데스크탑에서의 OpenGL 과 Unity 3D간의 성능 비교

김민상*, 성낙준**, 최유주***, 홍 민#
*순천향대학교 컴퓨터소프트웨어공학과
**순천향대학교 컴퓨터학과
***서울미디어대학원대학교 뉴미디어학부
(ben399399, njsung, mhong)@sch.ac.kr, yjchoi@smit.ac.kr

Comparing Performance between OpenGL and Unity 3D on Desktop Environment

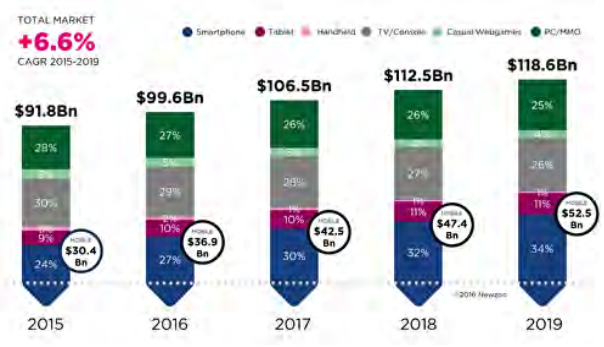
Min-Sang Kim*, Nak-Jun Sung**, Yoo-Joo Choi***, Min Hong*
*Dpet of Computer Software Engineering, Soonchunhyang University
**Dept of Computer Science, Soonchunhyang University
***Department of Newmedia Content, Seoul Media Institute of Technology

요 약

데스크탑 성능의 상향과, 그래픽 소프트웨어의 발전으로 더욱 현실적이고 자연스러운 컴퓨터 그래픽을 지원하는 게임에 대한 수요가 증가하고 있다. 3D 게임 내에서 사용자들의 게임 수행 능력과 컴퓨터 그래픽의 자연스러움은 초당 프레임 수(FPS)에 비례하므로, 더 높은 초당 프레임 수를 보장한다면 발전된 게임 수행 능력을 기대할 수 있다. 따라서 본 논문에서는 크로스 플랫폼을 지원하는 대표적인 게임 엔진인 Unity 3D와 오픈 그래픽 라이브러리인 OpenGL 간의 초당 프레임 수를 비교한다. 이를 바탕으로 추후 3D 물체에 대한 자연스러운 움직임에 대한 연구를 수행할 수 있을 것으로 예상한다.

1. 서론

최근 그래픽 하드웨어의 성능이 증가함에 따라 개인용 컴퓨터에서 현실적인 그래픽을 실시간으로 렌더링 할 수 있게 되면서 더욱 실감나는 그래픽을 가진 고사양 게임에 대한 관심이 증가하고 있다. 그림 1은 2015년부터 2019년까지의 전 세계의 게임 시장의 규모 및 예상을 나타낸다[1]. 그림 1에 따르면 2019년까지 전 세계 게임 시장의 규모는 6.6%정도의 성장률을 보일 것으로 전망하고 있다.



(그림1) 2015년~2019년의 전 세계 게임 시장의 규모

본 연구는 한국연구재단 이공학개인기초연구지원사업 기본연구지원사업(NRF-2015R1D1A1A01059304)에 의하여 수행되었음
본 연구는 순천향대학교 학술연구비 지원으로 수행하였음

통상적으로 3D 게임에서 자연스러운 화면을 사용자에게 제공하기 위해서는 초당 프레임 수(Frame Per Second : FPS)를 최소 30 이상을 유지해야 한다. 그리고 더 높은 프레임을 제공할수록 사용자는 더 나은 게임 수행 능력을 갖추므로[2], 사용자는 같은 하드웨어 자원 내에서의 더 높은 FPS를 지향한다. 따라서 본 논문에서는 사용자의 더 나은 게임 수행 능력을 제공할 수 있는 3D 게임의 제작을 위해 크로스 플랫폼을 지원하는 그래픽 라이브러리인 OpenGL과 게임 엔진인 Unity 3D와의 렌더링 및 연산에 대한 성능 비교를 수행하고자한다.

2. 본론

2.1 수치 적분(Numerical Integration)

본 논문에서 성능 비교를 위한 프로그램 상에서 움직이는 입자의 움직임을 표현하기 위해 수치적분을 이용하였다[3,4]. 수치적분이란 입자에 가해지는 힘으로부터 현재 시간으로부터 일정한 시간Δt가 지난 뒤의 입자의 가속도를 구할 수 있으며, 구한 가속도를 통해 Δt 후의 입자의 속도를 구할 수 있고, 구한 속도를 통해 Δt 후의 입자의 위치를 구하는 방법이다. 본 논문에서 사용된 수치적분법은 명시적 오일러 적분법(Explicit Euler Method)을 이용하였다. 명시적 오일러 적분법은 미지의 곡선이 주어질 때, 시작점 A_0의 위치를 바탕으로 미분 방정식에서 A_0 지점에서의 기울기가 의미하는 값인 A_0에서의 순간 속도

를 구해 최종적인 A_1의 위치를 예상하는 것이다. 본 시뮬레이션 시스템에서는 이러한 방식을 응용하여 일정한 시간의 입자의 위치에 따라 다음 시간단위 뒤의 입자의 위치를 구하는 방식을 사용하였다. 입자에 가해지는 힘은 중력가속도 G 와 입자의 질량 m의 곱으로 표현할 수 있으며 입자의 질량을 1이라 가정하면 입자의 가속도 a는 $a=\frac{Gm}{m}$ 로 표현할 수 있다. 이를 이용하여 시간에서의 x축의 속도 $v_{x,n+1}$을 구하는 식은 (1)과 같다.

$$v_{x,n+1} = v_{x,n} + \triangle t a_{x,n} \quad (1)$$

(1)에서 구한 속도 $v_{x,n+1}$을 이용하여 $n+1$에서의 x 좌표의 값 x_{n+1} 는 (2)와 같이 표현할 수 있다.

$$x_{n+1} = x_n + \triangle t v_{x,n} \quad (2)$$

2.2 테스트 환경

본 논문에서 OpenGL과 Unity 3D 엔진을 활용해 렌더링 및 연산에 대한 성능 측정을 수행할 환경은 다음 표1과 같다.

<표 1> 테스트 환경

분류	이름
CPU	Intel Core i5 4460
MainBoard	ASUSTeK COMPUTER H97M-E
BIOS	American Megatrends version 2302
RAM	Samsung DDR3 PC3-12800(800MHz) 8GB
VGA	NVIDIA GeForce GTX 750 VRAM 1024MB

본 실험을 수행하기 위해 CPU는 인텔 i5 4460모델을 사용하였다. 또한 최근 OpenGL 3.0 버전 이상을 이용하는 그래픽 프로그램들은 CPU만을 이용하여 화면에 입자를 표현하기보다는 더 많은 연산장치를 가진 GPU를 이용하여 화면에 보이는 요소들에 대한 연산을 병렬로 처리한다[5]. 이러한 GPU 기반의 병렬처리로 보다 빠르게 연산을 수행하는 방법을 지향하고 있으며, 이를 활용하기 위해 VGA로는 GLSL 4.5버전을 이용할 수 있는 NVIDIA GeForce GTX 750 모델을 사용하였다.

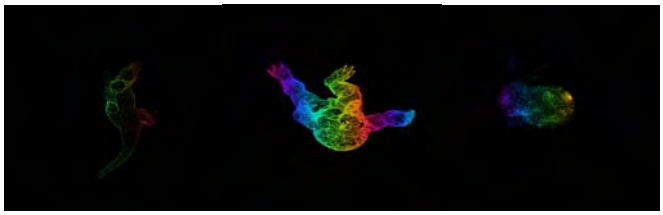
본 논문에서 사용된 소프트웨어 및 플랫폼, 라이브러리의 버전은 다음 표2와 같다.

<표 2> 실험 환경 플랫폼 및 라이브러리

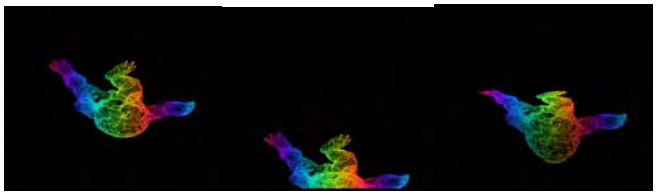
플랫폼/라이브러리	이름
Windows	Windows 10
Unity	Unity 5.4.3f1 Personal
OpenGL	4.5
GLSL	4.5

2.3 OpenGL 및 Unity 3D Particle Simulation

본 논문에서 연산속도의 비교를 위해 구현한 시뮬레이션 시스템은 각각 1,024개, 4,096개, 16,384개, 65,536개, 262,144개, 1,048,576개, 4,194,304개의 입자를 가지도록 구성하였다. 각각의 시뮬레이션 시스템은 명시적 오일러 적분법을 이용하여 자유낙하운동을 하며 바닥면에서 튕겨져 올라가는 시뮬레이션을 수행한다. 이 때, GPU 환경에서 병렬 프로그래밍을 수행하기 위해 사용된 멀티 스레드의 사이즈인 워크그룹의 크기는 (1024, 0, 0)으로 설정하였다. 다음 그림2와 같이 입자는 Vertex Shader 내부에서 Sin 함수를 이용하여 무지개색을 계산한 후 Fragment Shader를 이용해 화면에 표시하였다.



(그림 3) 성능 비교를 위한 다양한 모델



(그림 3) 성능 비교를 위한 프로그램의 실행 화면

Unity 3D 환경에서는 Unity 3D 내부에서 지원하는 Compute Shader를 이용하여 HLSL(High Level Shading Language)문법으로 작성하였다[6]. 이 때, 버퍼에 저장되는 데이터는 vector3 형식으로 두 환경에 공통으로 저장되며, 매 프레임마다의 각 입자의 위치와 색을 저장한다. 따라서 (24 * particle size) byte만큼의 메모리를 사용한다.

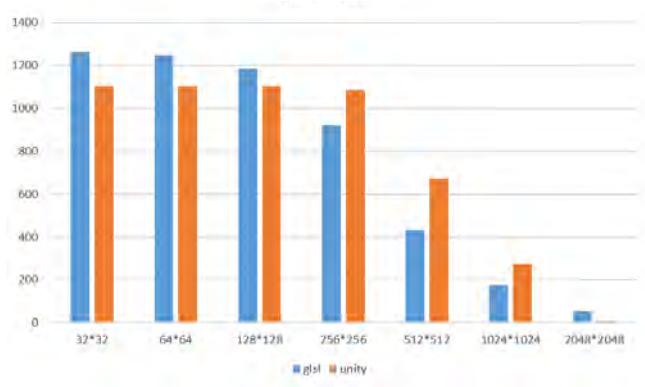
2.3 성능 비교

위의 프로그램을 통해 얻은 FPS는 매 프레임마다 약 10%의 범위 내에서 변동하므로 보다 정확한 결과를 도출하기 위해 2000프레임동안 매 프레임간의 시간의 평균을

이용하여 구하였으며, 각 입자의 횟수 별 10회씩 측정한 후 평균을 통해 구하였다.

<표 2> FPS 측정 결과

Particle Size	Platform	평균 FPS
32 * 32	GLSL	1260.426
	Unity3D	1105.888
64 * 64	GLSL	1247.141
	Unity3D	1105.366
128 * 128	GLSL	1182.364
	Unity3D	1106.346
256 * 256	GLSL	923.234
	Unity3D	1086.41
512 * 512	GLSL	432.206
	Unity3D	672.977
1024 * 1024	GLSL	175.111
	Unity3D	273.895
2048 * 2048	GLSL	55.204
	Unity3D	8.599



(그림 5) 평균 FPS 측정 결과 비교 그래프

본 연구를 통해 얻어진 그림 3과 같은 결과를 통해 같은 입자를 그릴 때, 4백만 개의 입자 수 미만에서는 Unity 3D가 약 1.56배의 성능을 보이는 것을 확인할 수 있다. Unity 3D의 경우 Compute Shader에서 연산을 수행하는 코드를 HLSL로 작성하면 엔진 자체에서 최적화된 GLSL 코드로 변환하는 과정을 거치기 때문에 기본적으로 OpenGL 4.5 버전의 GLSL을 통해 작성된 시뮬레이션 시스템보다 빠른 성능을 보인다.

3. 결론

최근 컴퓨터 하드웨어의 발전을 통해 더 자연스러운 컴퓨터 그래픽을 지원하는 게임의 수요가 증가하고 있다. 본 논문에서는 더 자연스러운 그래픽을 지원하기 위한 기본적 요소인 초당 프레임 수를 크로스 플랫폼을 지원하는 대표적인 그래픽 라이브러리 OpenGL과 물리 기반 시뮬레이션을 수행하는 3D 게임 엔진인 Unity 3D에서 구현된 물리 기반 입자 시뮬레이션을 통해 비교하였다. 본 연구의 결과 GLSL로 작성된 시뮬레이션 시스템의 FPS는 같은 입자수를 표현하는 데에 있어서 Unity 엔진 대비 63.93%의 성능을 보여준다. 추후 본 논문을 통해 얻어진 연구 결과를 바탕으로 지속적인 병렬처리 기반의 물리 시뮬레이션에 대해 연구를 수행한다면 더 빠른 연산속도를 바탕으로 3D 물체의 자연스러운 움직임을 제공하는 시뮬레이션 시스템으로 발전이 가능할 것으로 기대한다.

참고문헌

[1]Mobile to overtake PC in $99.6bn global games market - Newzoo, Lakraft, 2016

[2] Claypool, Mark, Kajal Claypool, and Feissal Damaa. "The effects of frame rate and resolution on users playing first person shooter games." Electronic Imaging 2006. International Society for Optics and Photonics, 2006.

[3] Davis, Philip. J, and Philip R, "Methods of numerical integration" Courier Corporation, 2007.

[4] Ryckaert, Jean-Paul, Giovanni Ciccotti, and Herman JC Berendsen. "Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes" Journal of Computational Physics Vol. 23, no. 3, 1977, pp. 327-341.

[5] Rost, Randi J, et al. "OpenGL shading language" Pearson Education, 2009.

[6] Unity - Manual: Compute Shaders [Internet] https://docs.unity3d.com/Manual/ComputeShaders.html