스크립트 프로그래밍 기반의 음향 생성 시스템 설계 및 구현

Design and Developing Aural Cueing System based on Script Programing

임승현*, 김귀하, 박태근(유니텍), 전향식, 전대근, 최형식(항공우주연구원)

1. 서 론

시뮬레이터는 시뮬레이션용 기계장치로써, 크게 풍동 실험이나 수조 실험과 같은 모형 중심과 물리 현상 및 움직임을 수학적 방정식으로 나타내는 전기현상으로 바꾸는 직접상사계산기로 구분할 수 있다. 이러한 시뮬레이터는 주로 연구용과 훈련용으로 주로 사용된다. [1]

훈련용 시뮬레이터는 항공기나 기타 장비의 조종훈련에서 실물을 사용하여 훈련하는 것이 경제상, 안전상, 실용상 곤란할 경우에 사용된다.

훈련용 시뮬레이터 개발에 필수적인 요소는 조종에 따라 실제 장비의 움직임을 모사하는 운동역학과 조정석 외부의 화면을 구성하는 영상 생성 시스템, 그리고 음향 생성 시스템이 있다.

그중, 음향 생성 시스템이란 실제 장비에서 나는 소리 및 소음을 실시간으로 사실감 있게 표현 하여 훈련생의 사실감 있는 경험을 위한 시스템이다.

음향 생성 시스템에서 생성하는 일반적인 소리들은 보통 엔진, 착륙 장치, 착륙 시 마찰음, 비행 시 공기 마찰음 등이 있으며 훈련생 사이의 사실감 있는 내부적인 통신 음향을 묘사할 수 있다.

음향 생성 시스템을 개발하는 고전적인 방법은 호스트 또는 UI에서 음향 생성 명령을 발생할 때 이를 처리하는 루틴을 해당 개발 코드에 직접 하드 코딩 하는 방법이다. 하지만 이 방법은 간단한 수정사항 발생 시 해당 코드를 수정하고 재 컴파일 하여 수정 사항을 즉각 반영하지 못한다는 단점이 있다.

본 논문에서는 이러한 문제점을 해결하기 위한 방안으로 실행 단계에서 수정 가능한 스크립트 프로그래밍 기법을 이용하여 음향의 생성이라는 기본적인 기능에 국한 하지 않고 특정 음원을 다양하게 생성 하는 시스템을 제시 한다.

2. 스크립트 프로그래밍

스크립트 프로그래밍은 컴퓨터의 기능을 사람이 손쉽게 사용할 수 있도록 하는 수단이다.

초기의 스크립트 언어는 일괄적인 명령 수행 도구로 쓰였다. 일련의 반복적인 명령들을 일종의 명령 처리기에 집어넣는 수단이었다. 초기의 스크립트 언어를 일괄 언어 또는 작업제어 언어라고 불렀던 이유도 거기에서 비롯된다.

스크립트 언어의 잘 알려진 예로는 마이크로소프트의 오래된 운영체제인 MS-DOS의 BAT 파일을 들 수 있다. 이 BAT 파일은 하나씩 차례로 수행될 일련의 DOS 명령들을 담은 텍스트 파일로, 언어 자체가 DOS 명령 집합이라 할 수 있다.

일반적으로 스크립팅 언어들은 빠른 개발을 위해 쓰이며 일반 사용자가 읽고 쓰기 쉬운 텍스트 기반 구문을 채용하는 경우가 많아서 일정 수준 이상의 사용자라면 개발자의 개입 없이도 스스로 스크립트를 작성하고 사용할 수 있다.

스크립팅 언어의 능력을 확장하기 위해 다른 어떤 저수준 언어로 만든 기능을 스크립팅 언어에 링크하여 개발자는 실행 환경을 더욱 세밀하게 제어하고 실행 결과를 즉시 확인할 수 있다. 또 개발 공정 도중 그 환경 안에서 소프트웨어를 좀 더 자유롭고 유연하게 수정, 실험해 볼 수 있게 된다. [2]

본 논문은 헬기 훈련용 시뮬레이터의 음향 생성 시스템 개발을 목적으로 작성 되었다.

헬기 훈련용 시뮬레이터에서 음향 생성 시스템은 음향을 생성하기 위해 기본적인 음원을 단순 재생을 하거나 특정 부분을 반복하여 루프 음향을 생성 한다.

하지만, 기본적인 음원의 자연스러운 루프 음향을 생성하기 위하여 반복 시작 트리거 부분과 끝 트리거 부분을 소스 코드에 매번 반영하여

컴파일 후 확인하는 작업에 걸리는 시간이 상당히 소요 되었다.

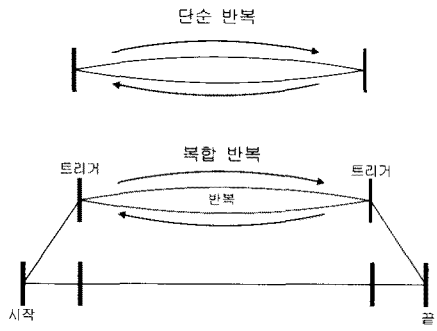

그림 1 루프 음향 생성 예

그림 1은 루프 음향 생성 예를 보여준다. 첫 번째 루프 음향은 단순하게 음원의 처음부터 끝까지 재생을 하고 다시 처음으로 돌아가서 재생을 한다.

두 번째 루프 음향은 시작 지점에서 두 번째 트리거를 만날 때 까지 기본적인 재생을 하다가 두 번째 트리거를 지나치는 순간 첫 번째 트리거 위치로 돌아가서 재생을 한다.

이러한 루프 음향의 트리거를 설정할 때 스크립트를 적용한 시스템과 적용하지 않은 시스템 사이에서 생성된 음향 확인의 효율성은 차이가 많이 난다.

본 논문에서는 이러한 스크립트 프로그래밍을 지원하기 위해 Lua 라는 스크립트 언어를 적용하였다.

Lua는 포르투갈어의 "달"에 해당 하는 말로, 1993년 브라질의 Pontifical Catholic 대학에서 탄생했다. Computer Graphics Technology Group (Tecgraf)의 한 팀이 Lua 언어를 개발하고 프리웨어로 공개했다.

Lua 개발팀의 목표중 하나는, C 프로그래밍 언어와 잘 맞는 효율적인 프로그래밍 언어를 만드는 것이었다. Lua는 스크립팅 언어들 중에서 가장 빠르고 효율적인 축에 속한다. [3]

Lua의 핵심 모듈은 크기가 120K 이하이다. 그래서 컴파일 하여 응용프로그램에 통합 했을 때의 용량 부담이 작다. 또, 대체로 Lua는 또 다른 스크립팅 언어인 파이썬(Python) 보다 훨씬 빠르다.

그림 2는 맥킨토시 G4 컴퓨터에서 프랙탈 계산을 동일한 알고리즘으로 수행결과의 속도를 비교한 그림이다. 그림에서 확인할 수 있듯이 Lua는 속도적인 측면에서 상당히 우월한 것을 알 수 있다. Emacs Lisp와 비교 할 때 24배 빠른 것을 확인할 수 있다.

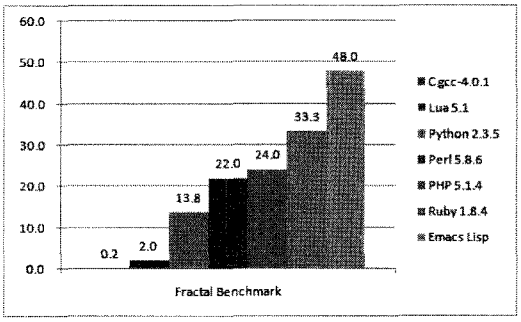

그림 2 Lua와 타 언어의 프랙탈 계산속도 비교

Lua는 C++ 같은 좀 더 강력한 저수준 언어와 함께 사용할 때 진정한 힘을 발휘하며 개발자는 응용 프로그램을 빠르게 프로토타이핑이 가능하다.

또한 응용 프로그램 자체를 빠르게 완성할 수 있으며 특정 개발자의 개입 없이도 특정한 기능 전체를 만드는 것이 가능하다.

Lua는 그 자체로 하나의 완전한 언어일 뿐만 아니라 다른 언어로 개발된 응용 프로그램과 자료를 교환할 수 있는 API(Application Programming Interface)까지 제공한다. Lua안에서 호출할 수 있는 C++ 함수를 만드는 식으로 루아를 확장하는 것도 가능하다.

응용 프로그램 개발 언어와 함께 쓰일 때, Lua는 고유한 프로젝트 언어를 구축하기 위한 하나의 틀 역할을 할 수 있다. 기본적인 명령 줄 환경에서 실행되는 독립적인 언어로서의 Lua는 상당히 제한적이며, 오직 학습 도구로 유용할 뿐이다.

이러한 Lua를 이용한 통합 구현은 저수준 언어와의 의사소통을 위해 함수 몇 개를 준비하는 정도로 단순할 수도 있고, 사용자가 만든 함수들에 근거한 새로운 언어를 만들어 낼 정도로 복잡할 수도 있다. [4]

3. 설계 및 구현

본 논문에서 제시하는 시스템을 구현하기 위해 3가지의 실행 객체로 구현 되었다.

첫 번째는 음향을 재생하는 Play 객체, 두 번째는 Play 객체들을 관리하는 Session 객체, 마지막으로 Play 객체에 다양한 효과를 적용하는 Filter 객체이다.

각 객체는 응용프로그램 안에서 실제 Class로 구현이 되어 있으며 이는 Lua 스크립트와 연동된다.

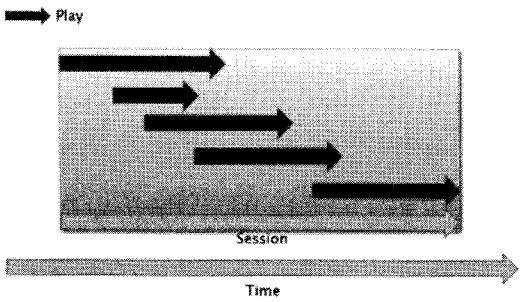

그림 3 세션 객체

그림 3은 Session 객체의 개념을 도식화 한 것으로 여러 Play 객체가 재생되는 시점과 길이가 다르지만 마지막 Play 객체가 재생을 마칠 때 Session 객체가 종료 되는 것을 확인할 수 있다.

이런 Session 객체는 시뮬레이터 호스트나 응용프로그램 UI에서 명령이 생성될 때 스크립트에서 Play 객체에 지정한 음원을 재생하거나 또 다른 Play 객체를 합성 하여 음향을 생성하며, Filter 객체를 적용함에 따라 스크립트에서 지정한 효과를 적용한다.

```
1  -- TEST_PLAY_START
2  tmpSession = Mother:GetSession();
3  if tmpSession == nil then
4   tmpSession = Mother:CreateSession();
5  end
6  tmpPlay = tmpSession:GetPlay("A");
7  if tmpSession ~= nil and tmpPlay == nil then
8   tmpPlay = tmpSession:CreatePlay("A");
9  end
10 if tmpPlay ~= nil then
11  tmpPlay.Loop = true
12 tmpPlay:Play();
13 end
```

그림 4 Lua 스크립트 예

그림 4는 Lua 스크립트의 예로써 A라는 음향을 재생하는 기능을 수행 한다.

시뮬레이터 호스트에서 TEST_PLAY_START라는 명령을 내리면 해당 스크립트가 실행된다.

명령과 동일한 Session 객체가 있는지 검색 하고 세션이 없으면 생성 한다. 생성한 Session 객체에서 "A"라는 음원을 지정한 Play 객체가 있는지 검색 하고 없으면 생성 한다. 그리고 Play 객체에 반복 설정을 하고 재생 명령을 내린다.

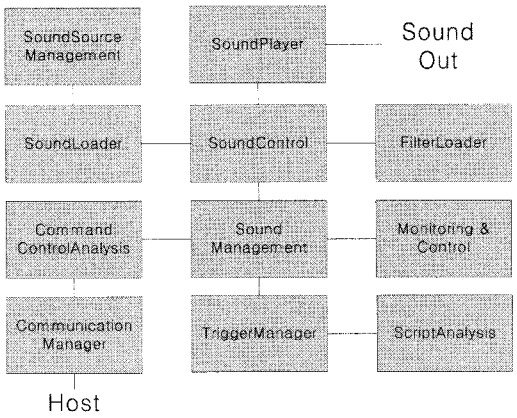

그림 5 음향 생성 시스템 블록 다이어그램

그림 5는 본 논문에서 제안하는 전체 시스템 블록 다이어그램이다.

표 1은 각 클래스의 기능을 정리한 것이다.

클래스	기능
Script Analysis	각 명령에 해당하는 스크립트를 분석하여 Lua State에 적용
Trigger Manager	각 명령에 해당하는 Trigger를 관리
Communication Manager	Host에서 생성하는 명령을 전달받는다. 이때 Tcp나 Udp, 혹은 Reflect Memory와 같은 다양한 연결 방식을 지원
Command Control Analysis	명령을 분석 하고 미리 정의된 명령 클래스를 생성하여 전달
Sound Management	Trigger Manager, Command Control Analysis 에서 전달한 명령을 관리
Monitoring & Control	Trigger Manager, Command Control Analysis, Sound Control에서 발생하는 로그를 관리
Sound Control	Sound Management에서 전달하는 명령으로 Sound Player를 제어
Sound Loader	Sound Player에 적용하는 음원을 메인 메모리에 로드
Sound Source Management	시스템에서 사용하는 음원을 각 ID별로 관리
Filter Loader	음원 재생에 적용하는 각종 필터를 관리
Sound Player	적용되는 음원을 재생

표 2 클래스 기능정리

음향 생성 시스템의 동작 과정은 다음과 같다.

시스템이 가동되면 Script Analysis 클래스에서 명령에 해당하는 각각의 스크립트를 분석하여 Trigger Manager 클래스에 Trigger를 정의 한

다.

Trigger 는 각 스크립트에서 정의된 Session 객체나 Play 객체의 실행이나 상태 조건을 정의한 것이며 해당하는 명령을 실행 한다.

Sound Loader 클래스는 Sound Source Management에 저장된 음원들을 메모리에 적재하고 고유 ID로 관리를 한다. Sound Source Management 클래스는 UI에서 입력하는 음원 파일을 등록하고 관리한다.

Communication Manager 클래스는 호스트와 다양한 연결 방법으로 통신을 하며 호스트에서 생성한 명령을 전달 받아 Command Control Analysis 클래스로 전달한다. Command Control Analysis 클래스는 해당 명령을 분석하여 Sound Management 클래스로 전달한다.

Sound Management 클래스는 Trigger Manager 또는 Command Control Analysis 클래스에서 전달하는 명령을 관리 하며 실행 시에 로그를 Monitoring & Control 클래스에 전달한다.

Filter Loader 클래스는 Sound Control 클래스에서 요청하는 Filter 객체를 관리한다.

Sound Control 클래스는 Sound Player 클래스를 관리한다. Sound Management 클래스에서 전달하는 명령을 통해 각종 음원이나 필터를 Sound Player 클래스에 적용 하여 음향 생성에 관련된 속성 값을 조절 한다.

그림 6은 본 논문에서 제안하는 음향 생성 시스템의 실행 화면이다.

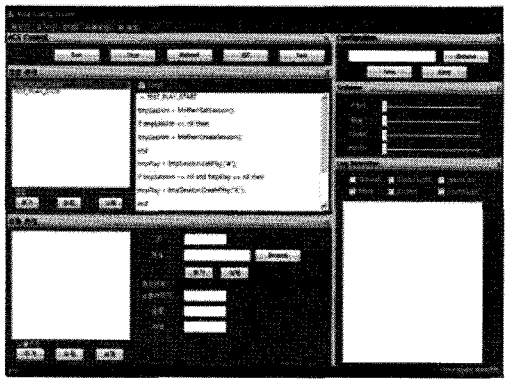

그림 6 음향 생성 시스템 실행 화면

그림 7은 Lua 코드와 내장 코드와의 수행 속도를 비교한 것이다.

내장코드, 20행, 50행 그리고 100행의 코드를 각각 1000회 반복 실행하여 얻은 수치로써 적은 스크립트는 내장 코드와 비교 했을 때 약 1.8 배 정도 느렸으나 이는 내부 알고리즘을 개선했을 때 수행 속도를 더욱 단축할 수 있을 것으로 판단된다.

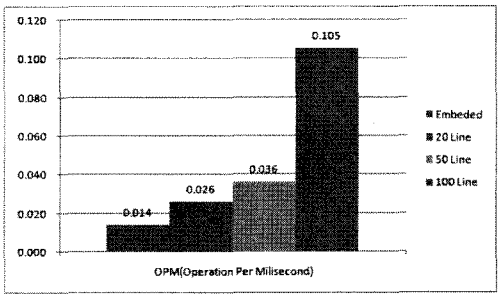

그림 7 수행 속도 비교

4. 결 론

기존의 시스템은 호스트와의 연동 및 명령 수행에 있어서 일정 명령을 응용 프로그램에 직접 작성하고 컴파일하여 복잡한 요구사항의 수정 및 검증에 따른 노력 및 시간이 효율적이지 않았다.

스크립트 프로그래밍 기반의 음향 발생 시스템은 기존의 시스템에 비교하여 확장성 혹은 개발 단계의 코드 확증 및 검증에 대한 노력 혹은 시간적인 효과를 증대 하여 향우 유지보수 및 요구사항 변경에 따른 추가적인 작업을 수월하게 적용할 수 있는 장점이 있다.

본 논문에서 개발된 시스템은 개발 목표에서 약 70% 정도 완료 하였지만 스크립트 프로그래밍을 중점적으로 적용하여 기존의 시스템에서 구현하기 힘들었던 확장성 , 유연성 및 수행 시 빠른 속도를 확보하였다.

참 고 문 헌

[1] 김귀하, 헬기 엔진 모델 구현을 위한 개발 기법에 관한 연구, 한국군사과학기술학회, 2006

[2] Christian Lindig, A Simple Object System for Lua, 2005

[3] Christian Vogler, Scripting Programs with Lua, 2004

[4] Wim Couwenberg, Simulating complex systems with Lua , 2005

감사의 글

본 연구는 건설교통부 항공선진화연구개발사업의 연구비 지원(훈련용 헬기 시뮬레이터개발 과제)에 의해 수행되었습니다.