Journal of The Korea Society of Computer and Information
Vol. 20, No. 4, April 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.4.049

파이선 스크립트를 이용한 태양계 행성 시뮬레이터 구현

최 은 영*, 이 임 건**

Implementing Solar System Simulator using Python Script

Eun-Young Choi*, Imgeun Lee**

요 약

본 논문에서는 3D 애니메이션 도구인 마야를 이용하여 태양계를 모델링하고 각 행성의 움직임과 물리적인 속성들을 정밀하고 객관적으로 구현하여 태양계의 구조를 시각적으로 쉽게 이해할 수 있도록 시뮬레이터를 구현하였다. 메뉴를 이용한 마야의 모델링으로는 정밀한 물리적 속성 값을 표현하기 어려우므로 파이선 스크립트를 이용하여 각 행성의 특성을 표현하였다. 제안하는 시뮬레이터는 가상현실 분야나 학생들의 교육 자료로서 사용 가능하도록 모델링과 렌더링을 실사와 동일하게 구현하였다. 또한 마야 애니메이션에서의 메뉴를 이용한 모델링과 스크립트 언어인 파이선을 활용하여 누구라도 쉽게 실사 영상을 볼 수 있도록 하였다.

▸ Keywords : 마야, 태양계, 시뮬레이터, 파이선, 3D 애니메이션

Abstract

In this paper, we introduce a simulation tool for solar system using 3D animation tool MAYA. It accurately models solar system's astronomical features, such as each planet's orbital period, orbital speed, relative size, and texture, etc. This simulator visualize the solar system in 3D, which can be used to easily understands the system's positioning and astronomical movements. With a conventional Maya modeling process using menus and UI windows, it is difficult to assign correct physical attributes of planets. We use Python script to set up each planet's astronomical parameters. The proposed simulator is rendered as real as possible to be used for virtual reality and educational purpose.

▸ Keywords : Maya, Solar system, Simulator, Python, 3D animation

•제1저자 : 최은영 •교신저자 : 이임건
•투고일 : 2015. 1. 7, 심사일 : 2015. 1. 27, 게재확정일 : 2015. 4. 1.
* 동의대학교 디지털미디어 공학과(Graduate School of Digital Media Eng., Dongeui University)
** 동의대학교 영상정보공학과(Dept. of Visual Information Eng., Dongeui University)
※ 이 논문은 2014학년도 동의대학교 교내연구비에 의해 연구되었음(과제번호 2014AA148)

I. 서 론

3D 애니메이션은 영화와 TV등 여러 분야에서 활용된다. 1998년 출시된 마야는 3D 모델링 및 애니메이션 도구로서 널리 이용되고 있으며 특히 실사로서는 표현하기 힘들거나 카메라의 사용이 불가능한 장면들을 인위적으로 만들어 제작자의 의도를 자유자재로 표현할 수 있게 하였다. 이러한 특성으로 인해 SF 영화나 CF 영상 등에서 재난 장면, 폭파 씬 및 실사로서는 표현 불가능한 카메라 워킹 등을 손쉽게 구현하게 되었다[1].

마야는 1999년 Alias와 Wavefront 두 회사의 합병으로 Explore 와 Alias Studio의 기술에서 파생되었던 Rhinoceros 3D로 발표 된 후 점차 개선된 UI와 오차 없는 모델링이 가능한 3D 제작 도구로 발전하였다. 현재 마야 프로그램은 주로 애니메이션 제작 도구로 많이 사용되고 있으며 보다 사실적이고 복잡한 장면 구성을 만들기 위해 번거로운 수작업을 대체할 수 있도록 프로그래밍 개념이 포함되는 추세이다[2-4]. 따라서 마야는 애니메이션 제작 도구에서 애니메이션에 특화된 일종의 프로그래밍 툴로 자리 잡고 있는 실정이다. 마야에는 전용의 MEL(Maya Embedded Language) 스크립트와 범용 스크립트 언어인 파이선(Python)이 지원된다. 파이선은 범용 프로그래밍 언어로서 다양한 플랫폼을 지원하며 라이브러리가 풍부하여 다양한 분야에서 활용되고 있다. 마야와 파이선의 연결은 실제 MEL 스크립트 명령어를 파이선용으로 랩핑(wrapping)한 형태로 구현되어 있으며 마야 환경에서도 파이선의 풍부한 다른 라이브러리를 불러 쓸 수 있으므로 애니메이션의 표현력과 개별 캐릭터의 기능은 더욱 넓어졌다[3].

본 논문에서는 이러한 마야의 특성을 활용하여 실제 태양계를 최대한 사실적으로 모델링하고 이를 애니메이션을 이용하여 시각화함으로써 태양계에 대한 이해를 도울 수 있는 시뮬레이터를 제안한다. 모델링의 대상은 행성의 상대적 크기, 위치, 지표면 형태 및 공전, 자전 주기와 같은 움직임에 대한 정보 등이다. 먼저 행성의 움직임을 맞추기 위하여 모션 패스(motion path)를 이용하여 기본적인 행성의 움직임을 표현하였고, 파이선 스크립트로 정확한 주기와 공전 주기의 정보를 입력 하였다.

본 논문의 시뮬레이터는 메뉴를 이용한 마야 애니메이션으로 정확하게 나타내지 못했던 행성의 물리적인 특성을 파이선을 이용하여 정밀하고 정확하게 표현함으로써 마야가 가진 고유한 기능인 모델링 강점을 살리면서도 물리적으로 정확한 시뮬레이션을 할 수 있음을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 모델링 방식과 툴에 대한 관련 연구를 설명하고 3장에서 제안하는 태양계 모델의 자료구조와 이를 구현하는 방법 및 결과분석을 제시한다. 마지막으로 4장에서 결론과 추후 연구과제에 대해 고찰한다.

II. 관련 연구

태양계를 모델링하기 위해 본 논문에서는 마야 툴을 활용하였다. 행성의 움직임을 정교하게 나타내기 위해 그래프 편집기, 모션 패스 등 마야에서 활용 가능한 다양한 툴뿐만 아니라 스크립트 프로그래밍 언어인 파이선을 사용하였다.

1. 파이선

파이선은 컴퓨터 그래픽 사용자들이 가장 관심을 갖는 스크립트 언어이다. 마야에는 기본적으로 지원되는 MEL스크립트가 있었음에도 불구하고 파이선이 스크립트 언어로 내포된 이유는 사용자가 컴파일을 하지 않고 바로 실행할 수 있으며 한 줄 단위로 실행되어서 사용자가 쉽게 결과를 확인할 수 있다는 것뿐만 아니라 파이선에서는 다양한 라이브러리를 사용 가능하다는 점 때문이다[5]. 또한 파이선은 사용하기 쉽고 강력한 제어 기능을 제공하므로 3Ds Max, Houdini, Blender, Rhino 등 다른 3D 툴들에서도 파이선을 지원하고 있다. 마야는 8.0 버전부터 파이선을 지원하고 있다.

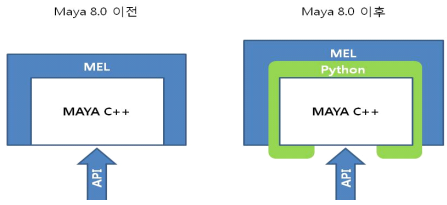

그림 1. MAYA 의 언어적 접근 구조
Fig. 1. The language structure of maya

기존에 지원되던 마야의 스크립팅 언어인 MEL은 그림 1과 같이 C++로 작성된 MAYA API 상에서 구현되어 있다. 하지만 MAYA 8.0 이후 파이선이 지원되면서 파이선은 MEL이 차지하는 영역과 C++로 접근해야 하는 API 영역까지 모두 아우르게 되었다. 따라서 파이선 만으로도 MEL이 할 수 있는 기능을 모두 구현 가능하고 마야의 플러그인까지

개발 가능하다[6]. 그러나 수행 속도가 중요한 모듈을 개발해야 하는 경우 여전히 C++로 작성된 MAYA API를 이용하는 것이 가장 빠르다[7].

2. Graph Editor

태양계의 행성을 표현할 때 가장 중요한 것이 행성의 자연스러운 움직임이다. 일반적으로 행성을 움직이고자 할 때 기본적으로 사용되는 키(key)는 Rotation 속성이지만 보다 자연스럽게 행성을 움직이기 위해서는 그림 2와 같은 Graph Editor를 이용한다. 특히 Rotation 속성으로는 대부분 타원형인 행성의 공전 궤적을 제어하기 힘들다. Graph Editor는 움직임에 감속 진입과 감속 진출의 원리를 적용함과 동시에 체공 시간을 늘리는 비교적 정밀한 시간 조절이 가능하다. 따라서 Graph editor를 사용하여 행성의 속도를 자유자재로 조절이 가능하며 자연스럽게 행성을 움직일 수 있다[1].

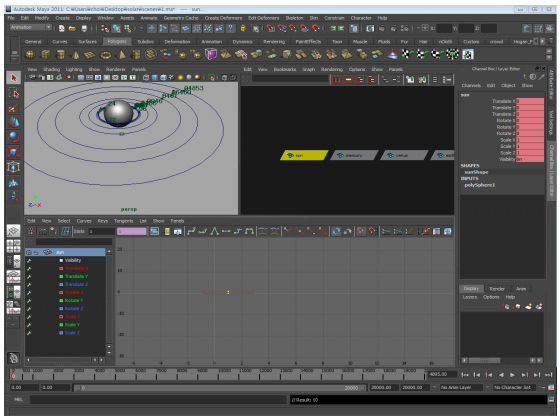

그림 2. 마야 그래프 에디터
Fig. 2. MAYA graph editor

3. Motion Path

마야 애니메이션에서 모션 패스는 일반적으로 카메라 워킹을 표현하기 위해 사용된다. 일반적으로 CVs 커브 툴을 이용하여 수작업으로 그린 궤적을 모션 패스로 설정하고 운동 궤적에 객체를 할당 하면 객체는 일정 속도로 경로를 따라 이동한다. 행성의 움직임은 케플러 제1법칙에 의해 완벽한 원이 아니라 태양을 중심으로 하는 타원궤도이므로 본 논문에서는 원일점과 근일점을 이용하여 파이선 스크립팅으로 생성시켰다. 행성의 속도는 케플러 제2법칙인 면적 속도 일정의 법칙에 따라 다음 식으로 구한다[8].

$$r_1 v_1 = r_2 v_2 = \ldots = S_1 = S_2 = \ldots = k \quad \text{(식 1)}$$

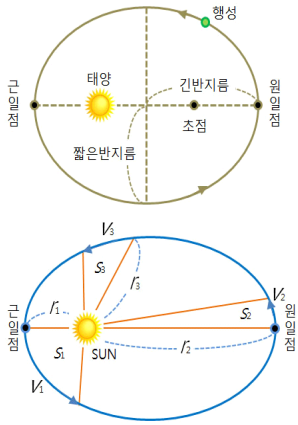

그림 3. 타원 궤도법칙과 속도 일정 법칙
Fig. 3. elliptical orbit laws and areal velocity laws

식 (1)은 속도일정법칙으로 태양과 행성을 이은 선이 같은 시간 동안 움직이며 차지하는 면적은 일정하다는 의미이다. 즉 타원궤도의 원일점과 근일점에서 행성의 운동속도가 다르다는 것으로 거리와 속도가 만드는 면적은 같다.

태양계를 모델링하는 기존의 방법은 주로 마야의 메뉴와 GUI를 이용한 것으로 시각적으로는 유사한 태양계 장면을 렌더링 할 수는 있어도 물리적으로 객관적인 정확한 모델링이 불가능하고 학문적으로 유의미한 중요성을 갖는 것으로 보기 어렵다[9]. 시각적인 애니메이션으로서의 태양계 모델링을 벗어나, 마야의 스크립팅 언어를 이용하여 물리적인 데이터에 근거한 시뮬레이션으로의 모델링이 시도되었다[10]. 그러나 이 작업은 마야에 파이선을 적용하여 태양계를 모델링의 가능성을 모색한 것으로 객관적인 데이터를 바탕으로 하지 않았다.

III. 본 론

태양계를 시뮬레이션하기 위해 객관적인 물리 데이터를 이용하여 행성을 모델링한다. 기존의 메뉴를 이용한 모델링 방식으로는 각 물리량을 정밀하게 맞추기 힘들므로 파이선을 이용하여 프로그램하였다. 가장 중요한 몇몇 태양계 각 행성의 객관적 정보를 표 1에 보였다. 이 외에도 자전속도, 이심률(eccentricity) 등의 객관적 정보를 이용하였다. 명왕성의 태양계 행성으로 인정 여부는 아직 논란이 있으며 2006년 이후 현재 행성에서 제외되어 왜행성으로 분류되어 있으므로 제외

하였다.

본 논문에서는 아래 표 1의 정보 값에 따라 태양을 포함하여 총 8개의 행성으로 실험을 진행하였다[11]. 태양을 기준으로 각 행성마다 공전 궤도 및 행성의 크기를 상대적으로 설정하였다.

표 1. 행성의 정보
Table 1. Information of the planets

행성	반지름 (km)	궤도 주기 (일/1년)	자전축 기울기 (degree)	공전 궤도 기울기 (degree)	근일점 (106km)	원일점 (106km)
수성	2,439	88.0	0.01	7.0	46.0	69.8
금성	6,052	224.7	177.4	3.4	107.5	108.9
지구	6,378	365.2	23.4	0	147.1	152.1
화성	3,396	687.0	25.2	5.1	206.6	249.2
목성	71,492	4,331	3.1	1.3	740.5	816.6
토성	60,268	10.747	26.7	2.5	1,352.6	1,514.5
천왕성	25,559	30.589	97.8	0.8	2,741.3	3,003.6
해왕성	24,764	59,800	28.3	1.8	4,444.5	4,545.7

마야에서는 파이선 언어를 지원하므로 파이선의 모든 자료구조 및 제어규칙을 사용할 수 있으며 마야의 고유 함수들은 MEL 함수들을 랩핑(wrapping)한 형태로 제공된다. 이러한 MEL 함수를 파이선에서 호출하기 위해서는 maya.cmds 모듈을 임포트해야 한다. 행성은 자료구조와 생성을 위한 메소드를 가지도록 파이선 class로 정의하였다. 행성의 모델링을 위한 클래스 정의를 표 2에 보였다.

표 2. Planet 클래스 정의
Table 2. Definition of Planet class

```
class Planet(object):
 def __init__(self, *args, **kwargs):
   self.name = kwargs.setdefault('name');
   self.radius = kwargs.setdefault('radius');
   self.rot_period = kwargs.setdefault('rotation_period');
   self.distance = kwargs.setdefault('distance');
   self.perihelion = kwargs.setdefault('perihelion');
   self.aphelion = kwargs.setdefault('aphelion');
   self.orb_period = kwargs.setdefault('orbital_period');
   self.orb_vel = kwargs.setdefault('orbital_velocity');
   self.orb_inc = kwargs.setdefault('orbital_inclination');
   self.orb_ecc= kwargs.setdefault('orbital_eccentricity');
   self.axial_tilt = kwargs.setdefault('axial_tilt');
   self.ring_inner = kwargs.setdefault('ring_inner');
   self.ring_outer = kwargs.setdefault('ring_outer');
```

공전 궤도의 기울기는 태양과 지구를 잇는 선분을 기준선으로 하여 타 행성의 공전 궤도의 기울기를 나타낸 것이므로 지구에 대한 물리량은 0도가 된다. 표 1에서의 데이터를 마야의 모델링 유닛으로 옮길 때 10^6km를 1 유닛으로 하였다. 태양의 위치를 좌표 원점으로 하여 각 행성을 배치하며 태양의 반지름은 695,500km로 행성들의 크기와는 편차가 너무 많아 마야의 뷰에 동시에 보이기 불가능하다. 따라서 본 모델링에서는 태양의 크기를 다시 1/30로 스케일링 하였다. 모든 행성은 그림 4와같이 자전축 기울기에 맞도록 기울이고 자전 방향에 맞도록 rotation시킨다. 태양계 행성의 자전의 방향은 모두 공전의 방향과 동일하게 반시계 방향이나 금성은 자전축이 거의 180도 뒤집어져 있어 자전방향이 시계방향인 것처럼 보인다. 또한 천왕성은 자전축이 거의 옆으로 누워 있다.

그림 4. 행성의 자전축 기울기와 자전 방향
Fig. 4. Axial tilt and rotation direction of planets

Planet 클래스의 개체인 각 행성을 생성한 다음 Planet 클래스에 구현한 메소드 generate()를 이용하여 행성을 생성

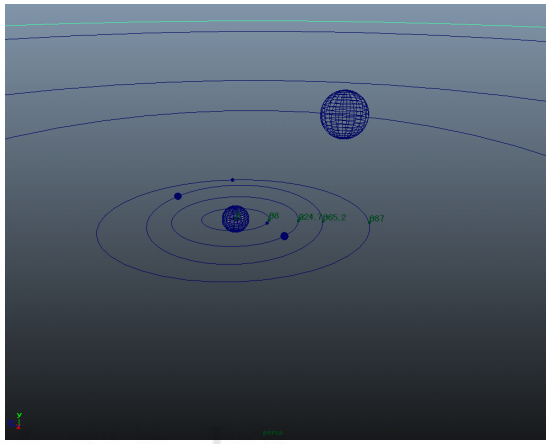

그림 5. 태양계의 메시 모델링
Fig. 5. Mesh modeling of solar system

하였다. 행성 생성 시 고리가 있으면 polyPipe() 함수를 이용하여 고리 형태를 만들고 polyUnite() 함수로 결합시켰다. 그림 5에 생성한 태양계 모델링을 보였다. 각 행성은 폴리곤 구를 이용하여 생성한다.

행성의 공전궤도는 완벽한 원이 아닌 타원 궤도를 그린다. 따라서 행성의 원일점, 근일점 그리고 이심률 정보를 이용하여 타원 궤도를 그리고 이를 경로로 하는 motion path를 적용하였다. 실제 대부분의 행성은 거의 원에 가까운 궤도를 그리며 공전하지만 정밀한 시뮬레이션을 위해 타원의 궤적을 구현하였다.

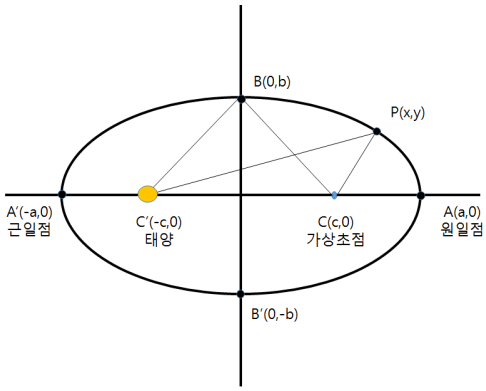

그림 6. 타원궤도에서의 원일점과 근일점
Fig. 6. Perihelion and aphelion of elliptical orbit

그림 6에서 타원의 방정식은 식 (2)로 표현 가능하며 $C'A + CA = C'B + CB$ 이므로 각 거리 a, b, c는 식 (3)과 같다.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad \text{(식 2)}$$

$$a^2 = b^2 + c^2 \qquad \text{(식 3)}$$

이심률은 장축(원일점)과 초점(태양의 위치)간의 비로 나타낼 수 있으며 식 (4)로 정의된다.

$$e = c/a = \sqrt{(a^2 - b^2)}/a \qquad \text{(식 4)}$$

마야에서는 수학 함수를 정교하게 그리는 도구가 지원되지 않는다. 즉 마야로는 파라미터로 정의되는 수학적 함수 그래프를 그릴 수 없으므로 위의 식을 이용하여 모션 패스를 타원형으로 정확하게 구현할 수 없다. 행성의 움직임에 expression을 적용하여 직접 수식으로 제어할 수 있으나 모든 행성에 독립적인 expression을 적용해야 하고 실행시간이 느려지는 단점이 있다. 따라서 본 논문에서는 정교한 타원의 궤적을 구현하기 위해 매트랩에서 함수 그래프를 구현하고 이를 마야로 import 시키는 방법을 사용하였다.

우선 매트랩에서 정교한 함수를 정의하고 각 점 데이터의 좌표값 쌍을 매트랩의 자료구조 파일인 mat 파일로 저장한다. 매트랩의 확장자 mat 파일은 파이선에서 확장모듈인 scipy.io 혹은 matlab.engine을 로딩하여 읽어 들일 수 있다. matlab.engine 모듈은 파이선에서 매트랩 연산기능을 호출할 수 있는 기능을 갖고 있으므로 마야의 파이선 모듈에서 매트랩을 직접 호출할 수 있을 것으로 생각되나 본 실험에서는 구현하지 않았다[12].

모션 패스를 적용하는 경우 시작과 끝의 시간을 설정할 수 있으므로 모션 패스 명령어인 pathAnimation() 함수의 stu(startTimeU) 및 etu(endYimeU) 플래그를 제어하여 행성의 공전주기를 설정하였다. 모션 패스는 편리한 애니메이팅 도구이지만 한 주기 이상의 애니메이션을 위해서는 경로로 지정된 개체의 키를 선택하고 setInifinity() 함수를 이용하여 키값을 cycle로 지정해야 한다. 한 주기의 전후에 관한 속성인 pre/post-infinity를 설정하면 애니메이션이 무한 반복하게 된다.

태양계의 행성 중 목성형 행성인 목성, 토성, 천왕성, 해왕성에는 고리(ring system)가 존재하지만 본 논문에서는 일반적으로 가장 잘 알려진 토성과 천왕성의 고리만을 구현하였다. 그림 7에 토성과 천왕성의 ring system을 모델링한 결과를 보였다. 천왕성 옆의 구는 해왕성이다.

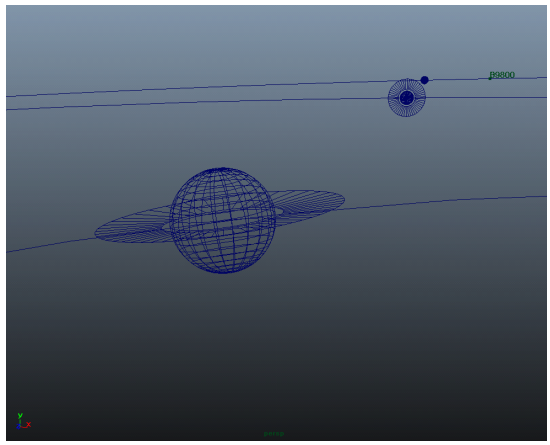

그림 7. 토성과 천왕성의 고리 구조 모델링
Fig. 7. Modeling the ring system of Saturn and Uranus

마야에서 개체에 텍스처를 입히는 작업은 shadingNode를 이용하며 영상 파일로 저장된 텍스처를 이용하기 위해 파일 노드를 생성한다. 마야의 파이선 스크립트는 노드들 사이의 연결로 프로그램을 구현하므로 먼저 shader와 file shadingNode를 생성한 다음 파일 노드의 'outColor' 속성을 shader의 'color' 속성과 연결한다. 그리고 shader의 'outColor' 속성을 set() 명령어를 이용하여 생성시킨 shading group의 'surfaceShader'에 연결한다. 그림 8은 HyperShade 윈도우에서 표현된 연결 관계를 보인 것이다.

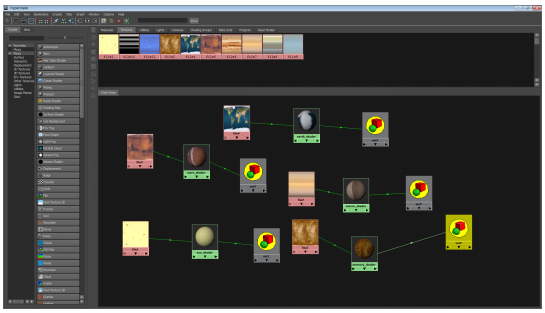

그림 8. 텍스처 처리를 위한 노드 연결
Fig. 8. Node connection for texture process

행성의 표면은 [13]에서 제공하는 행성 텍스처 맵을 이용하였으며 그림 9에 이를 보였다. 태양을 포함한 8개 행성 및 고리의 텍스처 맵을 이용하여 렌더링을 수행하였으며 그림 10은 렌더링된 지구와 목성을 보인 것이다.

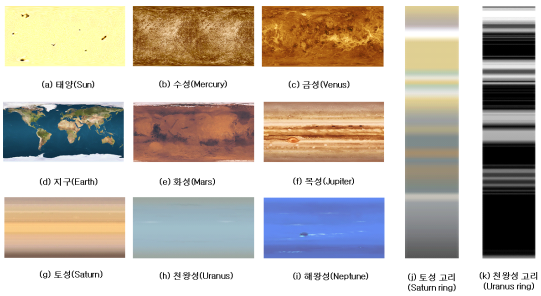

그림 9. 태양과 행성의 텍스처 맵
Fig. 9. The texture map of the sun and planets

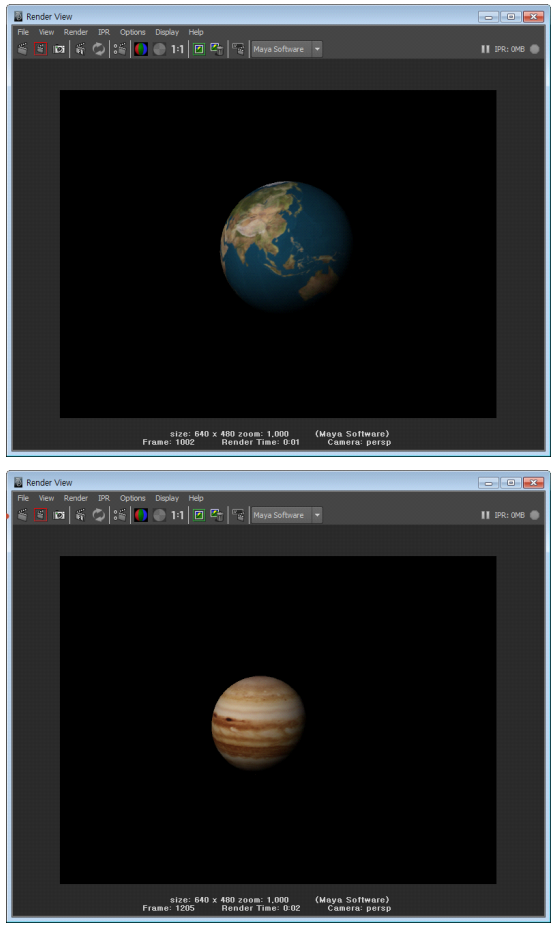

그림 10. 지구와 목성의 렌더링 장면
Fig. 10. Render result of the Earth and Jupiter

애니메이션은 전체 재생 시간을 프레임 단위가 아니라 초 단위로 설정하였다. 이는 프레임 단위의 애니메이션으로는 정밀한 물리적 속성값을 구현하기가 힘들기 때문이다. 재생시간을 설정하는 마야의 파이선 함수는 playbackOption()이며 minTime과 maxTime 속성을 이용하여 시간 지정이 가능하다. 전체 재생 시간은 100초로 하였으며 따라서 프레임 수는 2,400프레임이 된다. 그림 11은 완성된 태양계 행성 시뮬레이터를 보인 것이다.

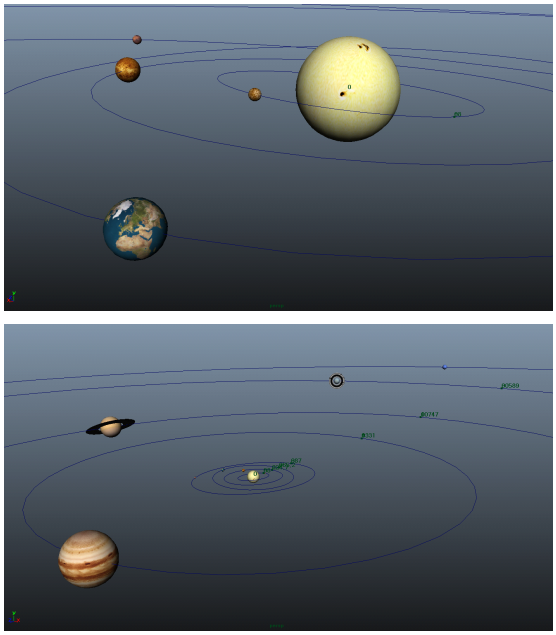

그림 11. 태양계 행성 시뮬레이터
Fig. 11. The solar system simulator

IV. 결 론

본 논문에서는 마야를 이용하여 태양계의 행성 시스템을 물리적으로 정밀하게 구현하였다. 메뉴를 이용한 기존의 마야 모델링 기법과 파이선 스크립팅을 동시에 활용하여 각 행성의 실제 물리적인 데이터들을 정확하게 나타냄으로써 마야 애니메이션으로 정확하게 표현해 내지 못하던 부분까지 정확하게 나타낼 수 있었다.

본 논문을 통해 모델링과 애니메이션에서 마야의 GUI 기반의 방식과 파이선 스크립팅을 병행하여 사용하는 것이 매우 효과적이라는 것을 알 수 있었다. 메뉴를 이용한 기존 모델링 방식은 직관적이고 쉽게 접근할 수 있지만 사용자의 수작업 기술이나 정성적인 드로잉 능력에 따라 결과물이 달라진다는 단점이 있다. 물론 속성창이나 채널 박스 등을 통해 정확한 수치 데이터를 입력할 수 있는 방법을 제공하기는 하지만 매우 번거로우며 수정 및 일괄작업이 불가능하다. MEL이나 파이선 스크립팅을 토한 공학적 접근으로는 일괄 작업이 가능하며 항상 동일한 결과물을 얻을 수 있다.

본 연구를 통해 마야 애니메이션의 생산성이나 효율성을 제고하기 위해서는 스크립팅을 더욱 적극적으로 활용해야 함을 알 수 있다. 또한 추후 파이선 스크립터를 활용한 효율적인 제작 기법에 심도 깊은 연구가 진행 되어야 할 것이다.

REFERENCES

[1] Eun-young Choi, Jun-Sang Lee, Imgeun Lee, "Simulating Solar System using MAYA Scripting". ICFICE, pp.149-152, 2014.

[2] Kim Hyun-Woo, Song Teuk-Seob, "A study of effect representation method for 3D contents development using maya system", Korea Computer Congress 2008, Vol.35, pp. 176-180, 2008.

[3] Eun-young Choi, Soojong Lee, Imgeun Lee, "Extracting Motion Information for Animation Character using Kinect Sensor" Proceedings of KSCI Conference, Vol. 21, No. 2, pp. 289-290, July, 2013.

[4] Yongwhan Lee, Changhoon Kang, Jinseob Shin, "A Study on the video production reflecting the characteristic of 3D stereoscopic", Proceedings of KSCI Conference, Vol.21 ,No 2, pp. 303-306, July, 2013.

[5] Adam Mechtley, Ryan Trowbridge, *"Maya Python for games and film : a complete reference for the Maya Python and the Maya Python API"*, Morgan Kaufmann, pp. 20-88, 2012.

[6] Han dong-il, "*Maya Python technique*", Vielbooks, pp. 224- 355, 2013.

[7] David A. D. Gould, "*Complete Maya Programming An Extensive Guide to MEL and C++ API*", Morgan Kaufmann, 2003.

[8] David P. Stern, Kepler and His Laws, *http://www. phy6.org/stargaze/Skeplaws.htm*

[9] Dariush Derakhshani , "*Introducing Autodesk Maya 2015*", pp. 14- 38, 2014.

[10] Jesse Walter, https://www.youtube.com/watch?v=EoDaFUAjHXE

[11] Planet Fact Sheet, *http://nssdc.gsfc.nasa.gov/planetary/factsheet/*

[12] MathWorks Documentation, http://kr.mathworks.com/help/matlab/matlab-

engine-for-python.html
[13] Planet Textures Map,
http://planetpixelemporium. com/planets.html

저 자 소 개

최 은 영(Eun-Young Choi)
2007년 : 동의대학교 영화영상공학과 공학사
2010년 : 동의대학교 디지털미디어 공학석사
2012년~현 재 : 동의대학교 디지털미디어 공학과 박사 과정
관심분야: 3D 애니메이션, 3D 모델링
Email : eychoi8127@naver.com

이 임 건(Imgeun Lee)
1991년 : 연세대학교 전자공학과 공학사
1993년 : 연세대학교 전자공학과 공학석사
1998년 : 연세대학교 전자공학과 공학박사
2002년~현재 : 동의대학교 영상정보공학과 교수
관심분야 : 영상복원, 영상 신호처리, 머신비젼
Email : iglee@deu.ac.kr