KIPS Trans. Softw. and Data Eng.
Vol.10, No.6 pp.243~250
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2021.10.6.243

Development of an X3D Python Language Binding Viewer Providing a 3D Data Interface

Ha Seong Kim† · Myeong Won Lee††

ABSTRACT

With the increased development of 3D VR applications augmented by recent VR/AR/MR technologies and by the advance of 3D devices, interchangeability and portability of 3D data have become essential. 3D files should be processed in a standard data format for common usage between applications. Providing standardized libraries and data structures along with the standard file format means that a more efficient system organization is possible and unnecessary processing due to the usage of different file formats and data structures depending on the applications can be omitted. In order to provide the function of using a common data file and data structure, this research is intended to provide a programming binding tool for generating and storing standardized data so that various services can be developed by accessing the common 3D files. To achieve this, this paper defines a common data structure including classes and functions to access X3D files with a standardized scheme using the Python programming language. It describes the implementation of a Python language binding viewer, which is an X3D VR viewer for rendering standard X3D data files based on the language binding interface. The VR viewer includes Python based 3D scene libraries and a data structure for creation, modification, exchange, and transfer of X3D objects. In addition, the viewer displays X3D objects and processes events using the libraries and data structure.

Keywords : X3D, X3D Language Binding, 3D Scene Access, Python 3D, Python X3D Language Binding, X3D Python Viewer

3D 데이터 인터페이스를 제공하는 X3D Python 언어 바인딩 뷰어 개발

김 하 성† · 이 명 원††

요 약

최근 VR/AR/MR 기술과 3D 주변 장치의 발전에 의해 3D 가상현실 응용 서비스 개발이 증가하고 있으며 이에 따라 3D 데이터의 호환성과 이식성에 대한 요구가 증가하고 있다. 여러 장치의 응용 프로그램의 데이터를 공동으로 사용할 수 있도록 하기 위해서는 3D 표준 데이터 형식의 파일 처리가 요구된다. 이 때 가상환경 장면 처리에 필요한 공통의 파일 형식과 함께 함수와 변수 클래스들을 표준화된 라이브러리와 데이터구조를 제공한다면 응용 별로 서로 다른 파일 형식과 데이터구조의 사용으로 인해 발생하는 불필요한 처리 과정을 생략할 수 있어서 보다 효율적인 시스템 구성이 가능할 것이다. 본 연구에서는 이러한 공통의 데이터 파일과 데이터구조 사용 기능을 위해서 특정의 프로그래밍 언어를 이용하여 공용의 3D 표준 파일에 접근하여 다양한 서비스를 개발할 수 있도록 표준화된 데이터를 생성하고 저장할 수 있는 프로그래밍 바인딩 도구를 제공하는 것이 목적이다. 이를 위해 본 논문에서는 Python 프로그래밍 언어를 이용하여 X3D 파일을 표준화된 방식으로 접근할 수 있도록 클래스와 함수를 포함하는 공통의 데이터구조를 정의하며 이 언어 바인딩 인터페이스를 기반으로 X3D 장면 데이터 파일을 렌더링 하는 X3D VR 뷰어인 Python 언어 바인딩 뷰어 구현에 대해 기술한다. 이 뷰어는 X3D 오브젝트의 생성, 변경, 교환 및 전송을 위한 Python 기반의 3D 장면 라이브러리들과 데이터구조를 포함한다. 그리고 이를 이용하여 X3D 오브젝트를 디스플레이하고 이벤트들을 처리한다.

키워드 : X3D, X3D 언어 바인딩, 3D 장면 접근, Python 3D, Python X3D 언어 바인딩 뷰어, X3D Python 뷰어

1. 서 론

최근 3D 데이터 관련의 빅데이터 처리가 증가하면서 지능적 데이터 처리에 있어서의 효율적인 데이터 구조 및 접근 방법에 대한 필요성이 증가하고 있다[1-3]. 기존의 3D 데이터가 렌더링에 초점이 맞춰져 있었다면 빅데이터 처리가 가속화되고 있는 현 시점에서는 3D 데이터 분석을 정확하게 할 수 있도록 효율적인 프로그래밍을 제공하는 표준화된 데이터 파일과 데이터처리 인터페이스를 제공하는 것이 필요하다[4].

다른 데이터 분석과 달리 3D 데이터는 화면에 장면이 디스플레이되는 렌더링 과정에서 필요한 많은 속성 데이터를

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터 지원사업의 연구결과로 수행되었음(IITP-2021-2016-0-00312).
† 준 회 원 : KTDS 사업수행1팀 사원
†† 종신회원 : 수원대학교 컴퓨터학부 교수
Manuscript Received : January 4, 2021
First Revision : February 16, 2021
Accepted : March 6, 2021
* Corresponding Author : Myeong Won Lee(mwlee@suwon.ac.kr)

사용한다[5-8]. 3D 데이터 분석을 위해서는 프로그래밍 과정에서 렌더링 속성이나 성능을 위해 효율적인 데이터 생성 방법과 이때 요구되는 표준화된 데이터 접근 인터페이스를 제공하는 일이 필요하다[9]. 또한, Python 언어는 데이터 분석 영역에서 관련 기법들에 대한 오픈 소스 제공도 증가하고 있고 가시화 데이터에 대한 분석에도 많이 활용되고 있다. 아직은 2D 이미지 영상에 대한 데이터 분석 연구가 많으나 3D 데이터에 대한 분석도 증가하고 있는 추세이다[10,11]. 본 연구의 배경은 표준화된 3D 데이터 접근 인터페이스를 정의하여 Python 프로그래밍 언어 환경에서 효율적인 3D 데이터 분석을 위해 사용하도록 하는데 있다.

클래스나 함수의 표준화된 3D 데이터 파일의 접근 방법의 제공은 기존 프로그램이나 데이터들의 재사용을 쉽게 하게 하는 장점을 갖는다. 3D 응용 개발에는 막대한 시간과 자원이 요구되므로 응용별로 서로 다른 파일이나 데이터 구조를 갖게 되면 3D 파일 처리를 위해 중복적인 일을 반복해야 하는 비효율성을 피할 수가 없다[12,13]. 본 연구에서는 이러한 문제를 해결할 수 있도록 표준화된 데이터 구조와 접근 방법을 제시하고 이를 Python 프로그래밍 언어로 구현한 3D 뷰어 개발 사례를 소개한다.

본 논문의 2장에서는 기존의 관련 연구 현황을 기술하며, 3장에서는 X3D 가상현실 시스템 구성을 소개하고 X3D 표준 파일 내 데이터 요소들이 장면에 표현되기까지의 과정을 설명한다. 4장에서는 본 연구에서 구현한 X3D Python 언어 바인딩 시스템 구성에 대해 기술한다. 5장에서는 X3D 파일의 각 데이터 요소들을 접근하는데 필요한 인터페이스 구성과 기능을 설명한다. 6장에서는 X3D Python 언어 바인딩 뷰어의 구현 과정을 설명하고 결과를 보여준다. 7장은 결론과 응용 및 향후계획을 포함한다.

2. 관련 연구 현황

3D 표준 데이터로 대표적인 것으로는 ISO/IEC 14772 VRML (Virtual Reality Modeling Language)와 ISO/IEC 19775-1 X3D (Extensible 3D)가 있다[14,15]. X3D 는 VRML을 기반으로 하여 설계되었고 XML (Extensible Markup Language) 형식으로 인코딩되어 있으며, 2004년 버전 1.0이 개발된 이래 현재에 이르기까지 업데이트가 계속되어 지금 버전 4.0 개발이 진행 중인 ISO/IEC 국제표준이다[15,16].

OpenGL 등 그래픽스 라이브러리의 민간 표준화단체인 크로노스 그룹에서 개발한 3D 데이터 호환 형식과 관련된 표준으로는 COLLADA (COLLAborative Design Activity)와 glTF가 대표적이다. COLLADA 는 대화형 3D 응용을 위한 호환성 파일 형식으로 디지털 자산 교환을 목적으로 하고 있어서 다양한 3D 파일 형식을 XML 스키마에 넣고 있어서 렌더링 자체는 각 다양한 3D 파일 형식에 의존할 수 밖에 없는 구조이다[17,18]. glTF (Graphics Language Transmission Format) 은 3차원 장면과 모델을 위한 파일 형식으로 JSON 파일 형식으로 장면 그래프 생성에 필요한 정보를 기술하고 있다[19].

X3D 장면에 접근하는 인터페이스로서 언어 바인딩을 정의한 국제표준으로는 X3D ECMAScript ISO/IEC 19777-1와 X3D Java ISO/IEC 19777-2의 두 종류의 프로그래밍 언어 바인딩 국제 표준이 있다[20-21]. 본 논문에서의 장면 접근 인터페이스는 Python 프로그래밍 기반의 언어 바인딩을 정의하는 것으로서 기존의 자바 및 자바스크립트 기반의 언어 바인딩 표준과는 달리 Python 프로그래밍 언어로 정의되는 장면 접근 인터페이스이다.

3. X3D 가상현실 시스템 구성

3D 가상현실 시스템을 구성하기 위해서는 표현하려는 3차원 가상공간을 저장하거나 교환하기 위해 표준화된 데이터 포맷을 필요로 한다. 이것은 표준 데이터 포맷을 사용하지 않고 독자적인 파일 포맷으로 생성된 가상환경을 이용하면 하나의 어플리케이션에서는 아무 문제가 없으나 여러 어플리케이션에서 공통으로 사용할 때는 매번 파일 형식에 맞춰서 프로그램을 고쳐야 하는 문제가 발생하기 때문이다. 이러한 문제를 해결하기 위해서 본 논문에서는 3차원 가상환경의 저장 형식으로 X3D 파일 포맷을 사용하여 가상현실 시스템을 구성하고 이를 통해서 가상환경을 생성하고 제어하는 방법에 대해 기술한다.

본 연구에서의 가상현실 시스템은 3차원 가상환경을 저장하는 X3D 파일 포맷을 읽고 수정하고 시뮬레이션하고 저장하는 가상환경 에디터의 기능으로 설명된다. 이 시스템은 가상환경을 생성하고, 가상환경의 변화를 장면에 표현하며 이 때 필요한 3차원 데이터의 처리를 제공하는 기능을 갖는다. X3D 파일로 저장된 가상환경은 X3D 파서에 의해 가상현실 시스템 안으로 들어와서 가상환경을 구성하는 객체들로 분해된다. X3D 파일은 유형에 따라 VRML, 바이너리, XML 인코딩으로 구분되며 파서에 의해 X3D 컴포넌트들로 분해된다. X3D 컴포넌트는 프로토타입과 X3D 노드들로 구성된다. 각 X3D 노드들은 3차원 가상공간의 장면을 구성하는 각 객체들의 기하 및 속성 정보를 포함하며 장면 그래프 관리자에 의해 장면 그래프 데이터 구조를 생성하는데 사용되고 3D 브라우저가 이 장면 그래프 데이터 구조를 사용하여 각 객체들을 화면에 표현되도록 한다. X3D 노드와 객체들은 외부 프로그램에 의해서 접근될 수 있으며 장면 그래프 데이터 구조에 변화를 줄 수 있다.

X3D 파일은 유형에 따라 VRML 인코딩, 바이너리 인코딩, XML 인코딩 등으로 표현되기도 한다. X3D 가상현실 시스템에서 각 객체들은 노드와 프로토타입들로 구성되며 X3D 노드들에 의해 가상환경을 위한 장면 그래프가 구성이 된다. X3D 노드는 가상환경에 표현될 가상 객체들을 유형별로 정의하며 이를 위해 다양한 기하물체 형식, 속성 표현 방법, 변

환 기술 방법 등이 각 노드에 포함된다. 가상현실 시스템은 X3D 노드들을 하나씩 처리하며 이를 이용하여 장면에 표현될 객체들로 구성된 장면 그래프를 생성하고 3D 브라우저가 이 장면 그래프를 이용하여 가상환경을 화면에 나타내도록 한다.

가상환경의 장면 변화나 시뮬레이션은 두 가지 유형으로 처리할 수 있다. 첫 번째는 일반적인 방법으로 브라우저에서 제공하는 사용자 상호작용 기능을 이용하여 브라우저에 의한 파라미터 값 변화에 따라 이벤트 처리를 이용하여 장면의 변화를 가져올 수 있다. 두 번째로는 X3D 코드 중에 외부의 입출력 장치를 통한 이벤트 처리에 필요한 데이터 입력을 받아들이는 별도의 어플리케이션 프로그램을 연결하는 방법이 있다.

본 연구에서는 위와 같은 기능의 X3D 기반의 가상현실 시스템을 이용하여 가상환경을 생성할 때, 브라우저나 외부 어플리케이션 프로그램에서 X3D 노드 처리, 장면 그래프 생성, 화면에 3D 객체 표현 등의 작업을 할 때 필요한 공통의 데이터구조와 라이브러리를 프로그래밍 언어 기반으로 정의하는 것이다.

4. X3D Python 언어 바인딩 인터페이스

X3D 표준 데이터 파일을 이용하여 가상환경 생성을 포함하는 응용 프로그램을 개발할 때 3D 파일에 정의되어 있는 각 3D 객체를 가상현실 시스템 안으로 불러오는 경우에 3D 데이터 형식이나 프로그래밍 환경에 따라 클래스, 필드, 함수 등의 처리가 달라진다. 앞 장에서 기술한 가상현실 시스템이 X3D 객체들을 처리할 때 각 어플리케이션에서 이 객체들을 처리할 데이터 형식이나 함수를 정의하게 되는데 객체들이 표현되어야 할 과정은 동일함에도 불구하고 정의해야 할 데이터구조는 사용자의 프로그래밍 방식에 따라 다르게 표현될 수 밖에 없는 것이 보통이다. 똑같은 기능을 해야 하는 객체의 데이터 저장 방식이 전부 다르게 구성되어서 어플리케이션 프로그램의 규모가 커질수록 더욱 큰 문제를 발생하게 된다. 이러한 문제를 해결하기 위해서는 가상현실 객체의 데이터구조를 통일화함으로서 호환하여 사용할 수 있도록 효율적인 프로그램 구조를 제공하는 것이 필요하다.

이를 위해서 본 연구에서의 가상현실 시스템에서는 X3D 파일을 읽을 때 데이터들을 추출해서 사용할 프로그래밍 환경에서 함수, 클래스, 변수 등 데이터구조를 미리 정의하고 각 응용 개발 시에 특정 언어에서 사용할 수 있도록 매핑을 시켜주는데 이 과정을 언어 바인딩이라고 한다. 예를 들면 3D 프리미티브를 렌더링하는 Box, Cone, IndexedFaceSet 등과, 색깔을 설정할 수 있는 Apperance, Material 그리고 이 클래스에서 사용할 수 있는 모든 변수들을 위해 각 데이구조를 Python 프로그래밍 언어로 정의했다.

X3D 물체들로 구성되는 3차원 가상현실 장면을 Python 프로그래밍 언어를 이용하여 처리하는 시스템 구성은 Fig. 1과 같다. 가상현실의 각 물체들은 X3D 노드로 표현되며 이들은 Python 프로그래밍 환경에서 Python X3D 물체로서 구성되며 Python X3D 뷰어를 통해 시각화하여 화면에 렌더링하는 과정으로 구현된다.

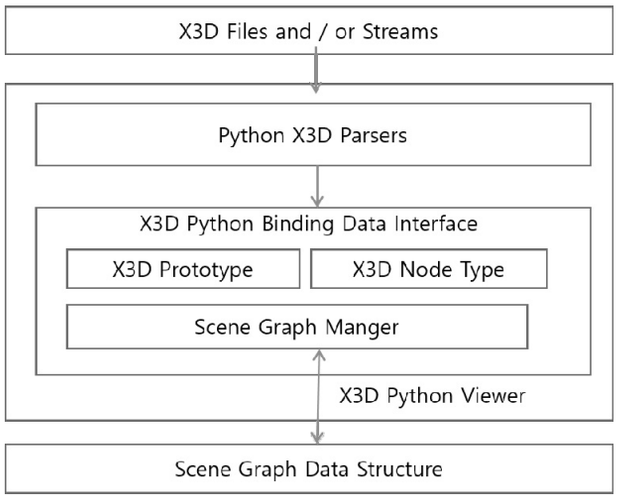

Fig. 1. X3D Python Language Binding Organization

5. X3D Python 언어 바인딩 진행 절차

본 절에서는 X3D 장면 인터페이스를 구성하는데 있어서 X3D 파일을 이용하여 Python 프로그래밍 언어 바인딩으로 X3D 객체를 구성하고 화면에 렌더링을 진행하는 절차에 대해 기술한다. 이를 위해서는 표현하려는 물체의 내용을 기술하는 X3D 파일의 구성, X3D 파일을 처리하는 Python 언어 바인딩 코드 작성, 3D 뷰어에 의한 렌더링의 세 단계 과정을 거치게 된다.

Fig. 2는 X3D 장면 구성의 한 예로서 객체를 클릭하여 장면의 변화를 가져올 수 있도록 X3D의 Route 문을 이용하여 객체의 마우스 클릭을 정의하고 3D 뷰어가 이벤트 처리를 할 수 있도록 하는 X3D 코드이다. Route 문은 X3D 물체를 기술하는 X3D 노드들간에 필요한 값들의 입출력을 가능하게 하는 기능이다. 즉 하나의 노드의 출력 값을 다른 노드의 입력 값으로 전달하여 장면의 변화를 가져올 수 있도록 해준다. X3D 장면을 동적으로 변화시키는 기능이라고 할 수 있다. 이를 위해 첫 번째 단계로 TouchSensor 가 마우스 클릭 이벤트를 받아서 Box 물체의 색상을 파란색에서 빨간색으로 변경하는 예제이다. Fig. 2에서 head 태그 안에는 X3D 파일의 메타 데이터가 담겨진다. 보통 파일의 저작권자, 파일의 이름과 같은 정보가 들어간다. Scene 태그부터 3D 정보에 대한 데이터가 담겨져 실제 3D를 구현하게 된다. Material 태그에 있는 diffuseColor 는 객체의 색깔을 정의하며 각각 R, G, B 값이 된다. TouchSensor 태그에 의해 X3D 파일이 마우스 클릭에 대한 이벤트를 받을 수 있다. field 태그는 X3D 파일 안에서 사용될 변수를 정의한다. 예를 들어 입력값만 받는 SFBool 타입의 isOver 변수, 출력값으로 사용될 SFColor

```
<X3D profile="Immersive">
  <head>
    <meta content="TouchSensorIsOverEvent.x3d" name="filename"/>
  </head>
  <Scene>
    <Group>
      <Shape>
        <Appearance>
          <Material DEF="MAT" diffuseColor="0 0 1"/>
        </Appearance>
        <Box/>
      </Shape>
      <TouchSensor DEF="TS"/>
    </Group>
    <Script DEF="SC" url="SAIExample1">
      <field accessType="inputOnly" name="isOver" type="SFBool"/>
      <field accessType="outputOnly" name="diffuseColor_changed" type="SFColor"/>
    </Script>
    <ROUTE fromField="isOver" fromNode="TS" toField="isOver" toNode="SC"/>
    <ROUTE fromField="diffuseColor_changed" fromNode="SC"
      toField="set_diffuseColor" toNode="MAT"/>
  </Scene>
</X3D>
```

Fig. 2. An X3D Example Including Route Statements

```
class SAIExample1(X3DScriptImplementation, X3DFieldEventListener):
  def __init__(self):
    SAIExample1.RED = [1.0, 0, 0]
    SAIExample1.BLUE = [0, 0, 1.0]
    self.initialize()
  def __del__(self):
  def setBrowser(self, browser):
  def setFields(self, externalView, fields):
    self.fields = fields
  def initialize(self):
    self.isOver = SFBool(fields.get("isOver"))
    self.diffuseColor = SFColor(fields.get("diffuseColor_changed"))
    self.isOver.addX3DEventListener(this)
  def shutdown(self):
  def eventsProcessed(self):
  def readableFieldChanged(self, evt):
    if self.evt.getSource() == self.isOver:
      if self.isOver.getValue():
        self.diffuseColor.setValue(RED)
      else :
        self.diffuseColor.setValue(BLUE)
    else :
      print("Unhandled event")
  RED = []
  BLUE = []
  isOver = SFBool()
  diffuseColor = SFColor()
```

Fig. 3. An X3D Python Example for Route Statements

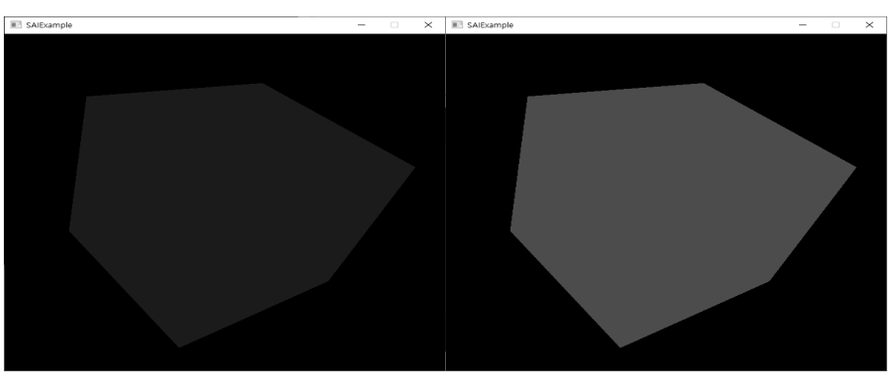

Fig. 4. X3D Event Processing using X3D Route Statements

의 diffuseColor_changed 변수를 보여준다. ROUTE 태그는 이벤트 생성 노드의 출력 필드를 이벤트 발생 노드의 입력 필드와 연결시킨다. 보통 fromNode 와 toNode는 태그의 DEF 값으로 연결이 된다. 첫 번째 ROUTE 노드는 이벤트를 발생시키는 TouchSensor 노드를 Script 노드로 연결시켜준다. 두 번째 노드는 Script 노드를 Material 노드로 연결시켜준다. 그 결과 이벤트가 발생하면 Material 노드의 색깔이 변경되어 Box 물체의 색상이 파란색으로 변경이 된다. Fig. 3은 두 번째 단계로 X3D 장면 접근 인터페이스를 Python 언어 바인딩으로 구현한 코드를 보여준다. Fig. 4는 세 번째 단계로

```
Class CBox(CX3DGeometryNode)
    m_strNodeName = "Box"
    size = [2.0, 2.0, 2.0]
    solid = True

    def __init__(self):
    def setSize(self, vec):
    def getSize1(self, value):
    def getSize3(self):
    def setSolid(self, value):
    def getSolid(self):
    def Draw(self):
```

Fig. 5. An Example of an X3D Class Interface for an X3D Box Node

X3D Route 문을 Python 언어 바인딩 뷰어에서 보여주는 결과 화면을 나타낸다. 왼쪽은 해당 파일을 읽었을 때의 상태고 오른쪽은 마우스 클릭을 하는 순간 이벤트 처리를 하여 색상이 변경되는 것을 보여준다.

Fig. 5는 이벤트 처리에서 육면체 도형을 나타내는 X3D 클래스 인터페이스 정의 예를 Python 언어 바인딩으로 정의한 SAI 코드를 보여준다. Fig. 6은 X3D 객체 생성을 위해 필요한 함수의 SAI 코드 사용 예이며 X3DScene(), X3DShapeNode(), X3DNode(), createNode(), setGeometry(), getRootNodes() 등이 함수 인터페이스를 나타낸다.

6. X3D Python 바인딩 뷰어 개발 및 결과

본 절에서는 X3D 가상현실 파일을 읽은 후에 파일 내 X3D 객체들을 화면에 표현하는 Python 바인딩 뷰어 개발과 Python 바인딩 라이브러리를 구현한 예제에 대하여 기술한다. 앞에서 기술한 X3D Python 언어바인딩을 이용하여 Python 에서 사용할 수 있는 X3D 표준 데이터와 함수를 만들어 뷰어에서 X3D 파일을 삽입하여 사용한다.

현재 X3D Python 바인딩 뷰어의 주요 기능으로는 X3D 파일 읽기, Vertex 모드로 보기, Wire 모드로 보기, Face 모드로 보기, 줌인, 줌아웃, 뷰 포인트 회전 등이 있다. 그리고, X3D Python 바인딩 뷰어는 사용자의 동작, 즉 이벤트를 받아서 처리하는 사용자 인터페이스를 포함한다.

이 인터페이스를 통해서 마우스의 이벤트 처리가 가능하며, 버튼과 메뉴, 그리고 X3D파일 구조를 확인할 수 있는 트리뷰(TreeView)가 있다. 트리뷰에서는 읽어들인 X3D파일과 데이터를 X3D 노드 구조에 맞추어 X3D 데이터를 계층구조로 클래스화한다. 그리고 클래스화한 노드들을 XML 구조에 맞춰 트리구조로 만드는 Python SAI 모듈이 있다. 이 모듈

```
def initialize(self):
    self.children = self.fields.get("children")
    scene = X3DScene()
    shape = X3DShapeNode()
    box = X3DGeometryNode()

    # Create nodes directly in the parent scene
    scene = X3DScene(self.browser.getExecutionContext()
)
    shape = X3DShapeNode(scene.createNode("Shape"))
    box = X3DGeometryNode(scene.createNode("Box"))
    shape.setGeometry(box)
    scene.addRootNode(shape)

    # Create children using the createX3DFromString
service
    vrmlCmd = "PROFILE Interchange  Shape { geometry
Sphere{} }"
    tmpScene = X3DScene()
    nodes = X3DNode()
    tmpScene =
self.browser.createX3DFromString(vrmlCmd)
    nodes = tmpScene.getRootNodes()
    size = tmpScene.getRootNodeSize()
```

Fig. 6. An Example of X3D Function Interfaces for the Creation X3D Objects

은 Box, Sphere, Cone, Cylinder와 같은 간단한 기하 오브젝트부터 Transform, Material과 같이 오브젝트들의 속성들을 변화시켜준다. Python X3D 바인딩 뷰어는 Python 장면 접근 인터페이스에서 만들어진 트리 구조를 이용하여 클래스화한 X3D 노드의 속성과 데이터를 읽은 후 OpenGL을 이용하여 3D 오브젝트들을 생성시키고 화면에 렌더링하는 역할을 한다.

Python X3D 바인딩 뷰어에서 주요 과정은 X3D 파일을 읽어서 X3D 노드를 만든 후에 X3D 노드를 Python 의 Tree 구조로 매핑시키는 과정이다. 이 과정은 Python 기반의 XML 파서를 개발하여 처리하였다. 현재까지 개발한 Python 바인딩 뷰어의 기능은 X3D 의 Primitive 타입과 IndexedFaceSet, Texture 타입 등을 처리할 수 있으며, X3D 파일 형식으로 인체 캐릭터 데이터 구조를 정의하는 HAnim 데이터[16-17]의 노드 구조 파싱 및 각 노드들의 버텍스 연결 후 폴리곤을 생성하며, URL 속성을 이용하여 텍스처 디렉토리 안에 있는 이미지를 불러서 오브젝트에 텍스쳐를 매핑하여 캐릭터를 렌더링하는 기능을 포함하고 있다.

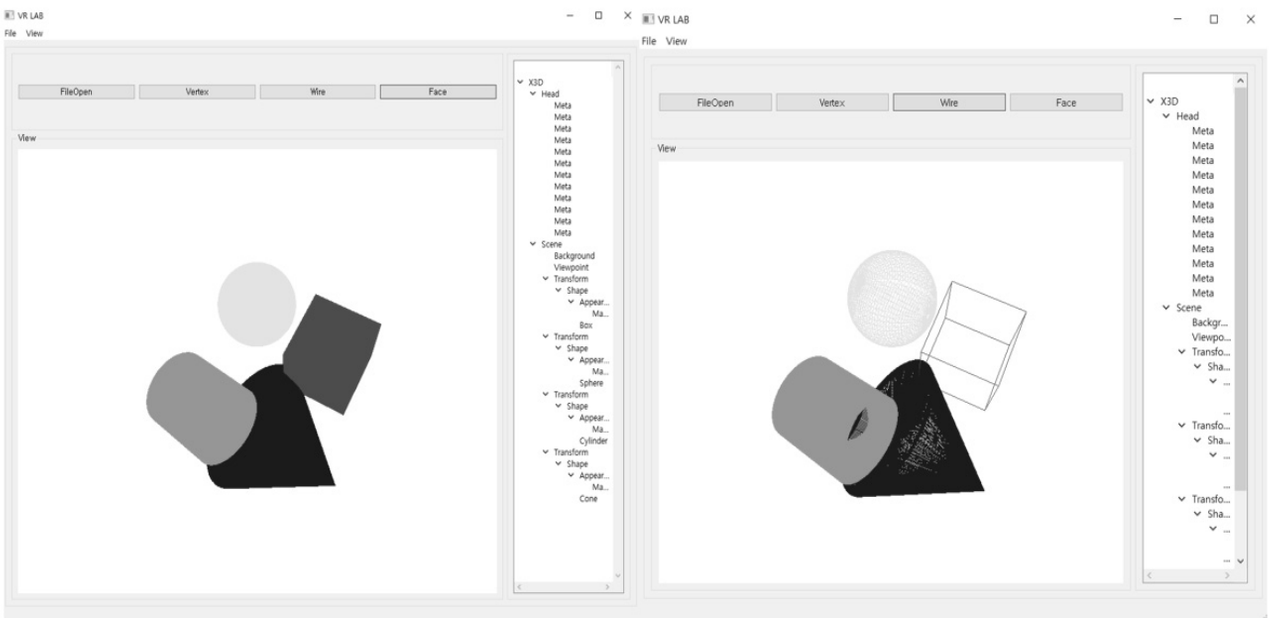

Fig. 7. Implementation Results of a Python X3D Viewer with X3D Primitives

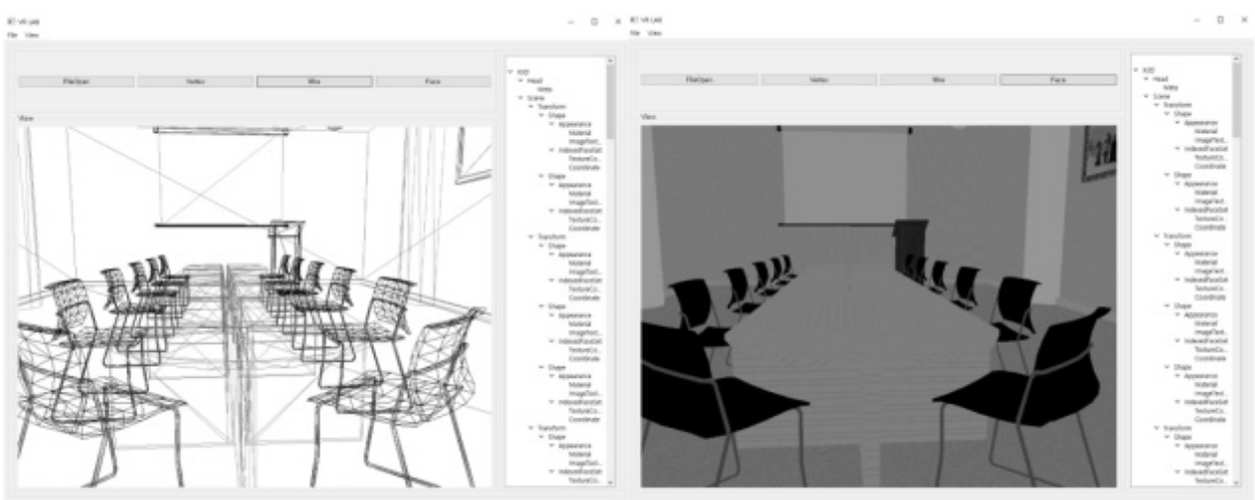

Fig. 8. Implementation Results of an X3D Python Viewer with Texture Mapping

Fig. 7은 X3D Python 언어 바인딩 뷰어에서 X3D 프리미티브 파일을 렌더링한 모습을 보여준다. X3D 물체의 선분만 표현이 되는 Wire 모드와 3D 면이 표현되는 Face 모드로 렌더링한 모습이다. 우측의 트리뷰를 통하여 X3D 파일 내 정의되어 있는 각 객체의 속성을 확인할 수가 있다. Fig. 8은 일반 3D 응용프로그램에서 제작한 여러 텍스처 이미지를 포함하는 실내 모델링 파일을 X3D 로 변환 시킨 뒤에 본 연구에서 개발한 X3D Python 언어 바인딩 뷰어에서 렌더링한 결과를 보여준다.

이 뷰어의 구현에는 윈도우 10 운영체제에서 프로그래밍은 Python 3.8.5을 이용하였으며 그래픽스 라이브러리는 PyOpenGL 3.10을 사용하였고 Numpy 1.185을 이용하여 Python에서 좌표 변환에 필요한 실수 연산을 가능하게 하였으며, 텍스처 매핑 등 이미지 처리를 위해서 Pillow 8.10을 사용하여 개발하였다.

7. 결 론

본 논문에서는 3차원 가상현실 공간을 구성하는 응용 프로그램을 개발할 때 3D 가상환경 인터페이스 표준인 X3D 장면 접근 인터페이스를 Python 프로그래밍 환경에서 사용할 수 있도록 Python X3D 데이터 인터페이스를 정의하고 구성

된 장면의 렌더링을 확인하고 제어할 수 있는 X3D Python 프로그래밍 언어 바인딩 뷰어 개발에 대해 기술하였다.

X3D Python 프로그래밍 언어 바인딩은 X3D 장면 접근 인터페이스를 Python 프로그래밍을 이용하여 활용하는 방법을 제공하는데 의의가 있다. X3D Python 뷰어는 X3D 파일을 읽어들여 파서를 거쳐서 X3D 데이터를 분석하여 X3D 노드들로 분해한 후 이를 이용하여 가상환경의 장면 그래프에 가상환경을 표현할 때까지의 과정을 처리하는데 있어서 X3D 장면 접근 인터페이스에서 정의하는 데이터구조를 이용하여 Python 으로 작성하여 가상환경을 생성한다.

본 연구에서의 X3D Python 프로그래밍 언어 바인딩은 Python 을 이용하여 X3D를 이용하여 가상환경을 생성할 수 있도록 표준화된 인터페이스를 제공한다. 본 연구의 결과는 Python 으로 개발하는 3D 가상현실 프로그램에서 재사용이 가능한 3차원 가상환경 구축과 3차원 물체의 대화형 시뮬레이션 기능이 필요한 다양한 가시화 응용 분야에서 활용될 수 있다. 향후 연구로는 본 연구에서의 Python 프로그래밍 언어 바인딩 기능을 기반으로 하여 X3D 데이터와 머신 러닝 알고리즘을 이용한 지능형 3차원 가상공간 시뮬레이션 기법이 포함된다.

References

[1] Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Niessner, "SceneGrok: Inferring Action Maps in 3D Environments," *ACM Transactions on Graphics (TOG)*, Vol.33, No.6, pp.1-10, Nov. 2014.

[2] Rui Ma, Honghua Li, Changqing Zou, Zicheng Liao, Xin Tong, and Hao Zhang, "Action-Drive 3D Indoor Scene Evolution," *ACM Transactions on Graphics (TOG)*, Vol.35, No.6, pp.1-13, Nov. 2016

[3] Marc Petit, Henry Boccon-Gibod, and Christophe Mouton, "Evaluating the X3D Schema with Semantic Web Tools," *Proceedings of the 17th International Conference on 3D Web Technology (Web3D Conference 2012)*, Aug. 2012.

[4] X. Cao and M. Klusch, "Advanced Semantic Deep Search for 3D Scenes," *2013 IEEE Seventh International Conference on Semantic Computing*, Irvine, CA, pp.236-243, 2013.

[5] Chul-Hee Jung, Mingeun Lee, and Myeong Won Lee, "Development of Exchangeable Character Animation Using a Moion Sensor," *J Korean Society for Computer Game*, Vol.27, No.4, pp.237-246, Dec. 2014.

[6] Andreas Plesch and Mike McCann, "The X3D geospatial component: X3DOM implementation of GeoOrigin, GeoLocation, GeoViewpoint, and GeoPositionInterpolator nodes," *Proceedings of the 20th International Conference on 3D Web Technology (Web3D Conference 2015)*, June 2015.

[7] ISO/IEC 19774-1: Humanoid Animation (HAnim): Architecture, 2019.

[8] ISO/IEC 19774-2: Humanoid Animation (HAnim): Motion data animation, 2019.

[9] Jun Liu, Wenzhen Su, and Yu Sun, "3D Model Semantic Automatic Annotation Based on X3D Scene," *Proceedings of the 2013 International Conference on Computational and Information Sciences (ICCIS '13)*. IEEE Computer Society, USA, pp.282-285, 2013.

[10] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko, "Learning Deep Object Detectors from 3D Models," *ICCV 2015*, Vol.1, pp.1278-1286, 2015.

[11] Stefano Berretti, Mohamed Daoudi, Pavan Turaga, and Anup Basu, "Representation, Analysis, and Recognition of 3D Humans: A Survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, Vol.14, Iss.1s, pp.1-36, 2018.

[12] Olavo da Rosa Belloc, Rodrigo B. D. Ferraz, Marcio Calixto Cabral, Roseli de Deus Lopes, and Marcelo Knorich Zuffo, "Virtual Reality Procedure Training Simulators in X3D," *Proceedings of the 17th International Conference on 3D Web Technology (Web3D Conference 2012)*, Aug. 2012.

[13] Jan Schilbach, "An event-based framework for animations in X3D," *Proceedings of the 17th International Conference on 3D Web Technology (Web3D Conference 2014)*, Aug. 2014.

[14] ISO/IEC 14772-1: 1997 and ISO/IEC 14772-2: 2004, VRML 97 Functional and EAI, ISO/IEC, 2004.

[15] ISO/IEC 19775-1: 2013 Extensible 3D (X3D) Architecture and base components V3, ISO/IEC, 2013.

[16] ISO/IEC 19775-2: 2015 Extensible 3D (X3D) Scene Access Interface, 2015, ISO/IEC, 2015.

[17] Khronos Group, COLLADA 1.4 Quick Reference, Lulu.com, 2014.

[18] Mark Barnes. "COLLADA" SIGGRAPH'06: ACM SIGGRAPH 2006 Courses, Jul. 2006.

[19] Arne Schilling, Jannes Bolling, and Claus Nagel, "Using glTF for streaming CityGML 3D city models," *Proceedings of the 21st International Conference on Web3D Technology (Web3D 2016)*, pp.109-116, Jul. 2016.

[20] ISO/IEC 19777-1: 2006 Extensible 3D (X3D) language bindings: ECMAScript, ISO/IEC, 2006.

[21] ISO/IEC 19777-2: 2006 Extensible 3D (X3D) language bindings: Java, ISO/IEC, 2006.

김 하 성

https://orcid.org/0000-0003-4933-8670
e-mail : khsh5592@naver.com
2020년 수원대학교 컴퓨터학부(학사)
2020년 ~ 현 재 KTDS 사업수행1팀 사원
관심분야 : Computer Graphics, Virtual Reality, Machine Learning

이 명 원

https://orcid.org/0000-0002-8948-6187
e-mail : mwlee@suwon.ac.kr
1981년 서울대학교(학사)
1984년 서울대학교 계산통계학과 전산전공(석사)
1990년 The University of Tokyo, 정보과학(박사)
1984년 ~ 1986년 DACOM 연구소
1990년 ~ 1993년 Kubota Corporation & U. of Tokyo
1993년 ~ 1996년 KT 연구소
1996년 ~ 현 재 수원대학교 컴퓨터학부 교수
관심분야 : Computer Graphics, Computer Animation, Virtual Reality, Augmented Reality, Mixed Reality