하는데 한계점을 가지고 있으므로, 정교한 이미지를 제공할 수 있는 광선 추적법(Ray-Tracing)이 제안되었다[1, 2].

광선 추적법(Ray-Tracing)은 관찰자의 눈으로 들어오는 빛을 추적하고, 물체에서의 반사와 굴절 현상을 시뮬레이션하는 렌더링 기법으로, 장시간의 계산을 필요로 하는 단점을 극복하기 위해서 각각의 광선을 병렬적으로 추적하는 방식들이 제안되었다[3, 4]. 본 논문에서는 자바를 사용하는 메시지 기반 병렬 프로그래밍 시스템인 JPVM[5] 상에서 동작하는 병렬 광선 추적기를 구현하였다 병렬 광선 추적기는 사용자에 의해 지정된 장면 정의 파일(Scene Definition File)을 읽어 들여 파싱(parsing)한 후, 생성된 장면 객체를 각 worker 프로그램에게 전송한다. 병렬 광선 추적기는 전체 화면 영역을 분할하여 각 worker 프로그램에 할당하며, worker 프로그램들은 자신에게 할당된 영역의 이미지를 병렬적으로 생성한다.

본 논문의 2장에서는 병렬 광선 추적기의 수행 환경인 JPVM에 대해서 소개하며, 3장에서는 광선 추적법에 대해서 언급한다. 4장에서는 구현된 병렬 광선 추적기의 알고리즘을 소개하고, 5장에서는 worker 프로그램의 개수를 증가시켜 가면서 광선 추적기와 단일 광선 추적기의 성능을 비교한다. 마지막으로 6장에서는 결론을 맺는다.

2. JPVM

네트워크에 연결된 이기종 컴퓨팅 시스템들을 사용하는 것은 이미 고성능의 수행을 위해서 폭넓게 사용되는 방식이 되었다. JPVM은 자바로만 구현된 네트워크 병렬 컴퓨팅 시스템으로, 플랫폼 독립적이며, 시스템 서비스에 대한 일관적인 인터페이스를 가짐으로써, 네트워크 병렬 어플리케이션 및 시스템 소프트웨어를 구현하기 위한 매력적인 환경을 제공한다[5].

JPVM은 PVM(Parallel Virtual Machine)[6]과 유사한 인터페이스를 가지며, 태스크 생성, 동기적·비동기적 메시지 전송을 수행하기 위한 라이브러리 루틴을 제공한다.

JPVM 환경을 구축하기 위해서는, 각종 JPVM 라이브러리 루틴의 수행을 지원하는 JPVM daemon 프로세스가 각 호스트에서 수동적으로 구동되어야 하며, 사용자는 JPVM 콘솔을 통해서 전체 JPVM 환경을 설정한다. 또한, JPVM 어플리케이션을 작성하는 프

로그래머는 해결하고자 하는 문제를 협력적인 여러 태스크로 분해하여 프로그램을 작성하고, 이 태스크들은 daemon 프로세스에 의해 생성되어 자바 가상 기계 상에서 동작하게 된다.

3. 광선 추적법

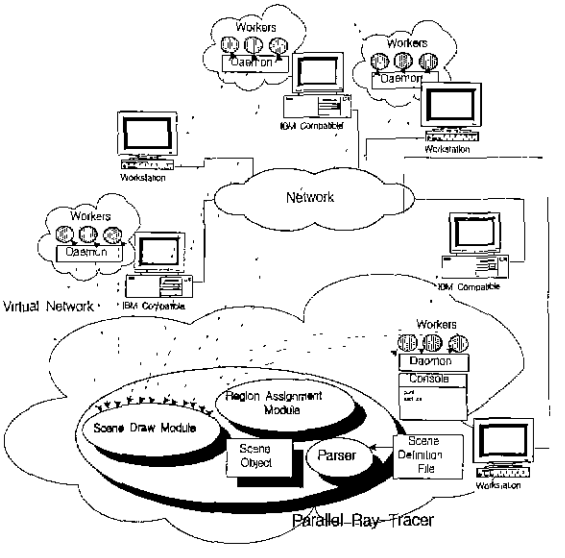
광선 추적법(Ray-Tracing)은 관찰자의 눈으로 들어오는 빛을 추적하고, 반사와 굴절 현상을 시뮬레이션 함으로써 사실적인 이미지를 얻을 수 있도록 해주는 렌더링 기법이다. 일반적으로 많이 사용되는 방식은 관찰자의 눈에서 시작해서 역으로 광원을 찾아내는 후진 광선 추적법(Backward Ray-Tracing)이다.

후진 광선 추적법에서는 광선의 경로를 역추적 하다가 물체를 만나게 되면, shading 모델을 통해 광선과 물체와의 교차점에 대한 색을 결정하고, 물체의 광학적 특징을 반영하여 반사, 굴절, 투과 현상을 시뮬레이션한다. 뿐만 아니라, 시뮬레이션을 통해서 생성된 반사광, 굴절광, 투과광 각각에 대해서도 광선 추적을 실시하며, 일정 회수를 넘지 않는 범위 내에서 광선과 물체가 교차하지 않을 때까지 이와 같은 과정을 반복적으로 수행한다.

광선 추적법은 많은 양의 수치적 계산을 필요로 하지만, 광선의 경로 추적은 타 경로와 관계없이 독립적으로 수행될 수 있으므로, 병렬 환경에서는 보다 효과적인 수행을 기대할 수 있다.

4. 병렬 광선 추적기의 구현

병렬 광선 추적기는 다수의 worker 프로그램을 실행하여 3차원 그래픽 이미지를 생성하는 렌더링 프로그램으로, 그림 1과 같은 기본 구조를 갖는다. 그림 2는 전반적인 병렬 광선 추적 알고리듬을 보여준다



[그림 1] 병렬 광선 추적 시스템 구성도

4.1 장면(Scene) 생성

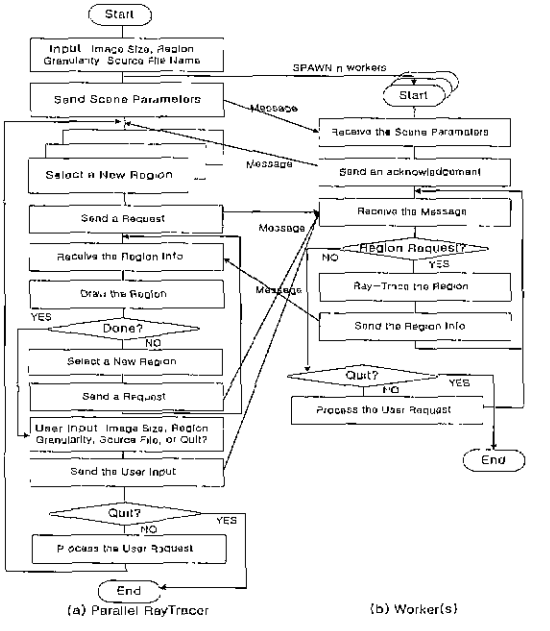
병렬 광선 추적기는 사용자가 정의한 장면 정의 파일(Scene Definition File) 및 장면 정의 파일 내에서 지정된 그래픽 이미지 파일(*.jpg, *.gif 등)들을 읽어 들인 후, 자체 구문 분석기(parser)를 사용해서 장면을 생성한다. 병렬 광선 추적기는 장면을 정의하기 위한 고유 문법을 가지고 있으며, 이러한 문법을 사용하여 관찰자의 위치, 빛, 평면, 구, 다면체 등을 정의한다. 평면인 경우에는 그림 2에서 보여주는 바와 같이 그래픽 파일을 이용해서 texture를 지정할 수 있다

```
BEGIN_SCENE
    VIEWPOINT 250,150,600
    LIGHTPOINT -500,-100,200
    AMBIENT 0.2,0.2,0.2
END_SCENE

LIGHT
    CENTER -500,-100,200
    RADIUS 120
    COLOR 1.1,1.0,0.9
    REFRACTIVENESS 0
    TRANSPARENCY 0
    REFLECTIVENESS 0.5
    DIFFUSION 0.5

SPHERE
    CENTER 359,149,-300
    RADIUS 200
    COLOR 1,1,1
    REFRACTIVENESS 1.5
    TRANSPARENCY 0.6
    REFLECTIVENESS 0.3
    DIFFUSION 0

PLANE
    CENTER 500,0,-400
    NORMAL -1,0,2
    COLOR 1,1,1
    TEXTURE Flower.JPG
    REFRACTIVENESS 0
    TRANSPARENCY 0
    REFLECTIVENESS 0
    DIFFUSION 1
```

[그림 2] 장면 정의 문법의 예

4.2 작업 초기화

병렬 광선 추적기는 n개의 worker 프로그램을 생성하며, 사용자가 입력한 정보(즉, 화면 크기, 화면 분할 단위 등)와 생성된 장면 객체를 메시지에 실어 모든 worker 프로그램에게 전송한다



[그림 3] 병렬 광선 추적 알고리즘

4.3 작업량의 분배

병렬 광선 추적기는 화면 분할 단위 크기에 따라 화면을 분할하며, 최초 n개의 요구를 각 worker 프로그램에게 전송한다. 병렬 광선 추적기는 worker 프로그램에 의해 계산된 이미지 정보를 수신하여 화면을 갱신하고, 메시지를 송신했던 worker에게 다시 새로운 화면 영역을 할당한다.

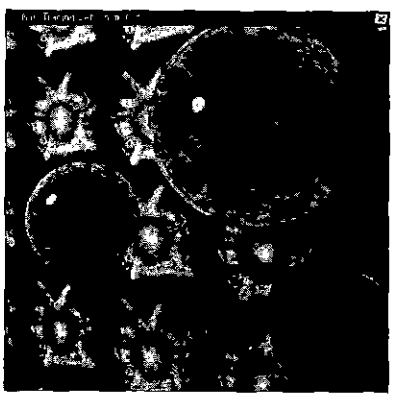
이미지 생성이 종료되면, 병렬 광선 추적기는 사용자의 입력을 받아들이며, 사용자 입력을 모든 worker 프로그램에게 전송하여, 각 worker 프로그램이 병렬 광선 추적기와 동일한 장면 정보를 가지도록 한다. 사용자가 종료를 요청하면, 종료 메시지가 각 worker 프로그램들에게 전송되며, 병렬 광선 추적기 환경이 종료된다.

4.4 Worker 프로그램

병렬 광선 추적기는 n개의 worker 프로그램을 생성하며, 생성된 worker 프로그램들은 작업 환경을 초기화하기 위한 메시지를 기다린다. 병렬 광선 추적기가 메시지를 전송하면, worker 프로그램은 작업 환경을 구축하고, 병렬 광선 추적기로부터 화면 영역 요청을 기다린다. 요청 메시지가 수신되면, worker 프로그램은 해당 영역에 대한 이미지를 계산하며, 그 결과를 메시지 형태로 병렬 광선 추적기에게 전송한다. 병렬 광선 추적기로부터 종료 메시지를 수신하면, 종료된다.

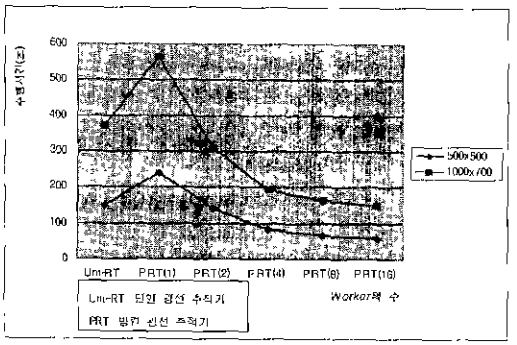
5. 수행 성능 비교

본 논문에서는 병렬 광선 추적기의 성능을 측정하기 위해 worker 프로그램의 수를 변경해 가면서 병렬 광선 추적기를 수행시켜 보았다. 실험 환경은 Ultra SPARK™ CPU 6개와 512 MB의 RAM을 갖는 Sun Enterprise 3000으로서, 500x500, 1000x700 두 가지 크기의 화면에 대해서 실험하였다.

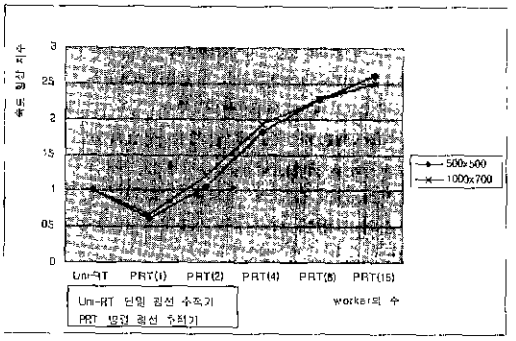


[그림 4] 생성된 이미지의 예 (500x500)

그림 4는 실험을 통해 생성된 이미지를 보여주며, 그림 5와 6은 worker 프로그램의 수를 증가시킴에 따라 변화하는 수행 시간과 속도 향상에 대한 그래프를 보여준다. 실험 결과, 병렬 광선 추적기는 2개의 worker 프로그램을 두었을 때, 단일 광선 추적기와 비슷한 수행 성능을 나타냈으며, 이는 메시지의 전송과 연관되는 오버헤드가 주요 원인인 것으로 판단된다. worker 프로그램의 수를 짝수 배로 16까지 증가시키는 실험을 통해서, 다음 그래프에서 보여주는 바와 같은 결과를 얻었다.



[그림 5] worker 프로그램의 개수 변화에 따른 수행 시간



[그림 6] worker 프로그램 개수 변화에 따른 속도 향상

6. 결론

본 논문에서는 자바를 사용하는 메시지 기반 병렬 프로그래밍 시스템인 JPVM 상에서 동작하는 병렬 광선 추적기를 구현하였으며, 실험을 통해 worker 프로그램의 수가 증가함에 따라 보다 더 빠른 속도로 이미지가 생성된다는 결과를 얻을 수 있었다. 병렬 광선 추적기를 구현함에 있어서, 기존의 JPVM은 버퍼링의 한계로 인해 대용량의 화면 정보를 전송할 수 없었다. 본 시스템은 이러한 문제점을 해결하기 위해 JPVM 소스 코드의 일부를 변경하여 재구성하였다.

참고문헌

[1] Alan Watt, "3D Computer Graphics," Addison-Wesley. 1993

[2] Nicholas Wilt, "Object-Oriented Ray Tracing in C++," O'reily, 1994

[3] John Stone, "An Efficient Library for Parallel Ray Tracing and Animation," 1995 Intel Supercomputer Users Group Conference, available from' http://www.cs.umr.edu/~johns/rayracer/isug95/isug.html

[4] 이효종, 임훈철, "트랜스퓨터 시스템에서의 병렬 광선추적 알고리즘," 정보과학회논문지(A), 제 23권, 제 9호, pp 924-933, 1996

[5] Adam J Ferrari, "JPVM: Network Parallel Computing in Java," Technical Report CS-97-29, Dep of Computer Science, Univ. of Virginia, 1997

[6] A. Geist, A. Beguelin, J. Dongarra, W Jiang, R Mancheck, and V. S Sunderam, "PVM' Parallel Virtual Machine," MIT Press, 1994