1. 개요

레이 트레이싱은 이미지 렌더링을 위한 간단하면서도 강력한 알고리즘입니다. 씬과 쉐이딩 모델의 정확도와 충분한 컴퓨팅 시간을 갖추면, 레이 트레이서가 생성한 이미지는 물리적으로 정확하고 실제 이미지와 구별할 수 없이 보일 수 있습니다. 그러나 당신의 레이 트레이서는 물리적으로 정확한 이미지를 생성할 수 없을 것입니다.

이 과제에서는 레이 트레이서가 다음을 지원할 것입니다:

• 구(50%)과 축에 정렬된 상자(0%)

• 램버티안(20%) 및 평 광택 쉐이딩(20%)

• 그림자가 있는 점 광원(10%)

• 임의의 원근 카메라(0%)

입출력(I/O), XML 파서, 벡터 연산 등을 구현하는 데 걸리는 시간을 절약하기 위해 프레임워크 코드(rayTracer.py - 60줄 미만)가 제공됩니다. 하지만 프레임워크는 실제 레이 트레이싱을 수행하는 코드를 포함하지 않습니다. 이 코드는 직접 개발해야 합니다.

레이 트레이싱 코드를 작성할 때, 최선으로 생각되는 설계를 결정할 수 있습니다. 프로그램에 몇 가지 클래스를 추가할 수 있으며, 계산의 다양한 부분을 어디에 놓을지에 대한 많은 선택지가 있습니다. 아래의 최소 요구 사항을 올바르게 충족하고 명확하게 작성된 모든 솔루션은 모두 인정됩니다. 예를 들어 광택 쉐이딩이 전혀 없는 구도 제출하는 경우에도 전체 크레딧의 80%를 받게 됩니다. 교재, 강의 및 교수를 비롯한 코스 스태프는 모두 레이 트레이싱 코드 작성에 대한 좋은 접근 방식에 대한 정보의 출처입니다.

다만 많은 양의 새로운 코드를 작성해야 한다는 뜻은 아닙니다. 예를 들어 프레임워크에는 약 60줄의 코드가 포함되어 있으며, **우리 솔루션은 세 개의 작은 추가 클래스와 약 200줄의 코드를 포함합니다.**

1. 요구사항
   1. 레이 트레이싱 알고리즘을 사용합니다.
   2. 구체를 지원합니다.
   3. Shirley 9.1-9.2 및 강의 노트에서 정의된 Lambertian 및 Blinn-Phong 셰이딩 모델을 지원합니다.
   4. 거리에 따라 감쇠하지 않는 조명을 제공하는 포인트 라이트를 지원합니다.

구문적으로나 의미론적으로 잘못된 입력에 대해서는 걱정하지 않아도 됩니다. 예를 들어 반지름이 음수인 구체나 카메라가 없는 장면은 제공되지 않습니다.

1. 파일 형식

레이 트레이싱에 대한 입력 파일은 XML 형식으로 구성됩니다. XML 파일은 HTML과 비슷한 꺾쇠 괄호 태그로 구분된 중첩된 요소 시퀀스를 포함합니다. 예를 들어, 다음과 같은 XML코드가 있습니다.

<scene>

<camera>

</camera>

<surface type=Sphere>

<center>1.0 2.0 3.0</center>

</surface>

</scene>

이 코드는 네 개의 요소를 포함합니다. 하나는 camera와 surface라고 불리는 두 개의 다른 요소를 포함하는 scene 요소입니다. surface 요소에는 값이 Sphere인 type 속성이 있습니다. 또한 center 요소가 있으며, 텍스트 "1.0 2.0 3.0"을 포함하며, 이 문맥에서는 3D 좌표 (1, 2, 3)으로 해석됩니다. 레이 트레이서의 입력 파일은 항상 하나의 scene 요소를 포함하며, 다음 유형의 태그를 포함 할 수 있습니다:

• surface: 이 요소는 기하학적 개체를 설명합니다. 값이 Sphere 또는 Box인 type 속성이 있어야합니다. 그리고 셰이더를 설정하는 shader 요소를 포함할 수 있으며, 그리고 그 유형에 따라 기하학적 매개변수도 포함할 수 있습니다:

* Sphere의 경우: 3D 좌표를 포함하는 center와 실수를 포함하는 radius.
* Box의 경우: 각각 3D 좌표를 포함하는 minPt와 maxPt입니다. 두 점이 (xmin, ymin, zmin)과 (xmax, ymax, zmax)인 경우 상자는 [xmin, xmax] × [ymin, ymax] × [zmin, zmax]입니다.

• camera: 이 요소는 카메라를 설명합니다. 다음 요소로 설명됩니다:

* viewPoint: 투영의 중심을 지정하는 3D 좌표입니다.
* viewDir: 카메라가 보는 방향을 지정하는 3D 벡터입니다. 크기는 사용되지 않습니다.
* viewUp: 이미지의 방향을 결정하는 데 사용되는 3D 벡터입니다.
* projNormal: 투사 평면의 법선을 지정하는 3D 벡터입니다. 그 크기는 사용되지 않으며, 방향을 반대로 하는 것은 영향을 미치지 않습니다. 기본적으로보기 방향과 동일합니다.
* projDistance: 이미지 직사각형의 중심에서 투영 중심까지의 거리를 나타내는 실수 d입니다.
* viewWidth와 viewHeight는 이미지 평면에서 보이는 창의 크기를 나타내는 두 개의 실수입니다.

카메라의 뷰는 투영 중심(뷰포인트)과 뷰포인트로부터 d 거리 떨어진 위치에 있는 viewWidth x viewHeight 크기의 뷰 창으로 결정됩니다. 이는 이미지 평면에 수직이며, 상하 가장자리가 up 벡터에 수직으로 방향이 지정됩니다.

• 이미지: 이 요소는 출력 이미지의 크기를 나타내는 두 개의 정수로 구성됩니다.

• 빛: 이 요소는 광원을 설명합니다. 3D점 위치와 RGB 색상 color을 포함합니다.

• 셰이더: 이 요소는 표면을 어떻게 쉐이딩할지 설명합니다. Lambertian 또는 Phong 값을 가진 type 속성을 가져야합니다. Lambertian 셰이더는 RGB 색상 diffuseColor를 사용하고, Phong 셰이더는 추가로 RGB 색상 specularColor와 실수 exponent를 사용합니다. 셰이더는 표면 요소 내부에 나타날 수도 있으며, 그 경우 해당 표면에 적용됩니다. 또한 직접 장면에 나타날 수 있으며, 이는 표면 내부에서 나중에 참조하려면 이름을 지정하고 ref 속성을 사용하여 참조를 포함하는 경우 유용합니다. 예를 들어:

<shader type="Lambertian" name="gray">

<diffuseColor>0.5 0.5 0.5</diffuseColor>

</shader>

<surface type="Sphere">

<center>0 0 0</center>

<shader ref="gray"/>

</surface>

<surface type="Sphere">

<center>5 0 0</center>

<shader ref="gray"/>

</surface>

같은 쉐이더를 두 개의 구체에 적용합니다. 실제로 파일 형식은 매우 간단하며, 우리가 제공하는 예제에서는 원하는 모든 장면을 구성하는 데 어려움이 없을 것입니다.

1. 프레임워크

이 과제를 위한 프레임워크에는 간단한 메인 프로그램, 기본 이미지 처리 및 입력 파일 형식 파싱을 위한 몇 가지 예제가 포함됩니다.

4.1 파서

프레임워크는 Python의 내장 XML 파싱을 활용합니다. 파서는 단순히 XML 문서를 읽고 각 XML 엔티티에 대한 객체를 인스턴스화하여 해당하는 요소에 추가합니다.

예를 들어, 입력으로

<scene>

<surface type="Sphere">

<shader type="Lambertian">

<diffuseColor>0 0 1</diffuseColor>

</shader>

<center>1 2 3</center>

<radius>4</radius>

</surface>

</scene>

는 파서에서 자동으로 생성된 파스 트리를 결과로 반환합니다. 궁금한 사항에 대한 자세한 내용은 rayTracer.py에 있습니다.

이 모든 것의 실용적 결과는 레이 트레이서가 입력 파일의 요소와 일대일 대응하는 개체가 포함된 장면을 전달받는 것입니다. 이미 사용 가능한 정보만 사용하면 됩니다.

4.2. rayTracer.py

main 메소드는 프로그램의 진입점을 보유합니다. 이 메소드는 우리와 호환되는 명령 줄 인터페이스를 제공하기 위해 제공됩니다. 각 명령 줄 인수를 입력 파일의 이름으로 처리하고, 해당 파일을 구문 분석하고 이미지를 렌더링한 다음 이미지를 PNG 파일로 작성합니다. 실제 렌더링을 수행하기 위해 일부 추가 코드를 작성해야 합니다.

4.3 이미지

프레임워크는 부동 소수점 배열을 만들고 픽셀을 설정하고 이미지를 PNG 파일로 출력하는 필요한 코드를 제공합니다.

4.4 컬러 클래스

프레임워크에는 RGB 색상을 나타내는 클래스가 포함되어 있습니다.

감마 보정을 지원합니다.

1. 제출 및 FAQ

5.1. 스켈레톤 코드 실행 방법

필요한 패키지 (python3, pil 및 numpy)를 설치한 후 zip 파일을 압축 해제하십시오.

그런 다음 실행하십시오.

python3 rayTracer.py scenes/one-sphere.xml

이것은 ".png" 확장자가 있는 동일한 이름의 PNG 파일에 렌더링됩니다.

예상 출력 파일은 이미 폴더에 있으며, 위의 명령은 scenes/one-sphere.xml.png 파일을 대부분 비어있는 이미지로 덮어쓸 것입니다.

5.2. 공지사항

제출은 코스 git 저장소를 통해 이루어집니다. 새로운 질문 및 그에 대한 답변이 코스 스태프의 주의를 끌면 코스 웹 사이트에 FAQ 페이지가 업로드됩니다.