한국ITS학회논문지
제11권, 제5호 (2012년 10월)
pp.136~146

열차 경로 패턴기반 애니메이션 성능 개선 기술 개발

Technology Development for Improving Animation Performance Based on Train Route Patterns

이 덕 희* (Duk-Hee Lee) 양 원 모** (Won-Mo Yang) 김 용 일*** (Yong-Il Kim) 양 윤 희*** (Yun-Hee Yang) 신 용 태**** (Yong-Tae Shin)

요 약

IT를 이용한 시뮬레이션 및 가상현실 기술이 발전하면서 시뮬레이션 결과를 사용자에게 효과적으로 전달하기 위한 애니메이션 기술에 대한 관심이 증가하고 있다. 애니메이션에 대한 재생품질, 재생속도, 입출력 속도, 저장공간 축소 등 애니메이션 성능을 개선하기 위한 다양한 노력이 있었다. 그러나 기존 연구는 대부분 애니메이션 영상에 대한 프레임별 이미지 압축에 초점을 맞추고 있다. 애니메이션의 저장용량 및 재생 속도를 획기적으로 개선하기 위해서는 애니메이션 자료를 벡터화 하고 시간적 공간적 중복성을 제거해야 한다. 본 연구는 애니메이션 자료에 대한 계층화, 벡터화를 통해 애니메이션 자료구조를 근본적으로 개선하였으며, 열차 이동경로 기반한 애니메이션 자료의 패턴화를 통해 애니메이션 자료의 시간적 공간적 중복성을 제거하여 저장공간을 현저히 축소하였고 입출력 속도 및 재생속도를 획기적으로 개선하였다. 실험결과 애니메이션 벡터화 이후 추가적인 패턴화를 통해 저장공간이 80%이상 축소되었으며 입출력 속도가 약 4배 향상되었다. 이러한 패턴화 기술은 객체의 이동경로가 존재하는 다양한 시뮬레이션 시스템의 애니메이션 자료 저장 방식으로 활용될 수 있으며, 아주 작은 애니메이션 자료 전송량으로 인해 사용자 맞춤형 애니메이션을 제공하기 적합한 기술로 활용될 수 있을 것이다.

핵심어 : A애니메이션, 가상화, 시뮬레이션, 애니메이션 패턴

Abstract

As information technology used for simulation and virtual reality developed, there is a growing interest in animation technologies which will effectively deliver simulation results to users. Various efforts have been made to improve animation performance, like playback quality and speed, input-output speed and storage space reduction. However, earlier studies generally focused on image compression frame by frame. To significantly improve storage space and playback speed, animation data should be vectorized. Also, spatial and temporal duplication have to be removed. In this study, animation data structure was improved fundamentally through establishment of hierarchy and vectorization. Also Spatial and temporal duplication of animation data was removed through vectorization based on train route. As a result, storage space was reduced, input-output speed and playback speed were considerably improved. According to the test, additional Patternization which followed vectorization brought reduction of over 80% in storage space and input-output speed was quadrupled. Patternization technology can be used as a proper storage method of animation data, and can provide user-specific animation by small data transmission.

Key words : Animation, Virtual, Simulation, Animation Pattern, XNA

* 주저자 : POSCO ICT 기술연구소 연구소장
** 공저자 : POSCO ICT 기술연구소 책임연구원
*** 공저자 : POSCO ICT 기술연구소 연구원
**** 공저자 : 숭실대학교 컴퓨터학부 교수
† 논문접수일 : 2012년 7월 9일
† 논문심사일 : 2012년 9월 7일
† 게재확정일 : 2012년 9월 21일

Ⅰ. 서 론

시뮬레이션이란 통계학의 가설검증 시험을 실제 시스템을 가상으로 모델링하여 구축한 시뮬레이터를 통해 실증하는 것을 말한다. 시뮬레이션은 IT기술이 발달하면서 많은 분야에 적용되고 있으며, 직접 검증 시 발생하는 비용, 시간, 안전등의 문제로 인해 앞으로 적용 분야가 더 확대 될 것으로 전망된다. 이러한 시뮬레이션은 교통 분야에서도 활발히 적용되고 있다. 교통 시뮬레이션에 대한 연구는 1950년대 미국에서 시작하여 현재는 자동차, 버스, 철도, 조선, 항공등 다양한 교통수단의 수요분석, 노선대안 평가, 운영대안 분석 등 많은 분야에서 사용되고 있다. 점점 교통수단이 많아지고 네트워크의 연결 복잡도가 증가해 앞으로는 다양한 교통수단에 대한 시뮬레이션 없이 통합적인 동적 운영대안을 마련하는 것이 불가능한 시대가 올 것이다. 교통 분야 뿐 만 아니라 국방, IT, 철강, 금융, 보안 등 다양한 산업 분야에서 이미 시뮬레이션 기술이 적용되고 있다. 국방 분야에서는 가상으로 전투를 시뮬레이션 할 수 있는 워 게임 시뮬레이션 시스템이 도입되어 실제 훈련을 수행하는 부대와 시뮬레이션 시스템 상의 가상의 부대가 동시에 모의 전투를 수행할 수 있는 수준으로 시뮬레이션 기술이 진보하였다. 특히 철도분야는 현재 국내에 수십여 지방자치단체에서 경전철을 도입 또는 계획 중에 있으며 앞으로 저탄소 녹색성장을 위해 대중교통으로 지속적인 철도 시스템의 도입이 전망된다. 이러한 철도 노선 건설은 사전에 시뮬레이션 분석을 통한 정확한 타당성 분석이 선행되어야 하며, 운영단계에서도 동적으로 운영전략을 대응할 수 있는 모니터링 시스템과 미래 운영전략을 분석하기 위한 예측시스템 등은 모두 시뮬레이션 시스템과 연동되어야 한다.

이러한 시뮬레이션에 대한 예측정확도가 향상되고 분석, 연산, 입출력, 애니메이션 등 IT기술이 발전하면서 가상화 기술이 주목받고 있다. 가상화란 물리적인 여러 시스템을 논리적으로 통합하거나 하나의 시스템을 논리적으로 분할하는 방법으로 효율적으로 자원을 사용하는 기술이다. 앞으로는 가상현실을 실생활에 이용하거나 수요예측, 제조공정에 대한 시뮬레이션을 수행하여 모니터링 시스템을 가상화하고 시뮬레이션 기술과 애니메이션 기술을 접목하여 사용자에게 적절하고 효과적인 예측 및 분석 정보를 제공하게 될 것이다. 시뮬레이션과 가상화 기술이 점차 발전하면서 이러한 분석 결과를 사용자에게 효과적으로 보여주기 위한 애니메이션 기술에 대한 관심이 증가하고 있다. 이것은 같은 시뮬레이션 분석결과를 사용자가 인지할 경우 숫자와 문자로 인지하는 것보다 애니메이션으로 통해 시각적으로 인지하는 것이 빠르기 때문이다. 애니메이션은 재생 품질, 재생 속도, 데이터 입출력 속도, 데이터 저장공간 크기 등에 대한 성능을 개선하기 위한 다양한 노력이 계속 되고 있다.

컴퓨터 애니메이션 기술이 발전하기 시작한 것은 1990년대에 들어오면서부터이다. 이전에는 텍스트 기반의 터미널 화면이나 이미지 기반의 화면 등을 통해 사용자는 정보를 인식해야만 했다. 텍스트와 이미지 기반으로 정보를 전달하기 때문에 모든 사용자에게 동일한 정적 애니메이션을 통해 정보를 전달했다. 사용자에 의해 특정 부분을 확대/축소하고 원하는 정보를 취사선택하여 볼 수 있는 가상 환경의 동적 애니메이션 환경은 실리콘그래픽스의 OpenGL과 Microsoft의 DirectX가 등장하면서부터 구축이 가능하게 되었다. 최초의 가상환경 구축 라이브러리가 발표 된지 불과 20년 정도 밖에 되지 않았으나, 3D 게임과 3D 가상현실이 우리에게 벌써 익숙해 졌다. 현재는 Microsoft의 XNA 4.0이 발표되어 애니메이션 품질 및 개발생산성이 향상되었고 크로스 플랫폼까지 지원되고 있다. 앞으로 휴대기기의 윈도우 모바일 OS가 발전하고 Slate PC와 Desk Top을 동시에 지원하는 Windows 8이 발표되면 사용자들이 언제 어디서든 시뮬레이션 시스템이나 가상 시스템에 접속할 수 있게 될 것이다. 이렇게 되면 많은 사용자와 다양한 기기에서 개별적인 사용자 요청이 발생하고 이러한 분석 결과를 사용자 맞춤형으로 각기 다른 애니메이션을 표시해 줄 수 있어야 한다. 또한 많은 이용자와 기기에 애니메이션 자료를 전송하기 위해서는 애니메이션 데이터를 최적

화시키고 전송하는 자료의 크기도 줄여야 한다. 따라서 애니메이션 자료에 대한 근본적인 저장방법을 고민하고 획기적인 압축방법을 사용하여야 한다.

본 연구는 철도분야에서 노선 및 운영대안 분석을 위한 정적 시뮬레이션 분석 시스템 및 실시간 데이터를 분석하는 동적 시뮬레이션 분석 시스템에 모두 적용 될 수 있는 애니메이션 기술에 대한 것으로, 애니메이션 입출력 성능 개선을 통해 애니메이션 자료 저장 방식 및 압축방식에 대한 새로운 방법을 제안하고자 한다. 구체적으로는 열차의 경로 패턴을 기반으로 애니메이션 자료를 계층화, 벡터화, 패턴화하여 애니메이션 자료의 저장 용량을 획기적으로 줄였으며, 입출력 속도를 개선하여 애니메이션 성능을 향상시켰다. 이러한 애니메이션 성능 향상 기술은 방대한 애니메이션 자료를 아주 작은 공간에 저장할 수 있으며, 입출력에 대한 빠른 속도를 보장하고, 시스템 성능이 낮은 단말 기기에서도 고품질의 애니메이션을 재생할 수 있으며, 미리 저장된 영상이 아닌 사용자와 양방향으로 데이터를 주고받는 사용자 맞춤형 애니메이션을 제공 할 수 있는 장점이 있다. 이러한 애니메이션 성능 개선 기술은 철도 분야에만 적용되는 것이 아니라 시뮬레이션 객체의 상태가 변하고 이동하는 경로가 존재하는 다른 분야의 시뮬레이션 시스템에도 적용 가능하다.

Ⅱ. 연구배경 및 관련연구

1. 애니메이션 기술 동향

기존 애니메이션 기술은 VRML, OpenGL, DirectX, XNA등 다양하다. 이러한 애니메이션 기술은 작동환경 및 구현대상에 따라 다른 특징을 가지고 있는데 주요한 차이는 다음과 같다.

- OpenGL은 SGI사에서 개발한 3차원 그래픽 라이브러리로 그래픽스 API 부문에서 우수한 성능을 보이며, 운영시스템, 윈도우 시스템, 하드웨어 환경에 독립적이다. OpneGL은 CAD, 가상현실, 정보시각화, 비행 시뮬레이션 등에 활용되고 있다[1].
- DirectX는 API(Application Programming Interfaces) 모음으로 멀티미디어, 윈도우, 세가, 드림케스트, 엑스박스 등을 위한 게임 개발에 널리 쓰이고 있다. DirectX는 2D, 3D 애니메이션은 물론 다양한 입출력기기에 대한 접근 및 멀티미디어 재생을 위한 API를 제공한다[2].
- XNA는 Microsoft에서 개발한 다중 플랫폼 게임 개발 플랫폼으로 현재 PC, XBOX, Zune, 윈도우폰까지 지원한다. XNA는 XNA Framework, 개발도구(Visual Studio), 웹상의 커뮤니티 (XNA Creators Club)를 모두 포함한다. XNA는 윈도우와 XBOX 상에서 구동되며 닷넷 프레임워크를 필요로 한다[3].

2. 기존 연구 고찰

애니메이션은 공간적, 시간적 중복성을 가지고 있으며, 이중 시간적 중복성을 제거하는 것이 애니메이션 압축의 주요한 핵심이다. 시간적 중복성을 줄이기 위해서는 움직임 예측을 통하여 시간적 상관관계가 있는 동영상의 프레임간의 움직임 벡터를 통해 이전 프레임과의 차이를 구하여 그 값을 압축함으로써 중복성을 줄일 수 있다[5].

애니메이션 압축 기술은 다양한 그래픽 기술에 대한 요구로 2000년 AFX(Animation Framework eXtension)으로 시작되어 2004년 ISO/IEC 14496-16 표준이 나온 이래 꾸준하게 많은 기술들이 소개되고 있다. Bone-based 애니메이션은 Face/Body 애니메이션을 Humanoid가 아닌 generic articulate 3D object로 확장하여 모델링과 애니메이션이 가능하도록 skin과 skin에 연결된 Bone의 구조로 애니메이션을 표현하는 기술이다. 모핑 애니메이션은 소스 모델과 타겟 모델로부터 일련의 연속적인 중간모델들을 만들어서 부드러운 애니메이션이 가능하도록 한다. 그 외에도 키 프레임 애니메이션 압축, 매 프레임에 해당하는 Geometry를 가지고 있는 시퀀스의 방축방법인 Frame-based Animation Compression 등이 있다[6].

애니메이션 자료의 벡터화 및 패턴화의 기존 연구를 살펴보면, Jinghua Zhang 등은 연속된 프레임

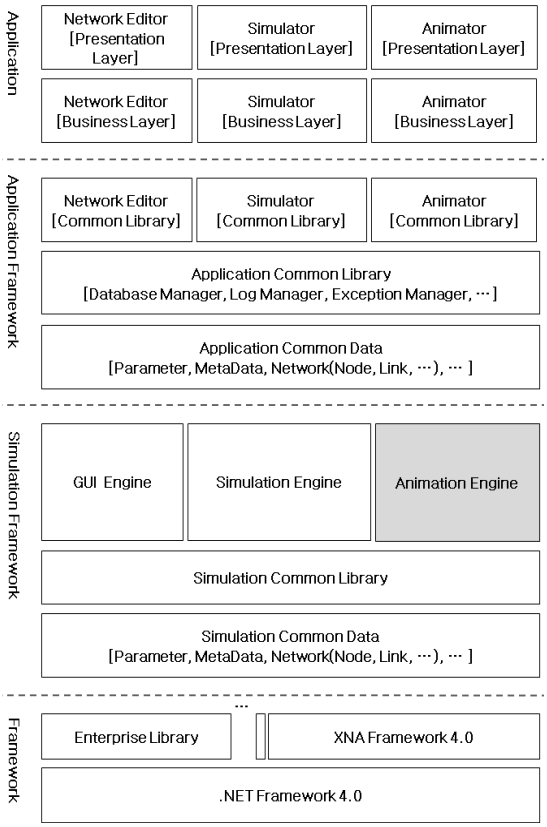
간의 모션 분석을 통해 각 프레임의 모션 벡터 세트를 구성 한 후, 이를 사용하여 3D 애니메이션에서 서로 다른 영역 간 모션 벡터를 공유할 수 있는 방법론을 제시하였다[7]. O.Petrik, L.Vasa은 3D 애니메이션의 압축을 위한 MPEG-4 FAMC(Frame-based Animated Mesh Compression) 알고리즘의 개선을 연구하였다[8]. A.R.Vamsidhar 등은 Linear Discriminant Analysis (LDA)에 기반한 3D 애니메이션 geometry component의 압축 방법론을 제안하였다. Principal Component Analysis(PCA)을 사용하여 데이터를 변형 하여 고압축시 손실을 줄이는 데 강점을 가지게 됨을 확인하였다[9]. 김낙우 등은 MPEG 압축 영역에서의 모션 벡터 정보를 이용하여 비디오에서의 효율적 모션 해석 방법을 제안하였다. MPEG시퀀스로부터 추출된 움직임 벡터의 재해석을 통하여, 영상 내 객체의 좌표 및 움직임 각도 등을 고려하던 기존의 알고리즘과 차별되어 움직이는 객체의 속도와 가속도 등을 고려한 새로운 객체 추출 및 추적 알고리즘을 제시하고 그 성능을 비교하였다[10]. 임훈, 이종원은 이미지의 라인 패턴을 자동으로 추출한 후, 자동으로 라인패턴을 입혀주는 방식을 연구하였다. 이를 이용하여 전체 프레임 중간에 애니메이터가 완성한 키 프레임을 받고, 다른 프레임들과의 대응 라인을 찾은 후 패턴을 자동으로 입혀 중간 프레임들을 얻어내는 연구를 수행하였다[11]. 원인수 등은 동영상의 장면 전환을 검출하기 위하여 장면 간 유사도와 장면 내 유사도를 정의하고 프레임 영상 간 비교 과정에서 생성된 패턴을 사용하여 장면 전환을 검출하는 방법을 제안하였다[12]. 이경희 등은 기존의 탐색 방법을 통해 동영상 움직임 벡터의 분포를 예상해보고 그에 따른 영역에 대하여 새로운 고속 탐색 패턴을 적용하여 움직임 벡터를 예측하는 방법을 제안하였다[5]. 진주경 등은 내용기반 중복 동영상 검출 알고리즘을 제안하였다. 동영상을 장면으로 분할한 후 각 장면 내에서 동영상 디코딩 시 얻어지는 매크로블록의 움직임 벡터와 프레임을 이용하여 중복 동영상을 검출하였다[13].

기존연구 고찰 결과, 지금까지의 애니메이션 압축 기술은 이미지 자체를 대상으로 압축하고 데이터의 벡터화 기술은 이미지의 영상을 인식하여 벡터화하는 차원에 머물고 있음을 알 수 있다. 또한 이미지 차원의 압축은 압축률에 한계를 지니며 고압축 시 원본 데이터 손실이 불가피하다. 기존 애니메이션 저장 방식은 많은 저장 공간이 필요하며 저사양 단말기에서 애니메이션 입출력 속도를 보장하기 어렵다. 따라서 이를 보완하기 위한 새로운 애니메이션 압축 기술이 필요하다. 본 연구는 애니메이션화 이전 원시 데이터를 계층화, 벡터화, 패턴화하여 기존 기술보다 데이터 저장 공간을 획기적으로 줄이고 애니메이션 입출력 속도를 개선한 기술을 제시하고자 한다.

Ⅲ. 패턴 기반 애니메이션 성능 개선

1. 시뮬레이션 솔루션 구조

시뮬레이션 솔루션의 구조는 <그림 1>의 시뮬레이션 구성도와 같이 크게 Framework, Simulation Framework, Application Framework, Application 계층으로 구성된다. 우선 Framework는 Microsoft의 .NET Framework 4.0과 Enterprise Library 및 애니메이션을 위한 XNA Framework 4.0을 포함한다. XNA는 객체지향 언어 C#을 통해 DirectX를 완벽하게 제어할 수 있는 최신 애니메이션 기술로, 다양한 OS에 대해 크로스 플랫폼을 지원함으로 본 연구의 애니메이션 기술 개발환경으로 사용되었다. Simulation Framework는 철도망 네트워크를 구성하는 노드, 링크, 분기기, 신호기 등 시설물 과 모델의 속성에 대한 메타데이터와 시뮬레이션 파라미터를 포함하는 Simulation Common Data와 공용함수인 Simulation Common Library위에 GUI(Graphic User Interface) 구성을 위한 GUI Engine과 시뮬레이션에 필요한 각종 시뮬레이션 모델링 라이브러리를 포함하는 Simulation Engine 그리고 애니메이션을 위한 Animation Engine으로 구성되어 있다. Application Framework은 Application을 구현하기 위한 기본 라이브러리 계층으로 Application에서 사용하는 메타데이터 및 파라미터를 포함하는 Application Common Data와 데이터베이스 접속, 로그관리, 예외처리 등을 관리하는 Application Common Library 그리고 각 모듈별로 사

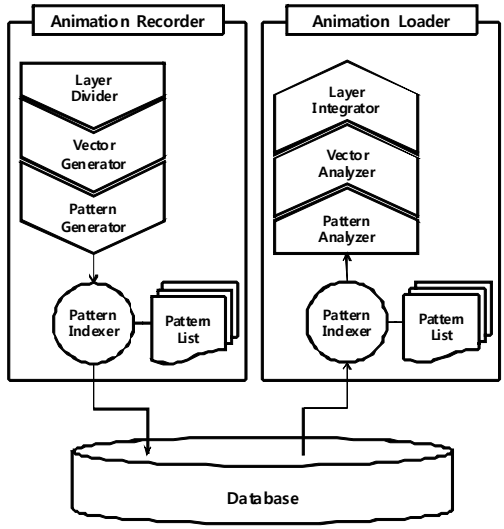


〈그림 1〉 시뮬레이션 솔루션 구성

용하는 공용라이브러리로 구성되어 있다. Application은 철도망을 편집하는 Network Editor와 시뮬레이션을 수행하는 Simulator, 결과를 애니메이션으로 재생하는 Animator로 구성되어 있다.

2. 애니메이션 엔진 구조

시뮬레이션 솔루션의 엔진은 크게 GUI 엔진, 시뮬레이션 엔진, 애니메이션 엔진으로 구분할 수 있으며, 애니메이션 엔진 상세 구조는 <그림 2>와 같다. 본 연구의 애니메이션 엔진은 기존 애니메이션 엔진과 비교하여 애니메이션 자료의 벡터화를 위하여 Layer Divider와 Vector Generator를 추가하였으며, 중복자료 패턴화를 위하여 Pattern Generator, Pattern Indexer를 추가하였다. 이와 같은 애니메이션 엔진을 사용하면 기존 애니메이션 엔진 보다 용량, 속도 및 품질을 획기적으로 개선할 수 있다.



〈그림 2〉 애니메이션 엔진 구조

애니메이션 엔진은 크게 시뮬레이션 결과를 데이터베이스에 저장하는 Animation Recorder모듈과 데이터베이스에 저장된 애니메이션 자료를 읽어 동영상으로 가공하는 Animation Loader 모듈로 구성되어 있다.

Animation Recorder는 애니메이션 자료를 여러 데이터 계층으로 구분하는 Layer Divider와 애니메이션 자료를 벡터 형태로 변환하는 Vector Generator, 변환된 벡터 데이터를 패턴화 시키는 Pattern Generator를 포함한다. Pattern Generator는 경로별 애니메이션 자료에 대해 기 저장한 패턴에서 중복 여부를 검사한다. 새로운 패턴인 경우 Pattern Generator는 Pattern Indexer를 통해 패턴ID를 생성하고 Pattern List에 등록한다. 이동객체(열차) i의 패턴(P_i)는 식 (1)과 같은 패턴해석함수 $f(X)$에 의해서 동일성 여부를 판단한다. 동일 패턴이 아닐 경우 Pattern Indexer에 의해 새로운 패턴 ID를 부여받는다. 패턴판단함수 $f(X_i)$는 이동객체 i의 속성(G_i), 이동객체 i의 이동경로(R_i), 이동객체 i이 이동경로(R_i)에 진입한 이후의 진행시간(T_i^r), 이동경로(R_i) 상의 이동객체 i를 제외한 다른 환경의 변화(E^r)에 의해서 패턴을 판단 한다 여기서 G_i는 이동객체 i의 속도, 가속도, 출력, 제동력 등 각종 속성을 포함하며, R_i은 현재

출발지부터 다음 목적지까지의 경로로 링크와 노드의 집합으로 구성된다. T_i^r는 경로 R_i에 진입한 이후 경과시간이며, E^r는 이동객체 i를 제외한 분기기, 신호기, 다른 열차 등 경로 상 주변 환경에 대한 동적인 상태 변화를 의미한다.

$$P_i = f(X_i) = f(G_i, R_i, T_i^r, E^r) \quad \text{식(1)}$$

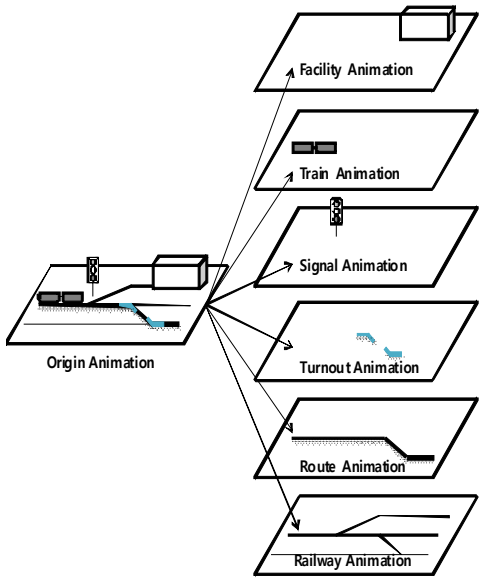
단, 여기서
P_i: 이동객체 i의 패턴
f: 패턴판단 함수
X_i: 이동객체 i의 패턴을 판단하기 위한 주변환경 집합
G_i: 이동객체 i의 속성집합
R_i: 이동객체 i의 단위경로
T_i^r: 이동객체 i가 경로 R_i에 진입한 이후 경과시간
E^r: 경로 R_i상의 주변환경에 대한 동적인 변화

Animation Loader는 데이터베이스에서 애니메이션 패턴 자료를 읽고 패턴을 분석하여 시계열 데이터로 조합하는 Pattern Analyzer와 벡터자료를 해석하여 화면에 표시하기 위한 객체의 위치와 속성자료로 가공하는 Vector Analyzer, 가공된 각 Layer별 애니메이션 자료를 하나로 프레임으로 통합하는 Layer Integrator로 구성되어 있다.

3. 애니메이션 패턴화 기법

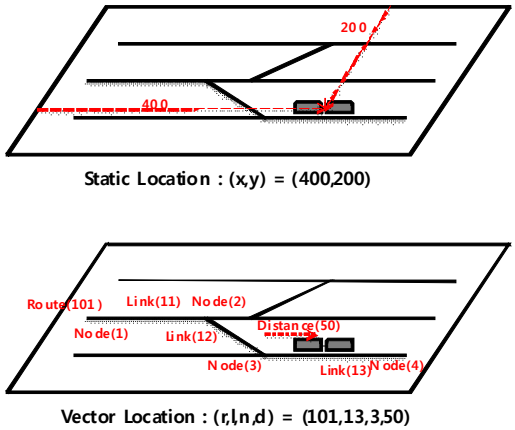
애니메이션 패턴화를 위해서 우선 애니메이션 자료를 계층별로 구분하고 벡터화 시켜야 한다. 계층화를 하는 이유는 계층별로 애니메이션 자료의 벡터화 방법이 다르기 때문이며, 보다 효율적인 데이터 압축방법을 적용하기 위함이다. 벡터화를 수행하는 이유는 애니메이션 데이터에 대한 저장방식을 장면에서 개별 객체의 위치와 속성 값으로 변경하여 애니메이션 데이터의 저장용량을 최소화 하는 작업으로 데이터를 패턴화 시키기 위한 사전 작업과 동시에 사용자 맞춤형 애니메이션을 제공할 수 있도록 하기 위하여 원시자료를 저장하기 위함이다. 이렇게 계층화, 벡터화, 패턴화를 적용하면 애니메이션 자료에 대한 시간적 공간적 중복성을 제거하여 데이터 저장용량을 최소화 시킬 수 있어 아주 작은 용량에 많은 애니메이션 원시 벡터 자료를 저장할 수 있다. 또한 모든 원시자료가 벡터 형태로 저장되어 있어 사용자 요구사항에 의한 다양한 애니메이션 영상을 복원하여 재생할 수 있다. 이러한 계층화, 벡터화, 패턴화는 애니메이션 재생 시에는 반대 순서대로 적용된다. 애니메이션 자료의 계층화, 벡터화, 패턴화 세부적인 절차는 다음과 같다.

- **애니메이션 자료의 계층화** : <그림 3>과 같이 전체 애니메이션 영역의 자료를 철도망 네트워크, 경로, 분기기, 신호기, 열차, 시설물 등 다양한 객체 Layer별로 애니메이션 정보를 구분한다. 전체 애니메이션 영역을 각 Layer로 구분하는 이유는 각 Layer별로 애니메이션 자료를 벡터화 하고 시간적 중복성을 제거하기 위함이다. 예를 들어 시간에 따라 위치가 이동하지 않는 Layer에 대해서는 단순히 속성 정보만 변경하면 된다. 이렇게 하면 각 Layer별로 벡터화 하는 방법을 다르게 적용하여 최적의 저장공간에 애니메이션 데이터를 저장할 수 있게 된다.

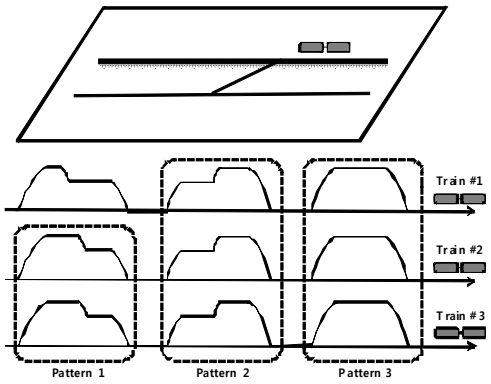


〈그림 3〉 애니메이션 계층화 개념도

- **애니메이션 자료의 벡터화** : 분리된 Layer별 애니메이션 정보를 속도, 좌표, 노드ID, 링크ID등을 이용하여 벡터형태로 변환한다. 철도망 네트워크의 경우 노드(점)와 링크(선분)로 구성되며 노드는 위경도 좌표로 벡터와 된다. 경로는 노드와 링크의 집합으로 구성되며, 분기기, 신호기의 위치는 위경도 좌표로 표시되고 링크 연결 정보가 저장된다. 열차는 각종 속성정보와 함께 위치가 경로에 대한 상대적 이동 거리로 좌표를 벡터값으로 표시하고 시설물은 각 모서리의 좌표값을 벡터화 시켜 저장한다. <그림 4>는 기존 좌표 저장방식과 벡터 저장방식에 대한 차이를 설명하고 있는 열차위치의 벡터화 개념도이다.
- **애니메이션 자료의 패턴화** : 애니메이션 자료를 계층화 벡터화 하는 것만으로도 데이터의 양은 현저히 줄어든다. 그러나 장시간 시뮬레이션의 결과를 저장 할 경우 전체 애니메이션 영역에 대한 원시자료를 모두 저장하는 것은 많은 저장 공간을 필요로 하게 된다. 선로를 애니메이션 패턴화를 위해 Segment로 구분한다. 동일한 Segment를 지나는 같은 재원의 열차의 경우 주행에 영향을 미치는 다른 열차가 없으면 애니메이션 자료가 동일하다. 이 경우 애니메이션 자료를 추가로 저장하지 않고 기존 애니메이션 패턴화 자료를 활용할 수 있다. 이와 같이 시간에 대한 중복성을 제거하여 재활용한다면 작은 애니메이션 벡터 자료를 활용하여 장시간 애니메이션을 표시할 수 있게 된다. <그림 5>는 열차의 경로를 기반으로 열차의 속도 프로파일과 주변 열차를 분석하여 패턴을 구분하는 방법에 대한 개념도 이다. 패턴해석함수에 의해 애니메이션 자료를 패턴으로 구분하여 Pattern Indexer를 통해 패턴 ID를 생성하고 Pattern List에 패턴이 저장되어 관리된다. 이렇게 각 개별 열차별로 패턴을 적용하면 부분적인 애니메이션에 대한 동일한 패턴의 중복저장을 피할 수 있어 애니메이션 자료 저장공간을 현저히 줄일 수 있는 동시에 전체 애니메이션 품질은 유지할 수 있게 된다.



〈그림 4〉 열차 위치 벡터화 개념도



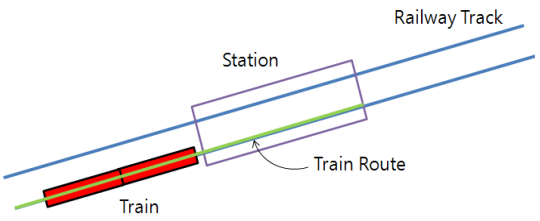
〈그림 5〉 열차경로 기반 애니메이션 패턴화 개념도

Ⅳ. 테스트 결과

이와 같이 패턴화를 통해 성능을 개선한 애니메이션 엔진을 프로토타입 애니메이터에 적용하여 패턴화 적용 전 후에 대한 성능을 비교 분석하였다. 테스트를 위한 프로토타입은 철도망을 입력하는 철도망 편집기, 열차 주행을 시뮬레이션 하는 시뮬레이터 및 애니메이터를 포함한다.

1. 애니메이션 자료크기 비교분석

철도 시뮬레이션 솔루션의 애니메이터에 본 연구에서 제안한 경로 패턴기반 애니메이션 개선기술을 적용하여 적용전과 애니메이션 자료의 크기 및



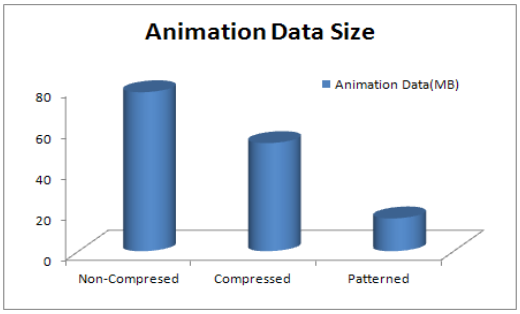
〈그림 6〉 애니메이터 프로토타입

입출력 성능에 대해 비교 분석 하였다. 정확한 패턴화 적용에 대한 개선효과를 검증하기 위해 계층화와 벡터화를 적용한 솔루션에 대해 패턴화를 적용하여 압축 전, 압축 후, 패턴화 적용 후 애니메이션 자료의 양과 저장시간, 로딩시간을 비교하였으며, 동일 시스템 환경에서 테스트를 수행하였다.

테스트 대상 철도망 네트워크는 인천 2호선 경전철 노선으로 양방향 1일 운행계획에 대한 열차 운행을 시뮬레이션을 수행하였으며, 차량기지 및 주박기지 열차 주행 및 작업 시뮬레이션을 포함하였다. 애니메이터 프로토타입은 <그림 6>과 같다.

애니메이션 자료에 대한 테스트 결과 <그림 7>과 같이 압축 전 77.87MB에서 압축 후 52.95MB로 약 32% 감소하였으며, 패턴화 적용 후 16.05MB로 약 80% 감소하였다.

이와 같이 패턴화 기술 적용 이전에는 모든 애니메이션 벡터 데이터를 데이터베이스에 저장하였으나 패턴화 기술 적용 이후에는 중복된 애니메이션 패턴이 발생하면 애니메이션 데이터를 저장하지 않고 패턴 ID만을 저장함으로 저장공간이 크게 감소함을 알 수 있었다.



〈그림 7〉 패턴화 전후 애니메이션 용량 비교
(비압축, 압축, 패턴화)

2. 애니메이션 자료 입출력 속도 비교분석

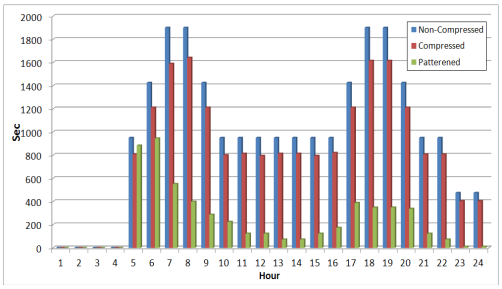
경로 패턴 기반 애니메이션 개선 기술을 적용하면 애니메이션 자료의 크기를 획기적으로 감소 시킬 수 있을 뿐만 아니라 애니메이션 자료의 입출력 속도 및 애니메이션 품질에도 상당한 개선이 이루어졌다. 동일 노선 및 테스트 시스템에서 입출력 속도에 대해 패턴화 이전과 이후를 비교 분석 하였다.

애니메이션 총 저장시간에 대한 테스트 결과 <그림 8>과 같이 압축 전 23,763초에서 압축 후 20,198초로 약 15% 감소하였으며, 패턴화 기술 적용 후 5,635초로 약 76% 감소하였다. 저장시간 감소는 시뮬레이션 속도에도 영향을 준다. 빠른 애니메이션 저장시간은 그만큼 고속 시뮬레이션 속도를 보장한다.

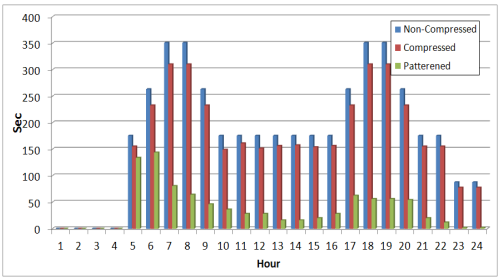
애니메이션 총 로딩시간에 대한 테스트 결과 <그림 9>과 같이 압축 전 4,387초에서 압축 후 3,878초로 약 12% 감소하였으며, 패턴화 기술 적용 후 904초로 약 80% 감소하였다. 로딩시간 감소는 애니메이션 품질에 영향을 준다. 빠른 애니메이션 로딩시간은 단위 시간당 읽을 수 있는 애니메이션 프레임수가 증가하여 애니메이션 품질을 향상 시킨다.

<그림 8~9>의 애니메이션 저장 및 로딩시간에 대한 분석결과를 보다 자세히 살펴보면, 패턴화 적용 전에는 주행열차의 편성 수가 애니메이션 자료 크기에 영향을 미치기 때문에 열차 운행 횟수가 많아지는 첨두시간대의 애니메이션 자료크기가 커짐으로 저장 및 로딩시간이 증가하였으며, 데이터 압축 후에도 이런 경향은 유사하다.

패턴화 기술 적용 후 시뮬레이션 초기에는 애니메이션 저장 및 로딩시간이 패턴화 이전과 비교하여 큰 차이를 보이지 않지만 이후 저장 및 로딩 시간이 급격히 감소하는 것을 알 수 있다. 이것은 편도별 열차 주행이 완료되는 시간부터 비슷한 패턴이 많이 발생하기 때문이다. 특히 오전 오후 첨두시간대에 많은 열차들이 운행하고 있음에도 애니메이션 저장 및 로딩 시간이 크게 늘어나지 않고 있음을 알 수 있다.

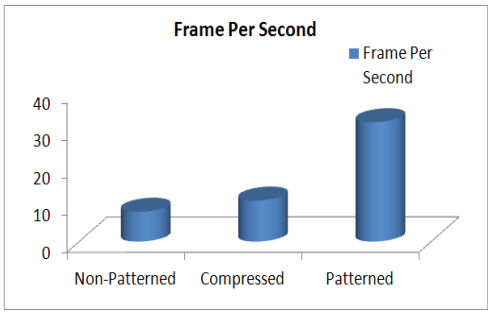


〈그림 8〉 시간별 애니메이션 저장 시간 비교



〈그림 9〉 시간별 애니메이션 로딩 시간 비교

이와 같은 빠른 저장 및 로딩 시간은 단위 시간당 애니메이션 프레임 자료 생성을 빠르게 하여 애니메이션 품질을 향상시킨다. <그림 10>과 같이 압축 전 8 FPS(Frame per Second)에서 압축 후 11FPS으로, 패턴화 적용 후 32FPS까지 애니메이션 재생 품질이 개선되었다.



〈그림 10〉 애니메이션 출력 속도 비교(최대 배속 기준)

V. 결 론

많은 산업 분야에서 다양한 데이터 수집 장치가 도입되고 데이터 해석에 대한 IT기술이 발전하면서 사용자는 단순한 모니터링 보다 미래를 예측하거나 수집된 데이터를 해석한 보다 분석적인 결과를 요구하게 되었고 복잡한 정보 속에서 빠른 의사결정을 위해 다양한 분석정보를 시각적 애니메이션으로 확인하려는 경향이 점차 증가하고 있다. 이와 같이 시뮬레이션 및 가상화가 실생활 및 다양한 산업분야에 도입되면서 많은 사용자 및 다양한 단말장치에 맞춤형 애니메이션 제공하기 위해 기존 이미지 압축 기술 보다 획기적인 애니메이션 성능 향상 기술이 요구되고 있다. 본 연구에서 애니메이션 자료에 대한 시간적 공간적 중복성을 제거하여 애니메이션 데이터를 획기적으로 압축하고 애니메이션 자료 입출력 속도를 향상 시킬 수 있는 계층화, 벡터화, 패턴화 기법을 제시하고 있다. 실험 결과 이러한 패턴화 기술을 적용하면 애니메이션 저장공간을 1/5수준으로 감소시킬 수 있으며 입출력 속도를 4배 이상 개선하여 애니메이션 재생속도 및 품질을 높일 수 있음을 살펴보았다. 이러한 애니메이션 압축기술은 애니메이션 원본 데이터를 보존하면서 작은 공간에 더 많은 애니메이션 자료를 저장 할 수 있고 애니메이션 자료 전송에 필요한 데이터 크기가 매우 작아 향후 많은 다양한 단말기에 사용자 맞춤형 애니메이션 전송 및 재생이 가능할 것으로 보인다. 또한 제시한 애니메이션 성능 개선 기술은 철도분야 시뮬레이션 시스템에 국한되지 않으며 애니메이션 객체의 이동경로가 존재하는 다른 다양한 시뮬레이션 시스템 또는 가상화 시스템의 애니메이션 자료 저장방식으로 활용 될 수 있을 것이다.

참고문헌

[1] Zhanwei Wu, Heng Wang, Hua Zhang, “Virtual Scene Modeling Technology Based on OpenGL and 3dsMAX”, Computational Intelligence and Design (ISCID), 2011.

[2] Xufeng Li, Fenghua Shi, Yonghui An, “Visualization Modeling of Mine Roadway Based on DirectX”, Information and Computing Science, 2009.

[3] 김윤기, 김슬기, 최원석, 김지주, 최영미, 주문원, 윤태복, “XNA를 이용한 캐주얼 게임 개발”, 한국멀티미디어학회, 추계학술발표논문집, pp. 653-655, 2009.11.

[4] M.Ujaldon, SG. Ebner, J.Saltz, “On the capabilities of the GPU for general purpose computing”, Online:OSUBMI_TR_2004_n18.pdf.

[5] 이경희, 석진욱, 서재원, “영역에 따른 탐색 패턴을 이용한 움직임 벡터 예측 방법”, 대한전기학회, CICS 정보 및 제어 학술대회 논문집, pp.421-422, 2007.10.

[6] 안정환, “MPEG AFX: 애니메이션 압축 기술”, 한국통신학회 정보와통신 제24권 제4호, pp.67-75, 2007.4.

[7] Jinghua Zhang, Jinsheng Xu, Huiming Yi, “Octree-Based 3D Animation Com pression with Motion Vector Sharing”, Information Technology, 2007.

[8] O.Petrik, L.Vasa, “Improvements of MPEG-4 standard famc for efficient 3D animation compression”, 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2011.

[9] A.R.Vamsidhar, P.K.Bora, S.Das, “Animation Geometry Compression using the Linear Discriminant Analysis”, Computer Vision, Graphics & Image Processing, 2008.

[10] 김낙우, 김태용, 강응관, 최종수, 강은관, “압축 영역에서 움직임 벡터의 재추정을 이용한 비디오 해석 기법”, 전자공학회지 제39권 제3호 pp.78-87, 2002. 5.

[11] 임훈, 이종원, “애니메이터의 예술적 특성을 반영한 라인패턴 라동생성 기법”, 한국콘텐츠 학회 게임&엔터테인먼트 논문지 제2권 제1호, pp.27-34, 2006.3.

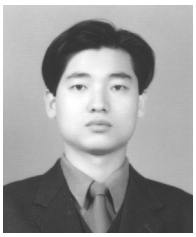
[12] 원인수, 조주희, 나상일, 진주경, 정재협, 정동석, “장면의 유사도 패턴 비교를 이용한 내용기반 동영상 분할 알고리즘”, 한국멀티미디어학회, 멀티미디어학회논문지, 제14권 제10호, pp.1252-1261, 2011.10.

[13] 진주경, 나상일, 정동석, “움직임과 영상 패턴 서술자를 이용한 중복 동영상 검출”, 대한전자공학회, 전자공학회논문지-SP, 제48권 SP편 제4호, pp.107-115, 2011.7.

저자소개

이 덕 희 (Lee, Duk-Hee)

2010년 2월 ~ 현 재 : 숭실대학교 IT정책경영학과 박사과정
2004년 2월 : 포항공과대학교 정보통신대학원 석사졸업
1990년 1월 ~ 현 재 : POSCO ICT 기술연구소 연구소장
1990년 2월 : 경북대학교 수학과 학사졸업



양 원 모 (Yang, Won-Mo)

2001년 9월 ~ 현 재 : POSCO ICT 기술연구소 책임연구원(차장)
2011년 8월 : 서울대학교 지구환경시스템공학부(교통공학) 박사졸업
2001년 2월 : 서울대학교 지구환경시스템공학부(교통공학) 석사졸업
1999년 2월 : 서울대학교 토목공학과(도시공학) 학사졸업

김 용 일 (Kim, Yong-Il)

2011년 1월 ~ 현 재 : POSCO ICT 기술연구소 연구원
2011년 2월 : 서울대학교 건설환경공학부(교통공학) 석사졸업
2009년 2월 : 서울대학교 지구환경시스템공학부 학사졸업

양 윤 희 (Yang, Yun-Hee)

2011년 10월 ~ 현 재 : POSCO ICT 기술연구소 연구원
2011년 8월 : 서울대학교 건설환경공학부(교통공학) 석사졸업
2009년 2월 : 서울대학교 농경제사회학부 학사졸업

신 용 태 (Shin, Yong-Tae)

1995년 3월 ~ 현 재 : 숭실대학교 컴퓨터학부 교수
1994년 5월 ~ 1995년 1월 : University of Iowa / Michigan State University 객원교수
1994년 2월 : University of Iowa Computer Science 박사졸업
1985년 2월 : 한양대학교 산업공학과 학사졸업