실시간 카메라 – 내비게이션 및 폐색
(Real-Time Cameras - Navigation and Occlusion)

Mark Haigh-Hutchinson

2009. 7. 1

http://www.gamasutra.com/view/feature/4065/realtime_cameras__navigation_and_.php

[Gamasutra 는 Mark Haigh-Hutchinson 의 Real-Time Cameras 를 발췌한 내용을 싣게 된 것을 자랑스럽게 생각한다. 그는 Retro Studios 의 직원으로 2008 년 췌장암 진단을 받은 후 2009 년에 사망하였다. 이 책은 작가가 사망한 후 동료 및 친구들의 협조로 출판되었고 현재 입수 가능하다. 책 작성에 관한 자세한 내용은 이 블로그 포스트를 참고하라.]

A.I. 게임 오브젝트로서의 카메라

인공 지능(AI) 영역 내에서는 카메라 내비게이션으로 인해 여러 가지 문제가 발생하고 있다. 이 책의 초반부에서 제시한 바와 같이 카메라는 게임 세계 내에서 위치와 방향을 결정할 경우 AI 캐릭터의 일부로 간주될 수 있다.

카메라는 물리적인 시뮬레이션의 영향 하에 있거나 표면에서 이동하도록 고정되어 있기 보다는 부동 상태(floating)에 있기 때문에 다른 AI 캐릭터 보다 자유롭게 이동할 수 있다. 이러한 이동의 자유는 실제 카메라에 비해 가상 카메라가 갖고 있는 이점 중의 하나이다.

카메라는 AI 캐릭터가 사용하는 것과 유사한 방식으로 경로 찾기를 결정할 수 있으나 대상 오브젝트 프레이밍, 대상 오브젝트 및 렌더링 형상 회피 등 미적인 문제와 관련된 추가적인 제약사항이 존재한다.

카메라에 적용할 수 있는 AI 주행 연구(로봇공학)에 대한 흥미로운 분야가 존재하는데, 특히 동적 게임 이벤트에 응답하는 카메라 모션을 원하는 경우 이동 결정에 대한 자동 메서드 연구가 필요하다. 이 발췌문의 끝부분에 있는 참고사항에는 Borenstein 와 같이 관련된 사례가 포함되어 있다.

다양한 환경 유형이 존재하므로, 카메라는 추가적인 독립 오브젝트와 함께 다각형 메쉬와 같은 충돌면 표현(다각형 메쉬)으로 이루어진 폐쇄된 환경에 구속되어 있는 것으로 가정한다. 여기서 폐쇄된이란 환경 표현은 모든 게임 오브젝트를 구속하는 연속면을 형성한다는 것을 의미한다(그렇다 하더라도, 카메라는 환경의 외부에 위치해야 하는 경우가 존재할 수 있다).

퍼페이스(per-face) 또는 서브 서페이스(sub-surface) 기반에 저장된 표면 재료에 관한 추가 정보를 바탕으로 렌더링 세계의 충돌면은 (실제로는 동일하지만) 별개의 엔터티로 간주한다. 또한 충돌 지점과 이 지점에서 발견된 재료에 관한 결과를 반환하는 환경에서 레이 캐스팅하기 위해 게임 엔진 내에 메커니즘이 존재한다고 가정한다.

또한, 특정 레이 캐스팅에서 감지할 수 있는 재료 유형을 필터링해야 한다(카메라는 일부 표면을 통해 게임 오브젝트를 통과시킴). 이 책에서는 충돌 데이터의 최적화와 구성은 다루지 않으며, 충돌 시스템 설계와 구현에 대한 자세한 내용은 VandenBergen 및 Ericson 을 참고하도록 한다. 9 장은 카메라 충돌 감지 및 회피에 사용된 주요 메서드의 일부를 기술한 것이다.

2 장에 기술된 바와 같이, 예측 카메라는 플레이어 캐릭터를 폐색하는 환경 변화를 예상할 수 있으므로 반응 카메라 보다 향상된 내비게이션 결과를 제공한다. 물론, 이러한 경우 추가적인 성능 비용을 감안해야 한다.

내비게이션 기법

내비게이션은 이동 경로 또는 잠재적인 폐색에 관련된 환경 데이터를 카메라가 요청하는 방식과 관련되어 있다. 게임 환경의 특성, 데이터 표현 및 게임 엔진이 제공하는 설비는 내비게이션 기법에 상당히 영향을 미친다. 이러한 내비게이션 기법은 동적 및 사전 정의된 내비게이션 기법 등 두 가지로 분류해 볼 수 있다.

동적 내비게이션 기법

동적 내비게이션은 카메라 이동과 위치에 관한 사전 정의된 정보에 의존하지 않는 기법이다. 그 대신, 퍼 프레임(per-frame) 기반에 카메라 위치를 지정하는 방법을 다룬다. 이러한 방식은 계산상 비용이 많이 들므로, 여러 기법에서 프로세서 비용을 상각해야 한다.

동적 솔루션을 사용할 때 고려해야 할 사항은 플레이어가 수행할 수 있는 무수한 동작과 위치를 설명하기가 어렵다는 점이다. 즉, 선택한 내비게이션 솔루션에 대해 문제가 될 수

있는 특정 사례가 존재하지 않을 뿐만 아니라 순열을 테스트하는 것이 어렵다는 것을 의미한다. 좀 더 일반적인 동적 내비게이션 기법을 살펴 보도록 하자.

레이 캐스팅

가장 간단하고 일반적인 내비게이션 방법 중의 하나는 레이 캐스팅으로, 가상 환경을 통해 직선을 정확하게 투영하는 것이다. 삼각법을 이용하여, 직선이 환경과 관련된 충돌 형상이나 환경 내의 오브젝트와 교차하는지 여부를 결정한다. 예상한 바와 같이 특히, 복잡한 환경인 경우 집중적인 계산이 필요할 수도 있다.

카메라 시스템을 구현할 때 주요 목표는 가능한 경우 테스트 분량을 제한하는 것이다. 레이 캐스팅 상각이 가능하며 내비게이션에 필요한 정보량 제한 비용에 맞추어 프로세서 요구사항을 줄이려고 하는 경우 효과적이다.

내비게이션의 경우 충돌 또는 직접적인 관계 보다 해당 결정이 오래 지속되고 평가하는데 많은 시간이 걸리는 정보에 대한 퍼 프레임(per-frame) 업데이팅이 필요하지 않는다. 또한 공간 분할 또는 재료 필터링을 이용하여 비교할 데이터 집합을 줄이는 것이 좋다.

레이 캐스팅은 카메라 시스템과 관련된 정보, 예를 들어 충돌 예상 및 플레이어 캐릭터 폐색 등을 제공할 수 있으며, 폐색에 대한 상세 뷰를 생성하기 위해 결합할 수도 있다. 또한 레이 캐스팅은 게임 세계를 통한 3D 라인의 투영과 관련되어 있으므로, 빛(ray)은 작은 틈을 통과할 수 있고 충돌을 기록하지 않는다.

대개 이러한 문제는, 잠재적인 성능 비용을 고려해야 하나 대규모 영역을 다루기 위해 레이 캐스팅 위치와 회수를 증가시켜 줄일 수 있으며, 카메라 이동과 관련되어 있고 작은 틈을 포함하지 않는 별도의 충돌 메쉬를 통해 회피할 수 있다.

대안적인 해결책은 레이 캐스팅 이력현상 즉, 다수의 업데이트를 통해 수집한 레이 캐스팅 결과에 대한 통계 정보를 적용하는 것이다. 이러한 히스토리로부터 빛(ray) 충돌에 대한 확률 그래프와 업데이트 당 레이 캐스트의 실제 양을 축소할 수 있다. 확률 그래프를 사용하여 카메라 모션에 적용된 영향을 결정한다. 예를 들어 카메라에 관한 레이 캐스팅 위치에 따라, 빛(ray)이 대상에 도달하지 못하도록 차단된 경우 카메라의 가로 및 세로 모션은 향후 충돌을 회피하는데 필수적이다.

레이 캐스팅 결과를 이용하여 잠재적인 카메라 모션을 다양한 방식으로 지정할 수 있다. 첫째, 레이 캐스팅의 성공 여부에 따라 각 레이 캐스팅의 가중치 또는 영향 요소를 적용할 수 있다. 빛(ray)이 대상에 도달하지 못하는 경우 해당 위치에서 멀리 떨어진 방향으로 카메라가 이동한다.

가끔 사용되는 대안적인 방법은 카메라 뒤쪽의 대상 오브젝트에서 레이 캐스팅하는 것이다. 이러한 레이 캐스팅 결과를 유사한 방식으로 활용한다. 다시 한번 빛(ray)의 길이가 짧아지는 경우 카메라의 움직임에 영향을 미치게 된다. 이러한 영향의 정도는 레이 캐스팅의 원점에 대한 충돌 지점의 근접 여부 또는 특정 레이 캐스팅 방향을 바탕으로 사전 정의된 가중치 요소에 따라 달라진다.

레이 캐스팅 관련 문제 중의 하나는 빛(ray)이 환경 또는 다른 게임 오브젝트와 교차하는지 여부를 결정하려면 계산상 비용이 많이 든다는 것이다. 또한 속성 정보를 충돌면에 적용해야 한다.

충돌 또는 빛(ray) 교차가 실제로 유효한 경우 이 정보를 사용할 수 있다. 일반적으로, 해당 필터가 적절하게 설정되어 있는 경우 게임 카메라는 환경의 주요 부분을 회피할 수 있다. 이러한 작업은 표면에 적용된 속성 집합의 적절한 속성에 대한 간단한 마스킹 연산으로 수행한다.

기술 측면

간단한 레이 캐스팅을 구현하는 경우 대상 오브젝트로 확장된 빛(ray)과 함께 카메라의 제안된 위치 주변에 광원 위치를 고정된 형태로 배열한다. 이러한 배열이 게임 세계의 형상과 교차하지 않도록 주의해야 한다.

가끔 빛(ray) 위치를 그리드 또는 대칭형으로 배열한다. 각 빛(ray)은 수평, 수직, 또는 수평과 수직으로 적용할 수 있는 영향 요소에 제공된다. 대상 오브젝트의 다음 위치에 레이 캐스팅을 적용하지 못하는 경우 그 결과, 해당 카메라 위치에 영향을 미치게 되고, 이러한 영향력을 대상의 위치에 따라 계산할 수 있다.

```
모든 레이 캐스팅은
  레이 캐스팅이 성공적인 경우
      어떤 영향력도 적용되지 않고
  레이 캐스팅이 실패한 경우
      대상의 레이 캐스팅 충돌에 대한 거리로 영향 요소의 크기를 조정하고
      영향력을 적절한 카메라 위치에 추가하여
종료한다.
```

일단 전체 영향력이 산출되면 카메라는 원하는 위치로 이동한다. 필요한 경우 영향력을 각 축에서 다른 방식으로 계산할 수 있다.

볼륨 투영

이 기법에서는 카메라 위치에서 대상 위치까지 게임 세계를 통해 볼륨을 투영한다. 대개 이러한 볼륨은 단순성과 성능을 위해 직사각형 또는 원통형의 돌출된 구(때때로 캡슐이라고도 함) 형태를 띤다. 볼륨 경계 주변에서 레이 캐스팅으로 볼륨을 자극하여 성능을 개선할 수 있다.

충돌 결정 기법과 함께 잠재적인 충돌 집합의 형상 또는 중요하지 않은 오브젝트를 필터링해야 한다. 축-정렬 경계 상자 또는 유사한 기법에 대한 경계 검사를 컬링 오브젝트(cull object)에 적용할 수 있다.

볼륨 투영은 오브젝트와 관련된 새로운 위치를 결정할 때 또는 카메라가 밀폐 공간에 적합한지 여부를 결정하는 경우 사용한다(예: 장면의 급전 후).사용된다. 또한 현재 위치에서 원하는 위치까지 모션 볼륨을 투영하여 카메라 모션을 효과적으로 캡처하기 위해 사용할 수 있다.

구/원통 충돌

카메라는 충돌 용도로 구를 고려하는데, 이렇게 하는 이유는 구 충돌이 상대적으로 간단하고 구는 충돌면을 따라 미끄러져 내려가는 경향이 있기 때문이다. 충돌 구의 반지름은 카메라에서 뷰 절두체의 근평면 거리 보다 약간 커야 하고 렌더링 형상과 함께 카메라의 상호 관통을 방지해야 한다.

그러나 이것은 오브젝트의 충돌 형상 또는 환경이 렌더링 형상의 범위를 넘어서 확장되는 경우에만 가능하다. 근평면 거리는 뷰의 수평 필드에 따라 달라짐에 유의해야 한다. 충돌 볼륨에는 근평면의 절두체 정점 4 개가 포함되어 있다.

카메라 충돌 볼륨은 수직 축 정렬 원통으로 간주할 수 있다. 카메라가 바닥 또는 천정과 같은 수평면 근처로 이동하지 않는 환경에서 이러한 근접성은 축소된 성능 비용으로 구에 대한 동일한 기능을 제공해야 한다. 근평면 거리와 관련하여 동일한 경고를 적용하고 수직 충돌은 카메라 높이를 고려해야 하며 이를 통해 형상이 근평면을 상호관통하는 방법이 변경된다.

1 인칭 카메라의 경우, 카메라가 플레이어 캐릭터의 충돌 경계 내에 완전히 포함되어야 하므로 충돌 볼륨 또는 테스팅이 필요하지 않을 수 있다(카메라 이동이 차단되어 있고 플레이어가 존재하지 않는 경우 다소 당황하게 된다).

그러나 3 인칭 카메라에서 흔히 볼 수 있는 동일한 방식의 렌더링 아티팩트를 피하기 위해 플레이어의 충돌 경계는 근평면 거리 보다 커야 한다. 플레이어에 대한 충돌을 테스트하지 않거나 충돌 볼륨으로부터 돌출된 렌더링 형상을 갖고 있는 게임 AI 오브젝트와 같은 게임 요소가 존재하는 경우, 근평면의 상호관통은 가능하다.

동적 경로 찾기

실시간 또는 동적 경로 찾기는 게임 세계의 AI 오브젝트 내비게이션을 위해 사용된 솔루션과 유사점을 갖고 있다. 그러나 카메라의 경로 찾기는 AI 경로 찾기 보다 규칙이 엄격하다. 기본적인 아이디어는 게임 세계의 모션에 유용한 공간 볼륨을 정의한 사전 계산된 정보를 사용할 수 있다는 것이다. 이러한 볼륨을 나타내기 위해 사용된 데이터 구조는 다양하지만 연결성 정보를 통해 볼륨을 탐색하는 메서드를 포함하고 있다.

최적의 특정 경로를 지정하는 가중치가 주어진 경우, 볼륨을 통한 가장 효율적인 경로를 결정하기 위해 A* 또는 Dijkstra 알고리즘(다양한 AI 또는 전산학 텍스트북에 제시되어 있음)과 같은 기존 검색 알고리즘을 사용할 수 있다. 최적의 카메라 모션을 결정하는 것은 AI 오브젝트와는 상당히 다를 것으로 생각된다.

일단 경로가 생성되면 카메라는 원하는 경로에 도달하거나 새로운 요청으로 이동이 중단될 때까지 이 경로를 따른다. 정기적으로 경로를 다시 계산하는 경우, 플레이어 모션 또는 환경 변화를 설명하기 위해 이러한 상황이 발생한다. 경로 검색은 두 가지 수준에서 이루어진다. 고수준 솔루션은 볼륨 간의 카메라 모션을 결정하고 저수준 솔루션은 충돌 형상에 근접해 있는 게임 오브젝트를 회피한다.

이러한 솔루션과 AI 용으로 사용된 것 간의 차이점은 후자는 표면을 따라 긴 모션 경로(나르거나 점프하는 등)를 다루는 경향이 있는 반면, 카메라는 상대적으로 대상 위치와 가까운 곳에 있다는 점이다. 카메라의 경우, 경로 정보는 모션 경로 보다는 충돌 회피를 위해 사용된다. 비대화형 영화 시퀀스에서 사용되는 것과 같은 긴 모션 경로는 대개 사전 정의되어 있으며 모든 유형의 내비게이션 로직을 적용할 필요가 없다.

동적 경로 생성

게임 세계에 구축된 사전 결정된 정보에 따라 새로운 경로의 형태와 길이가 결정되지 않으므로, 경로 생성은 앞에서 언급한 경로 찾기와는 다르다고 할 수 있다. 게임 세계에 관한 정보는 현재 상황에 따라 동적으로 결정되고 필요한 경우 가장 적절한 경로를 생성하기 위해 사용된다. 또한 이러한 경로는 시간이 지남에 따라 변경되고 불필요한 카메라 이동을 유발할 수 있다. 동적 경로는 표준 내비게이션으로 적절한 이동을 결정할 수 없는 경우 사용된다.

동적 경로 생성을 위한 또 다른 기법은 플레이어 캐릭터의 모션으로부터 확보한 정보를 사용하는 것이다. 다양한 상황에서 플레이어 이동 데이터의 히스토리를 이용하여 카메라의 잠재적인 이동 경로를 도출해낸다. 플레이어가 밀폐 공간(예: 출입구)을 통과한다는 사실을 알고 있는 경우 카메라는 동일한 경로를 명확하게 따라갈 수 있다.

그러나 이러한 경로는 보통 볼 수 있는 것과는 매우 다른 플레이어에 대한 뷰를 제공한다. 따라서 유효한 카메라 모션의 잠재적 제한점을 발견하기 위해 플레이어의 이동 지점 주변에서 볼륨을 생성해야 한다. 이 볼륨을 사용하여 계속적으로 카메라를 플레이어 캐릭터와 교체하면서 뷰를 표시한다.

가시성과 렌더링 솔루션

하드웨어 지원 렌더링과 함께 원하는 카메라의 위치 결정을 위해 다양한 솔루션을 사용할 수 있다. 최종 위치가 카메라의 대상 오브젝트와 관련된 어딘가이고 원하는 위치에서 대상 오브젝트를 장애물 없이 환히 볼 수 있어야 한다.

다음으로, 대상 오브젝트와 관련된 원하는 위치의 거리 및 높이에 대한 제약 조건을 추가한다. 이를 통해 대상 오브젝트 주변에서 잠재적인 위치의 원통형 밴드가 생성된다. 대상 오브젝트와 관련된 각 변위를 허용하는 것과 같은 요구사항과 함께 제약 조건을 계층화할 수 있다.

이러한 제약조건이 주어진 경우, 잠재적인 위치의 소규모 하위집합에 대한 실행가능성을 테스트한다. 카메라가 대상 오브젝트를 향하고 있는 잠재적인 위치 각각에서 게임 세계를 렌더링하는 경우, 게임 오브젝트 또는 형상 폐색에 따라 대상 오브젝트를 보여 주거나 보여 줄 수 없는 장면이 존재하게 된다.

대상 오브젝트를 구별하기 쉬운 색상으로 묘사하기 위해 뷰를 렌더링하는 경우, 대상 오브젝트가 폐색되었는지 여부를 결정하기 위해 렌더링한 전면 버퍼를 분석할 수 있다. 오브젝트가 충분히 폐색되지 않은 경우 렌더링한 카메라 위치를 실행가능한 위치로 간주한다. 다른 하드웨어 가속 기법을 사용하여 각 잠재적인 위치에서 대상 오브젝트의 폐색을 결정할 수 있다(깊이 버퍼 분석 포함).

그래픽 하드웨어의 병렬 특성을 활용하여 이러한 결정에 대한 비용을 상각할 수 있다. 이러한 접근법에서는 주의할 점이 몇 가지 있다. 첫째, 각 잠재적인 위치에서 장면 렌더링을 설정하지 못할 수 있다.

둘째, 하드웨어를 통한 렌더링이 빠르지만 자유롭게 진행하기는 어렵다. 평면 형태로 음영처리하고 렌더링하는 기법을 이용하여 GPU 및 CPU 비용을 절감하고, 장면을 대상 오브젝트 주변 영역으로 한정하기 위해 뷰 절두체를 변경하여 렌더링한 장면의 해상도를 줄일 수 있다.

원래 잠재적인 카메라 위치 중 어느 위치를 가장 선호하는 지를 결정하면, 현재 위치에서 원하는 위치로 카메라를 옮겨가기 위해 경로 또는 이동 방법을 계산할 수 있다. 이러한 기법은 위치 결정 시 유용하나, 현재 위치와 원하는 위치 간의 형상 연계를 고려하고 있지

않다. 추가 제약조건을 적용하여 이러한 문제를 방지할 수 있다. 이러한 기법의 실제 구현은 Halper 에서 참고하도록 한다.

콜라이더(Collider)

콜라이더는 앞에서 언급한 바와 같이 표준 레이캐스팅 기법의 변형된 형태이다. 이 기법의 원리는 레이 캐스팅이 이루어진 위치는 변경될 수 있다는 것에 바탕을 두고 있다. 이러한 위치는 작은 충돌 볼륨 내에 둘러 싸여 있다(대개 구 또는 거의 구에 가까운 형태).

이러한 위치 오브젝트는 환경 내에 머물러 있도록 구속되어 있으므로 이를 콜라이더라고 한다. 콜라이더는 카메라 주변에 원 또는 반원 등 특정 패턴으로 배열된다. 콜라이더는 카메라가 환경으로 이동함에 따라 카메라의 최적의 위치와 관련된 원하는 위치를 다시 확보하려고 한다.

콜라이더의 배열은 카메라가 환경에 반응하는 방식에 영향을 미친다. 각 콜라이더는 카메라가 주시하는 위치 또는 플레이어 캐릭터에 대한 사전 정의된 지점을 레이 캐스팅한다. 레이 캐스팅 결과를 이용하여 카메라 모션의 원하는 위치를 오프셋할 수 있다. 다음과 같이 콜라이더가 미칠 수 있는 영향의 양은 요소의 수에 따라 달라질 수 있다.

- 로컬 카메라 공간으로 계산하고 동일한 방식으로 적용한 카메라의 콜라이더에 대한 물리적인 거리.
- 전체 그룹 내 특정 콜라이더의 위치에 따라 결정된 콜라이더 당 사전 정의된 영향 요소.
- 각 콜라이더에서 형상을 레이 캐스팅하는 지점까지의 거리.
- 원하는 동작에 따른(예: 플레이어의 속도에 따라 높이 변경) 한 축만의 플레이어 캐릭터의 가속도 또는 속도.

이상의 요소의 조합을 일부 또는 모든 콜라이더에 적용할 수 있다. 다양한 요소에 의해 영향을 받는 몇몇 그룹의 콜라이더가 존재한다. 영향이 적용되는 방법은 다양하며 가중 평균 또는 중심 계산을 이용한다. 특정 게임에서 접하게 되는 환경 유형에 따라 일부 실험이 필요하다.

환경에 필요한 동작 유형에 따라 콜라이더를 다양하게 배열하는 방법을 적용할 수 있다. 원형 또는 반원 배열은 수평 배열과 마찬가지로 일반적으로 사용되는 형태이다. 이러한 배열은 플레이어 캐릭터의 변경사항과 속도에 따라 달라진다.

콜라이더와 다른 레이 캐스팅 기법은 레이 캐스팅(카메라 모션에 나쁜 영향을 미쳐서는 안되는 부수적인 환경 기능과 충돌함)으로 생성된 “노이즈(noise)”의 영향을 받기가 쉽다. 환경에 대한 일부 지식을 사용하여 카메라 모션에 영향을 미쳐서는 안되는 결과를 필터링할

수 있다. 표면 속성에 대한 태깅을 사용하여 카메라 위치 결정에 대한 영향을 축소하거나 배제할 수 있다.

또는 표면을 태깅하여 모션 제약조건을 적용하거나 카메라 접근을 차단할 수 있다. 콜라이더의 배열과 영향의 양은 환경 속성(예: 밀폐 공간, 수직 열) 또는 게임 플레이 모드(예: 플레이어 캐릭터의 변경된 이동 특성에 맞춤)에 따라 달라질 수 있다.

기술 측면

콜라이더의 배열이 상당히 다양하므로 여기에서는 원하는 카메라 위치에 관한 영향을 결정하는데 사용된 알고리즘의 개괄적인 내용만 다루도록 한다.

```
stl::vector influence;
// might be useful as a class
for (int i = 0; i < colliders.size(); ++i)
{
influence.push_back(colliders[i].offset *
    colliders[i].GetWeighting());
// the weighting depends on line of sight and/or
// other factors such as the relative collider
// position
}
// will need to get an average or similar
return GetCentroid(influence);
```

콜라이더 배열에 따라 영향의 평균량을 결정하는데 사용된 계산 방법이 달라진다. 콜라이더가 평면에 구속되어 있는 경우 중심 계산이 효과적이나, 최종 영향을 계산하여 콜라이더의 오프셋과 동일한 공간에 적용해야 한다.

사전 정의된 내비게이션 기법

잠재적인 카메라 움직임을 사전에 계산하거나 환경 지식을 바탕으로 사전 정의된 지원을 이용하여 카메라 충돌과 관련된 다양한 문제를 방지할 수 있다(카메라 디자이너가 명시적으로 정의하거나 자동으로 사전에 계산하든지 여부와 관계 없음).

카메라 내비게이션의 사전 계산은 메모리 오버헤드와 허용 가능한 카메라 위치의 계산 시간에 대한 비용 측면에서 다양한 형태로 이루어진다. 대개 사전 계산은 실시간 동안 낮은 프로세서 오버헤드에 유용하다.

게임 플레이와 환경 관심도에 대한 특정 요구사항에 맞추어, 디자이너가 배치한 게임 스크립트 오브젝트의 형태로 내비게이션을 지원한다. 6 장에서는 스크립팅과 카메라 모션과 위치에 대한 적용을 다룬다. 카메라 동작을 제어하기 위해 특별히 설계하기 보다는 다른 게임 오브젝트로부터 내비게이션 지원 정보를 이끌어낼 수 있다.

두 가지 형태의 정보를 이용하여 카메라 내비게이션과 충돌 회피를 다양한 방식으로 지원한다. 이러한 지원 체계는 AI 또는 로봇 연구 기법과 유사한 측면이 있으며 참고문헌에는 관련된 논문 개요가 제시되어 있다.

일반적인 솔루션의 일부는 다음과 같다.

사전 정의된 경로

사전 계산에 대한 일반적인 솔루션은 전체 모션 경로를 저장하는 것이다. 경로를 저장하는 경우 카메라 위치와 방향을 완전히 제어할 수 있으므로 몇몇 게임에서는 이러한 방식을 효과적으로 활용하고 있다. 그럼에도 불구하고, 경로 모션은 제한적이며 잠재적으로 플레이어는 제어가 힘들어진다는 느낌을 가질 수 있다.

카메라 위치를 직접 지정하는 기능은, 특히 플레이어 위치 또는 동작에 따라 위치 결정이 달라질 수 있으며 복잡한 환경에서 모션을 처리해야 할 때 유용하다. 경로 기반 카메라 모션에 대한 전체 내용은 7 장 경로 절에 포함되어 있다.

경로 모션 동작

모션을 정의하기 위한 경로를 이용하여 카메라를 구속할 수 있으나 스플라인이 모션을 실제로 구속하는 방법에 대한 해석을 통해 몇 가지 대안을 고려해 볼 수 있다.

- **구속된 경로 카메라.** 구속된 경로 카메라는 게임 세계 내의 정의된 경로를 따라 정확하게 이동한다. 경로가 더 이상 활성화되지 않는 경우를 제외하고 카메라는 어떤 상황에서도 이 경로를 벗어날 수 없다. 이것은 경로 카메라의 전형적인 사용법 중의 하나로, 환경 기능과의 충돌 또는 상호 관통을 피하기 위해 카메라 위치를 완전히 제어할 수 있다. 비대화형 영화는 고정 또는 종속 위치 카메라뿐만 아니라 이러한 유형의 카메라로 제작된다.

- **추종 경로 카메라.** 많은 경우, 구속된 경로상의 카메라 모션이 원활하더라도 이 모션은 게임 디자이너가 요구한 대로 자연스럽거나 유기적으로 나타날 수 없다. 그러나 특정 방식으로 카메라 모션을 제한할 수는 있다. 추종 경로 카메라는 카메라 모션을 제한하면서 일부 변화를 가하는 메커니즘을 제공한다. 이러한 경우, 경로는 원하는 위치를 지정하고 카메라가 원하는 위치에

도달하는 방법을 다양한 방식으로 정의할 수 있다(예: 카메라는 경로에 직접 고정되어 있지 않음).

이러한 사항을 카메라 앞에 매달려있는 “당근”에 비유할 수 있다. 카메라는 경로에 대한 현재 평가로 정의된 위치를 탐색하나 해당 위치로 바로 이동할 수 있다(위치 결정 방법과 관계 없이 이러한 동일 기법을 적용할 수 있음).

또한 카메라가 이동할 수 있는 원하는 경로로부터 멀리 떨어진 거리를 제어하는 규칙을 이용할 수 있다(예를 들어, 단면은 원형이 아닌 타원형 또는 직사각형이나 경로는 원통관임). 경로를 따라 카메라 모션의 변동량을 고려하기 위해 경로 볼륨에 관한 형태 정보를 노트(knot)에 포함시킬 수 있다(원통관의 반경 포함).

또 다른 접근법은 경로를 따르는 카메라 위치와 같은 입력 변수와 함께 매개변수 곡선을 사용하는 것이다. 이러한 방법을 통해 다양한 경로 없이도 카메라를 로컬로 제어할 수 있다. 추종 경로와 떨어져 있는 카메라 모션을 제한하는 경우 불필요한 모션 아티팩트를 유발할 수 있으므로 고정 제한조건을 사용하지 않는 것이 좋다. 감쇠 스프링 또는 어트랙터/리펄서(attractor/repulsor)를 사용하여 거리 제한에 대한 아날로그 응답을 제공할 수 있다.

사전 정의된 볼륨

사전 정의된 볼륨 크기로 카메라를 구속하는 것과 같이 카메라 모션도 제한할 수 있다. 볼륨의 실제 정의는 가변적이며, 구, 원통, 타원체 또는 관 등 간단한 볼록 렌즈 형태로 이루어져 있다. 또한, 다른 게임 엔터티와 같이 복잡한 충돌 메시로부터 도출된다.

형태가 복잡한 경우 내비게이션 작업이 힘들 수 있다. 볼륨 내의 원하는 위치는 다양한 방식으로 정의할 수 있다. 볼륨 내의 추가 제약조건과 함께 플레이어 캐릭터에 해당되는 위치와 관련하여, 이 장의 앞부분에서 언급한 것과 동일한 기법을 사용할 수 있다.

초기 결정을 내린 후 사전 정의된 볼륨 제약조건을 적용하기 위해, 이러한 접근법을 대부분의 동적 결정 방법에 적용할 수 있다. 대개, 고정 차원의 볼륨을 사용하는 것이 더 쉬우나 게임 플레이 요구사항에 따라 달라질 수 있다(예: 참조 프레임에 관한 플레이어의 위치).

경로 행태와 동일한 방식으로 카메라 위치를 직접 구속하고 있는 경우, 불연속 모션과 힘 또는 감쇠 스프링 사용으로 인한 이점을 확보할 수 있으며 볼륨 내에 카메라가 유지될 수 있다.

어트랙터/리펄서(attractor/repulsor)

카메라를 사전 정의된 위치, 영역 또는 다른 게임 오브젝트로 밀거나 끌어당기는 어트랙터/리펄서를 이용하여 효과적인 내비게이션을 지원할 수 있다. 해당 위치는 카메라 디자이너가 배치한 스크립트 오브젝트를 이용하여 명시적으로 정의한다. 자동 배치를 결정하기 위해 환경에 대한 오프라인 분석을 수행할 수 있다. 즉, 사용을 관리하는 명확한 규칙 없이는 이러한 작업을 예측하기가 어렵다.

어트랙터/리펄서는 플레이어 또는 카메라의 근접성 및 방향에 따라 카메라에 힘을 가함으로써 제 기능을 한다.

이러한 힘은 물리 시뮬레이션의 일부로 구현할 수 없으며, 원하는 위치 결정에 영향을 미치거나 실제 카메라 모션이 진행되는 동안 엄격한 제약조건으로 작용할 수 있다.

근접성을 힘 계산의 일부로 사용하는 경우 어트랙터/리펄서는 사실상 구형이나 이러한 형태가 반드시 적절한 것은 아니다. 게임 세계의 상향 벡터에 맞추어 조정된 주요 축과 함께 원통은 가장 유용하게 사용할 수 있는 형태이다.

필요한 힘과 게임 오브젝트의 근접성에 대한 계산 시 상대적으로 낮은 비용 때문에 단순한 형태를 흔히 사용하고 있다. 즉, 구, 원통 및 평면은 일반적으로 선택하고 있는 형태이다.

카메라에 문제가 되는 경우는 캐릭터가 신속하게 이동할 때 출입구를 통해 내비게이션해야 할 때이다. 카메라와 대상 사이 또는 카메라 상단에서 문이 물리적으로 닫혀있거나, 출입구가 밀폐 공간이며 플레이어 캐릭터가 카메라를 뒤에 남겨두고 이동할 가능성이 높은 경우 출입구가 문제가 될 수 있다.

이러한 사례는 수직 출입구의 문이 게임 세계의 상향 축에 맞추어 정렬되어 있는 경우 흔히 발생한다. 특히, 수평 출입구는 짐벌 락으로 인한 카메라 방향과 관련하여 문제가 발생할 수도 있다. 또한 카메라를 닫힌 문 뒤에 남겨 둔 상태에서(또는 문 형상을 상호관통하거나) 외부 이벤트로 인해 문이 재빨리 닫힌 경우도 문제가 발생한다.

이러한 상황에 대처하기 위해 카메라가 플레이어의 반대편에 있는 경우 문을 열어 두는 것이 좋다. 따라서 카메라가 닫혀진 문의 잘못된 위치에 남겨지지 않도록 안전장치를 구현하도록 해야 한다.

카메라 시스템이 출입구에 대한 카메라의 근접성을 추적할 수 있거나 카메라가 출입구에 대해 어떤 위치와 방향을 취하고 있는 지를 알고 있는 경우(트리거 볼륨 이용), 축 기반 어트랙터를 사용하여 출입구를 통해 카메라 모션을 안내할 수 있다.

비례 어트랙터는 출입구의 원하는 모션 축과 관련된 카메라 위치에 따라 카메라에 힘을 가할 수 있다. 이러한 힘은 진동을 전달하지 않으며 해당되는 경우에만 적용됨에 유의해야 한다.

일반적인 경우를 고려해 볼 때, 어트랙터 또는 리펄서가 힘을 가하는 방향은 가변적일 수 있다. 힘은 어트랙터 또는 리펄서의 물리적인 형태와 관련하여 작용한다(예: 구형 어트랙터는 원래 위치로 끌어당김).

어트랙터 또는 리펄서는 게임 세계의 방향에 따라 힘을 가할 수 있다. 즉, 원통형 리펄서는 원통의 중심 축에 대한 플레이어 캐릭터의 근접성으로부터 힘을 결정할 수 있으나 고정된 게임세계의 방향에 따라 반발력을 가한다.

또한 어트랙터 및 리펄서는 다른 게임 오브젝트에 연결되거나 일부로 생성되며, 위치 및 효과에 따라 다양한 형태를 띤다.

힘 기반 솔루션처럼 이러한 오브젝트가 해당되는 경우에만 힘을 가할 수 있도록 해야 한다. Stout 는 이러한 내용을 상세히 검토하고 있다. 힘이 카메라에 악영향을 미치지 않도록 설정하기는 어렵다.

원활한 모션을 제공하기 위한 요구사항을 고려해야 하는 미적인 고려사항이 존재하므로, 카메라는 위치(방향) 결정 측면에서 AI 오브젝트와는 다르다. 카메라가 어트랙터를 우연히 지나치는 경우 반드시 영향을 받지는 않는다. 또한 카메라가 어트랙터로부터 멀리 떨어져 있는 경우 힘을 가할 수는 없다.

어트랙터 위치를 조심스럽게 선택하여, 복잡한 수동 스크립팅 카메라 솔루션 없이 카메라 모션이 형상을 회피하거나 밀폐 환경을 통과하도록 할 수 있다. 어트랙터 또는 리펄서를 게임 오브젝트(예: 문)와 연결하여 추가적인 스크립팅 없이 카메라 모션을 지원할 수 있다.

유동장

유동장은 2D 평면 또는 3D 볼륨 내의 그리드에서 일정 간격으로 떨어져 있는 벡터의 스파스 배열이다. 벡터의 간격은 반드시 일정하지는 않다. 카메라 또는 다른 오브젝트가 유동장을 통과하는 경우 카메라 가까이에 있는 벡터는 벡터 방향 쪽으로 힘을 가하여 모션에 영향을 준다. 힘의 강도는 평면 또는 다른 요소로부터 카메라까지의 거리에 따라 달라진다.

유동장이라는 명칭은 벡터 배열이 오브젝트 주변을 흐르는 물의 흐름 또는 동체와 유사하다는 사실에서 비롯된 것이다. 벡터 방향을 조정하여, 카메라가 동일한 방식으로 장애물 주변을 돌아다니게 할 수 있다.

경쟁 상태의 벡터가 상호간의 영향을 상쇄하거나 카메라가 장애물을 제거하지 못하는 경우, 벡터 방향으로 인해 카메라가 고정되어 있는 상황이 발생하지 않도록 해야 한다.

유동장 사용을 카메라 모션에 관한 추가적인 영향 요소(유일한 요소가 아닌)로 적용한다. 또한 추가적인 영향을 적용해야 하는 시기를 결정해야 한다. 예를 들어, 벡터는 단일 방향 또는 양방향 벡터일 수 있다.

카메라가 유동장 벡터 방향과는 다른 방향으로 유동장을 내비게이션할 수 있는 경우 카메라 모션이 벡터 방향과 평행한 경우에만 영향이 적용될 수 있다. 영향의 정도는 카메라의 속도에 비례한다.

영향력 분포도

영향력 분포도는 유동장과 유사한 또 다른 스파스 배열이다. 그러나 영향력 분포도에서 카메라 위치의 매핑은 카메라 자체 보다는 영향력 분포도에 대한 대상 오브젝트의 위치를 바탕으로 이루어진다. 영향력 분포도에 관한 각 위치는 카메라가 차지하고 있는 공간의 다른 위치와 동일하다.

현재 위치의 카메라 모션이 원활하게 유지되도록 하기 위해 영향력 분포도에서 도출된 원하는 위치를 보간으로 사용한다. 어떤 점에서는, 스플라인 모션을 영향력 분포도에 대한 단일 차원의 변형으로 간주할 수 있다. 이러한 경우, 스플라인 맵에 대한 대상 오브젝트 위치는 동일한(또는 다른) 스플라인에서 카메라가 차지해야 하는 위치보다 뒤쪽에 있다.

충돌 또는 가시성 정보를 통해 영향력 분포도를 오프라인으로 생성하거나 위치를 지정할 수 있다.

전위장

로봇 주행에 관한 연구로부터 비롯된 전위장은 동적 카메라 내비게이션 및 충돌 회피에 대한 유용한 접근법이다. 이러한 접근법은 이 책의 범위를 넘어서는 주제이기는 하나 그 내용을 간단히 살펴보도록 한다. 전위장은 정전기력의 원리에 바탕을 두고 있다. 즉, 오브젝트에 영향을 미치는 힘은 힘을 가하는 물체로부터 오브젝트까지의 거리에 비례한다.

활성화된 카메라가 충돌 또는 렌더링 형상에 접근하지 못하도록 하기 위해 이러한 힘을 가한다. 또한 힘의 구성요소를 카메라 모션에 추가하고 밀폐 공간의 내비게이션을 지원하며 카메라가 형상과 충돌할 때 유발되는 다양한 문제를 회피한다.

앞에서 언급한 바와 같이 어트랙터 및 리펄서와 유사한 방식으로 디자이너는 이 힘을 지점 또는 평면으로 정의하고 환경 내에 배치할 수 있다. 또한 해당 위치와 방향을 게임세계의 데이터 쿠킹(cooking)의 일부로 자동 생성할 수 있다.

유사하게, 충돌 형상을 사용하여 힘을 생성할 수 있다. 그러나 충돌 형상의 복잡한 특성으로 인해 카메라에 가장 가까운 힘의 결정 시 비용이 많이 들 수 있다. 한 가지 해결책은 음함수 곡면을 생성하는 것으로, 원활하고 연속적인 방식으로 충돌 형상의 근사치를 구하고 카메라의 가장 가까운 거리를 신속하게 결정할 수 있다. 음함수 곡면을 소개한 내용은 Bloomenthal 을 참고하면 된다.

로봇공학 연구에서는 이러한 기법을 다루고 있으며 다양한 결과가 산출되고 있다(예: Koren 참고). 환경의 복잡성(특히 장래성이 없는 것)과 관련된 주요 제한사항이 존재하며, 동시에 반대 방향에서 힘이 가해질 때 진동 문제가 발생할 수 있다(예: 좁은 복도).

게임 내에서 전위장을 실제로 응용하는 방법에 관한 자세한 내용은 Stout 를 참고하라. 전위장 막대그래프(VFH)를 이용한 대안 및 유용한 해결책은 Borenstein 에 제시되어 있다. VFH 는 적외선 또는 다른 센서로 결정한 오브젝트로부터 장애물까지의 거리에 관한 정보를 저장한다.

이 센서는 고정 각 오프셋에 장착되어 있거나 중점 주위를 회전하며 환경에 대해 효율적으로 레이 캐스팅한다. 시간이 지남에 따라 각 오프셋에서 반환된 거리 정보를 분석할 수 있다. 알려진 장애물에 대한 막대 그래프를 구성하고 잠재적인 충돌을 결정하기 위해 충분한 정보를 제공한다.

참고문헌

[Bloomenthal97] Bloomenthal, Jules (ed.). *Introduction to Implicit Surfaces.* Morgan Kaufmann Publishers, 1997.

[Borenstein90] Borenstein, J. Koren, Y. (1990). "Realtime Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments." *The 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, May 13-18, pp. 572-577.

[Ericson05] Ericson, Christer. *Real-Time Collision Detection.* Morgan Kaufmann Publishers, 2005.

[Halper01] Halper, Nicolas, Helbing, Ralf, Strothotte, Thomas. "A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence." *Proceedings of EUROGRAPHICS 2001 (Volume 20, Number 3).* Blackwell Publishers, 2001.

[Koren91] Koren, Y. Borenstein, J. (1991). "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation." *Proceedings of the IEEE Conference on Robotics and Automation*, Sacramento, California, April 7-12, pp. 1398-1404.

[Stout04] Stout, Bryan. "Artificial Potential Fields for Navigation and Animation." *Presentation at Game Developers Conference 2004*, San Jose, 2004. Written GDC 2004 proceedings.

[VandenBergen04] Van den Bergen, Gino. *Collision Detection in Interactive 3D Environments.* Morgan Kaufmann Publishers, 2004.