온라인 게임 클라이언트 기술 동향

김 성 용*

1. 서 론

온라인 게임은 보통 클라이언트와 서버로 이루어진다. 온라인 게임 클라이언트는 사용자가 직접 실행하는 프로그램으로서 과거 패키지 게임 개발에서부터 사용되어 왔던 기술들이 그대로 계승이 되어 그래픽스, 사운드, 인공지능, UI, 거기에 네트워크까지 일반적으로 게임 개발에 필요하다고 일컬어지는 거의 대부분의 요소 기술이 결합되어야 하는 분야이다. 온라인 게임 서버는 온라인 게임의 발달과 더불어 발전한 분야로 다수의 사용자가 게임을 즐기기 위해서 중앙에서 제공해야 하는 기능을 서비스하고, 또한 영속적 데이터 관리를 비롯한 중요한 기능을 제공한다. 서버는 많은 클라이언트의 요청을 안정적이고 빠르게 제공하는 방향으로 발전해왔고, 클라이언트는 보다 높은 품질의 그래픽과 빠른 성능을 제공하면서 보다 복잡하고 다양한 기능을 통해서 게임의 재미를 구현해내는 방향으로 발전해왔다.

클라이언트 개발은 기술적 관점에서 보면 핵심 기술 요소인 엔진을 개발하는 영역과 엔진을 이용해서 게임 콘텐츠를 구현하는 영역으로 구분할 수 있다. 최근의 게임은 대부분 3D 그래픽 기반의 게임이므로 엔진에서도 3D 그래픽을 표현하기 위한, 3D 렌더링이나 애니메이션, 지형이나 이펙트 같은 그래픽스 기술을 제공하는 것이 매우 중요하다. 주어진 엔진을 기반으로 게임 클라이언트를 개발할 때에는 목표로 하는 게임의 특성에 따라서 다양한 기술이 필요하기 마련이다. 이때 클라이언트 개발시 필요한 여러 기술들 중에서 필수적으로 고려해야 하는 것이 스크립트 기술이다. 스크립트 기술이란 간단한 스크립트 언어을 이용해서 게임의 콘텐츠를 쉽고 빠르게 개발할 수 있도록 하는 방법을 말한다. 한편, 게임 개발에 필요한 세부적인 기술적 이슈들 뿐만 아니라 개발단계에서 중요하게 여겨지는 것이 소프트웨어 개발 방법론이다. 게임 프로젝트는 다른 패키지 소프트웨어 분야에 비해서 아직도 상대적으로 체계적인 방법론보다는 수공업 형태로 개발이 이루어지는 경향이 있지만, 그럼에도 불구하고 프로젝트의 성공 가능성을 높이고 리스크를 줄이기 위해서 효과적인 방법론에 대한 필요는 꾸준히 대두되고 있다. 최근에는 게임 업계에서 애자일 소프트웨어 개발이라는 방법론에 대한 관심이 커지고 있다. 본고에서는 클라이언트 기술 동향에 대해서, 그중에서 특히 게임 엔진의 3D 그래픽스 기술 동향에 대해서 중점적으로 설명하겠다. 그리고 클라이언트 개발에 필

※ 교신저자(Corresponding Author) : 김성용, 주소 : 경기도 성남시 분당구 야탑동 367-1 (463-827), 전화 : 031)789-6614, FAX : 031)789-6599, E-mail : kimsungyong@gmail.com
* (주)제이씨엔터테인먼트 게임개발실장

요한 스크립트 기술과 최근 관심이 증대되고 있는 애자일 소프트웨어 개발에 대해서도 기술하고자 한다.

2. 게임 3D 그래픽스 기술 동향

1992년 Id Software에서 최초의 본격 3D FPS 게임인 Quake 가 출시된 이래, 게임은 3D 그래픽스 기술을 기반으로 발전해왔다[1]. 웹보드 게임과 가벼운 캐주얼 게임을 제외하고는 현재 출시되는 대부분의 게임들이 3D로 개발된다고 해도 과언이 아니다. 특히 최초로 그래픽스 하드웨어 가속 기능을 활용한 GLQuake 가 나온 이래로, 그래픽스 하드웨어의 성능과 품질이 발전함에 따라서 게임에서의 3D 기술과 그 품질도 나날이 향상되고 있다. 최긴 기술을 적절하게 활용한 Unreal Engine 3 와 같은 게임 엔진에서는 이런 그래픽스 기술과 하드웨어를 잘 활용해서 매우 사실적이면서 높은 품질의 영상을 보여준 바 있다[2]. 게임에서 이와 같은 높은 품질의 영상을 얻기 위해서는 다양한 요소 기술이 필요하지만, 여기서는 그 중에서 픽셀 라이팅(Per-Pixel Lighting)과 HDR 렌더링, 그리고 실시간 래디오시티(Real-time Radiosity)와 같은 최신 3D 게임 기술에 대해서 살펴보기로 하자. 또한 픽셀 라이팅이나 래디오시티의 개념을 응용한 것으로 게임에서 고품질 배경 렌더링에 많이 활용되는 래디오시티 노말 매핑(Radiosity Normal Mapping) 기법과 그래픽 하드웨어 활용을 극대화하고 렌더링 성능을 높이기 위해서 사용되는 지연 쉐이딩(Deferred Shading) 기법에 대해서 소개하겠다.

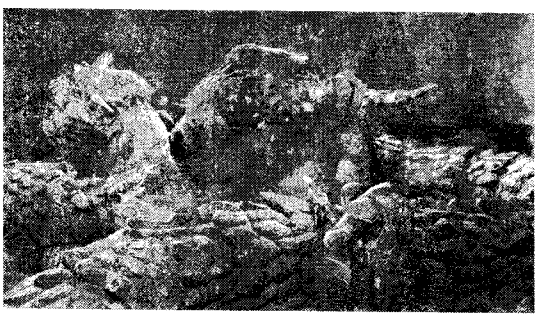


그림 1. Unreal Engine 3 로 개발된 Gears Of War 2 화면

2.1 픽셀 라이팅(Per-Pixel Lighting)

게임에서 보다 사실적이고 뛰어난 품질의 화면을 얻기 위해서는 그래픽스 하드웨어의 프로그램가능한 파이프라인(Programmable Pipeline)을 잘 활용할 수 있어야 한다. 프로그램가능한 파이프라인은 기존의 고정 기능 파이프라인(Fixed-Function Pipeline)과 달리 프로그램가능한 쉐이더 언어, 즉 버텍스 쉐이더(Vertex Shader)와 픽셀 쉐이더(Pixel Shader)를 이용하여 그래픽스 하드웨어의 기능을 이용하는 것을 말한다.

게임은 실시간으로 보다 빠르게 화면을 그리기 위해서 3D 모델의 폴리곤수에 제약이 따르기 마련이다. 이때 프로그램가능한 파이프라인을 이용하여 픽셀 라이팅(Per-Pixel Lighting)을 활용하게 되면 적은 폴리곤으로도 매우 사실적이면서 디테일한 렌더링 결과를 얻을 수 있다. 기존의 게임에서 주로 사용하는 라이팅 방식은 버텍스 단위로 라이팅을 하는 방식이다. 즉, 게임에서 주로 적용하는 라이팅식이 (식1)이라고 하면, 버텍스 단위로 이러한 라이팅 연산을 수행하고, Gouraud Shading 에 의해서 폴리곤의 각 픽셀의 값을 보간해서 모델을 그리는 방식이다. 반면, 픽셀 라이팅은 노말맵(Normal Map)을 이용하여 픽셀 단위로 라이팅 계산을 함으로써 기존 라이팅에 비해서 보다 사실적인 렌더링 결과를 얻을 수 있다. 노말맵이란 3D 모델의 각 부분에서의 법선벡터를 샘플링해서 텍스쳐 이미지에

RGB값으로 저장한 것을 말하는데, 게임에 사용하는 로우폴리곤 모델보다 디테일이 많은 하이폴리곤 모델에 의해서 노말맵을 계산한다면, 픽셀 라이팅을 이용하면 적은 폴리곤으로도 하이폴리곤 모델처럼 디테일한 렌더링 결과를 얻을 수 있다. 그림 2를 보면, 적은 폴리곤 모델에 노말맵을 적용함으로써 하이폴리곤처럼 디테일한 표현이 가능함을 볼 수 있다.

$$I = I_a + k_d(N \cdot L) + k_s(N \cdot H)^{\alpha}$$

식 1. Blinn-Phong Lighting 식

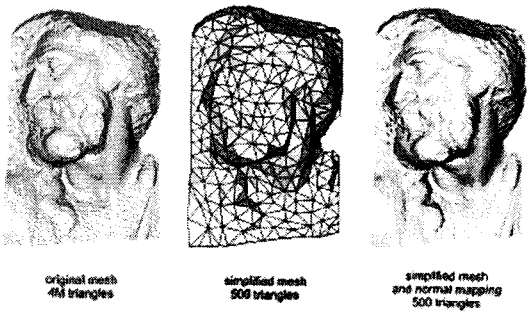


그림 2. 노말맵에 의한 세부적인 표현

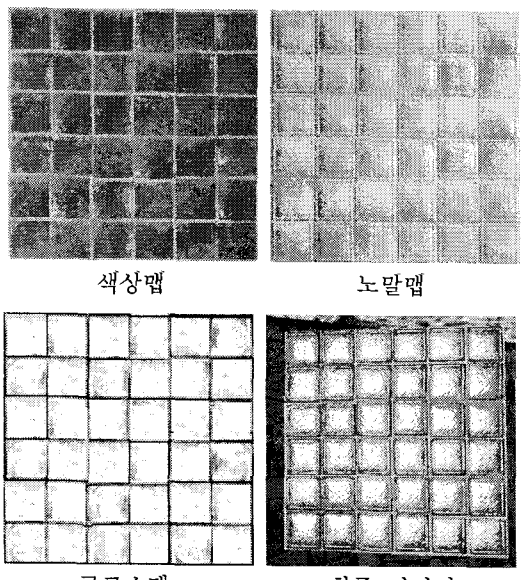


그림 3. 픽셀 라이팅 예제

모델의 각 부분에서의 법선벡터 뿐만 아니라 스페큘라 값에 대해서도 픽셀 단위로 계산을 할 수 있다. 이 스페큘라 값을 저장한 이미지를 글로스맵(Gloss Map)이라고 하며, 글로스맵을 사용하면 특정 부분만 반짝거리는 것과 같은 세부적이고 사실적인 라이팅 결과를 얻을 수 있다. 그림 3을 보면 단순한 사각형 폴리곤에 노말맵과 글로스맵(Gloss Map)을 적용하여 매우 사실적인 라이팅 결과를 얻은 것을 확인할 수 있다.

2.2 HDR 렌더링

HDR 은 High Dynamic Range 의 약자로 일반적으로 우리가 눈으로 보거나 표현할 수 있는 노출의 범위보다 훨씬 넓은 범위를 표현할 수 있는 이미지 형식 또는 그 형식을 사용한 렌더링 방법을 말한다. 이 기법은 컴퓨터로 렌더링된 이미지의 품질을 개선하기 위해 개발되었는데, 게임에서는 블룸 필터(Bloom Filter) 나 톤매핑(Tone Mapping)과 결합되어 사용되어지곤 한다. 게임에서 기존의 텍스쳐 이미지와 프레임 버퍼의 형식은 크기가 제한되어 아주 밝은 값이나 아주 어두운 값을 표현할 수 없었지만, 그래픽스 하드웨어에서 부동소숫점(floating point) 형식의 텍스쳐와 프레임 버퍼를 제공하게 됨에 따라서, 아주 밝은 값이나 아주 어두운 값까지 표현가능한 HDR 이미지를 게임에서도 사용할 수 있게 되었다. HDR 형식으로 렌더링된 영상은 블룸 필터를 거치면서 밝은 부분을 과장함으로써 더욱 밝게 보이는 착각을 불러일으킨다. 그림 4는 Unreal Engine 3를 이용한 렌더링 장면으로 블룸 필터에 의한 효과를 적용하여 밝은 주변에 빛이 번짐으로써 더욱 밝게 느껴지는 것을 확인할 수 있다. 또한 게임에 톤매핑 기법을 적용함으로써 사람의 시야가 주변 밝기에 따라서 명적응 또는 암적응 되는 것과 비슷하

그림 4. Unreal Engine 3 Bloom 예제

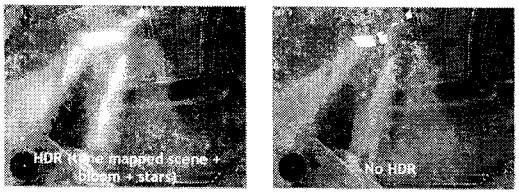


그림 5. FarCry HDR 적용예

게 노출이 조절되는 효과를 부여하기도 한다[3]. 블룸 필터는 소위 뽀사시 기법이라고 해서 많은 게임들에 사용되어 오고 있으며, 톤매핑도 일부 게임에서 사용된 바 있다. 그림 5는 Far Cry 라는 게임에서 HDR 렌더링 기법을 적용한 사례이다.

2.3 실시간 래디오시티(Real-Time Radiosity)

라이팅 품질을 높이기 위해서는 적은 수의 광원을 사용하는 것보다 많은 수의 사실적인 광원을 사용하면 당연히 라이팅 품질이 높아질 것이다. 만약 어떤 3D 모델을 렌더링할 때 실세계에서 추출하거나 혹은 보다 사실적인 글로벌 렌더링 알고리즘에 의해서 계산된 360도 방향에서 입사하는 빛의 양을 샘플링하여, 이 값에 의해서 라이팅 계산을 적용한다면 보다 사실적인 영상을 얻게 될 것이다. (여기서 모든 방향에서 입사되는 빛의 양을 샘플링한 뒤, 각 방향의 표면에서의 조사도를 구하면, 임의의 각도의 표면에서의 분산(Diffuse) 라이팅 계산이 가능하다.) 이를 위해서 저장해야 하는 것은 특정 지점에서의 조사도 분포 함수 (Irradiance Distribution Function)이며, 이는 큐브맵(Cube Map)에 저장할 수 있다[4].

최근 그래픽스 하드웨어가 발달함에 따라서, 연산 속도 뿐만 아니라 지원하는 비디오 메모리의 양도 늘어나면서 이렇듯 주변의 광원 환경을 큐브맵으로 저장한 뒤에, 이를 라이팅 연산에 사용하면 글로벌 라이팅 방법과 같은 높은 품질의 영상을 얻을 수 있다.

그러나 비디오 메모리의 제약이 있으므로 이러한 환경맵을 사용하는 것은 한계가 있다. 최근에는 구형 하모닉스(Spherical Harmonics; 이하 SH)를 이용한 압축 기술을 이용하여 실제로 큐브맵을 사용하지 않고 몇 개의 계수만으로 큐브맵을 표현함으로써 적은 메모리 용량으로 글로벌 라이팅을 적용한 것과 같은 고품질의 라이팅을 구현하기도 한다[5]. 한편 DirectX 9.0에서는 PRT (Precomputed Radiance Transfer) 라고 하는 최신 기술을 소개한 바 있다. PRT 는 실시간 래디오시티(Realtime Radiosity)라고도 불리는 기술인데, SH 압축 기술을 이용하여 3D 모델의 각 지점에서의 빛의 반사되고 투과되는 정도를 샘플링하

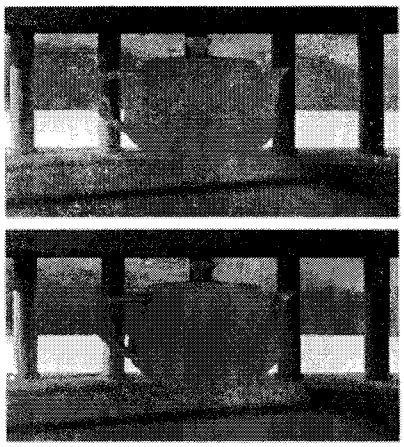


그림 6. (상) SH를 이용한 조사도 분포 함수 이용한 렌더링 (하) PRT 이용한 렌더링 (그림자까지 표현이 가능하다)

고 계산하여 분포함수를 저장한 뒤, 이 분포함수를 이용하여 실시간으로 글로벌 라이팅의 효과를 구현하는 방법이다[6].

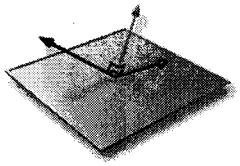
PRT 는 빛이 산란하는 효과를 비롯하여 고품질의 렌더링 결과를 얻을 수 있는 장점이 있다. 반면, 연산량이 많아서 빠른 반응 속도를 요하는 게임에는 아직 본격적으로 사용되고 있지는 않는 것으로 보인다. 향후 그래픽스 하드웨어의 품질이 향상됨에 따라서 점차 활용 빈도가 높아질 수 있을 것이다.

2.4 래디오시티 노말 매핑(Radisity Normal Mapping)

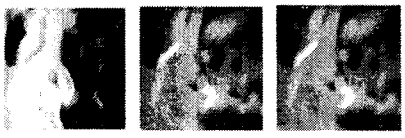
래디오시티 노말 매핑(이하 RNM)은 Half-Life 2 개발에 사용된 Source Engine 에서 처음 사용된 방법으로, 기존에 사용되어 온 래디오시티 라이트맵의 한계를 극복하여 보다 사실적인 영상을 얻을 수 있게 한다[7,8]. 이 기법은 Unreal Engine 3 에서도 적용된 바 있다. 래디오시티 라이트맵은 게임에서 정적인 배경에 대해서 사실적인 라이팅 결과를 얻기 위해서 사용하는 방법이다. 즉, 래디오시티 알고리즘과 같은 글로벌 렌더링 기법을 이용하여 배경내의 물체들에 대해서 사실적이고 높은 품질의 라이팅 결과를 미리 계산해서 이를 라이트맵이라는 텍스쳐 이미지에 저장한 뒤에, 실제 게임 화면에서는 이 라이트맵을 이용해서 보다 빠르게 실시간으로 물체의 라이팅 연산을 수행하는 방법을 말한다.

래디오시티 라이트맵을 사용하여 보다 사실적인 품질의 영상을 빠르게 렌더링할 수 있지만, 라이트맵의 해상도가 일반적으로 높지 않은 관계로 저해상도의 부드러운 그림자를 표현하기에는 적합하지만, 물체 표면의 세부적인 디테일을 표현하기는 어렵다. 라이트맵은 배경 내 모든 물체의 표면에 대응이 되어야 하기 때문에 용량을 고려하여 일반적으로 저해상도로 저장을 하며, 물체 표면의 상세한 라이팅을 표현하기 위해서 라이트맵의 해상도를 높일 경우에 텍스쳐 메모리 용량을 초과하는 문제가 발생할 수 있다. 게다가, 래디오시티 알고리즘 자체가 부드럽고 급격하게 변화하지 않는 분산된 라이팅의 표현에 더 적합하기도 하다.

RNM은 하나의 라이트맵 대신에 물체 표면상의 3 방향의 축에 대한 휘도(Radiance) 성분을 기록한 3개의 라이트맵과 한장의 노말맵을 사용하여 전체적으로 래디오시티에 의한 사실적인 라이팅을 표현하면서 물체 표면의 세부적인 디테일한 음영을 표현한다. 이때, 라이트맵은 용량이 크지 않기 때문에 3장으로 사용해도 무리가 없고, 노말맵의 경우는 디테일한 표현을 위해서 고해상도 이미지를 사용하지만 여러 폴리곤에 대해서 반복적으로 사용할 수 있으므로, 전체 텍스쳐 메모리 용량을 제어하면서 디테일한 표현이 가능하다. 아래 그림 7의 (b)는 3축 방향의 입사광을 기록한 3개의



(a) 래디오시티 노말 매핑의 3축(Basis)



(b) 3축 방향에 따른 라이트맵

(c) 최종 라이팅 결과

그림 7. 래디오시티 노말 매핑

라이트맵이다. 물체 표면의 각 픽셀에 대해서, 노말맵에 기록된 노말값을 이용하여 3개의 라이트맵에 기록된 값을 3개의 광원으로 간주하여 간단한 라이팅 연산을 수행하면 (c)와 같은 사실적이면서도 세부적인 라이팅 결과를 얻을 수 있다.

2.5 지연 쉐이딩(Deferred Shading)

그래픽스 하드웨어의 성능이 꾸준히 발전하고 더불어 그래픽스 엔진 기술도 발전함에 따라서 게임에 등장하는 배경이나 물체의 개수나 광원의 개수를 포함하여 장면의 복잡도가 빠르게 증가하고 있다. 그에 따라서 게임에서 많은 물체와 많은 광원이 들어있는 장면을 빠르고 효과적으로 표현하기 위한 여러 가지 기술들이 도입되고 있는데, 지금 소개하고자 하는 지연 쉐이딩(Deferred Shading)도 그러한 기술중 하나이다[9].

기존의 쉐이딩 방식은 물체별로 렌더링을 하는 싱글 패스 라이팅 방식과 광원의 개수만큼 렌더링 패스를 도는 멀티 패스 라이팅 방식이 있다. 이를 요약하면 다음과 같이 표현할 수 있다.

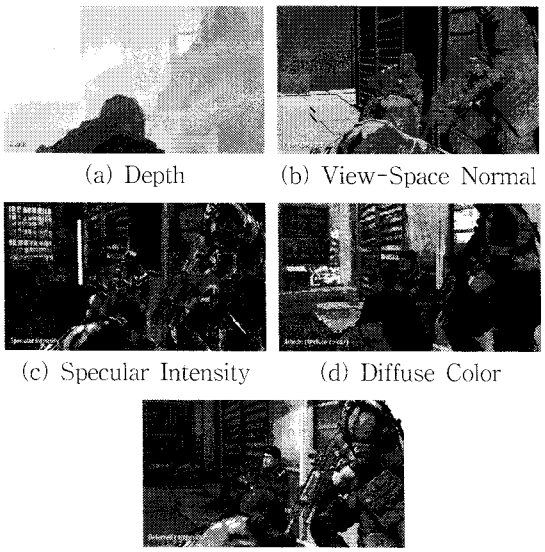
```
싱글 패스 라이팅:
  For each object
      Render mesh, applying all lights in one
      shader
멀티 패스 라이팅:
  For each light
      For each object affected by the light
          Framebuffer += object * light
```

위의 두 방식은 물체와 광원에 따라서 수행하는 패스의 개수는 다르지만, 기본적으로 물체의 기하 정보를 버텍스 버퍼로부터 직접 입력 받아서 라이팅 계산을 수행해서 프레임버퍼에 저장하는 방식이다.(Forward Shading이라고도 함) 이와 달리 지연 쉐이딩은 스크린 공간, 즉 렌더 타겟(렌더링 결과를 저장하는 텍스쳐 이미지)에서 라이팅 계산을 수행하는 방식으로, 라이팅에 필요한 모든 정보를 렌더 타겟에 저장한 다음에 라이팅을 후처리(post-processing)로 수행하는 방식이다. 요약해서 다음과 같이 표현할 수 있다.

```
지연 쉐이딩:
  For each object
      Render to render targets ( Position (or
      Depth), Normal, Color )
  For each light
      Apply light as 2D post-process
```

싱글 패스 라이팅은 광원의 수가 적은 경우에 적합하고, 광원이 많은 복잡한 장면은 대개 멀티 패스 라이팅을 사용한다. 멀티 패스 라이팅은 최악의 경우 "물체 개수 x 광원 개수" 만큼의 시간 복잡도가 발생하기 때문에 부하가 크고, 따라서 성능을 높이는 것이 중요한 이슈이다. 지연 쉐이딩의 경우는 최악의 경우에도 "물체 개수 + 광원 개수" 만큼의 시간 복잡도가 발생하므로 복잡한 장면에서 멀티 패스 라이팅에 비해서 적은 연산으로 렌더링을 수행할 수 있다. 그림 8은 킬존2라는 게임에서 지연 쉐이딩을 사용하여 렌더링한 예제이다. (a)부터 (d)는 장면속의 물체들을 각각의 렌더 타겟에 Depth, Normal, Specular Intensity, Diffuse Color를 렌더링해서 저장한 결과이며, (e)는 지연 쉐이딩을 사용하여 (a)부터 (d)까지의 결과를 입력으로 라이팅 연산을 수행해서 나온 최종 이미지이다.

지금까지 3D 게임 그래픽스의 여러 가지 기술 요소들 중에서 픽셀 라이팅, HDR 렌더링, 실시간



(a) Depth (b) View-Space Normal

(c) Specular Intensity (d) Diffuse Color

(e) 최종 렌더링 결과

그림 8. 지연 쉐이딩(Deferred Shading) 사례

래디오시티와 래디오시티 노말 매핑, 그리고 지연 쉐이딩에 대해서 소개하였다. 3D 게임을 개발하기 위해서는 이러한 세부적인 요소 기술도 중요하지만, 기존 기술이나 새로운 기술을 포함한 다양한 방법들을 얼마나 잘 조합하고 효과적으로 활용하느냐 하는 것이 사실은 가장 중요하다. 그리고, 이런 요소 기술을 얼마나 보다 적은 메모리와 보다 적은 연산량을 사용해서 구현하느냐에 따라서 게임 클라이언트의 전체적인 품질과 성능이 좌우된다고 볼 수 있다.

3. 스크립트 기술 동향

스크립트 언어는 게임을 개발함에 있어서 여러 가지 용도로 다양하게 활용 가능하다. 그중에서도 핵심적인 시스템을 C++ 프로그래머가 개발하고, 거기에 내용을 변경하거나 기존의 시스템을 활용하고 조합하여 새로운 콘텐츠를 개발하기 위한 용도로 스크립트 언어를 사용하곤 한다. 이 경우 기획자나 콘텐츠 개발자가 직접 원하는 내용을 수정하고 구현할 수 있어서 콘텐츠의 품질도 높아지고 개발 프로세스 관점에서도 유리하다. 또한 기술적으로도 모든 것을 C++로 개발하는 것에 비해서 스크립트 언어를 도입함으로써 몇 가지 이점이 있다. 첫째, 핵심 시스템과 콘텐츠가 자연스럽게 설계상 분리된다는 것이다. 따라서 개발된 코드가 보다 명확하고 응집도가 높을 가능성이 많다. 둘째, 스크립트 언어는 상대적으로 문법이 단순하고 메모리 관리나 세부적인 고려를 할 필요가 없어서 개발 생산성이 높아질 가능성이 있다. 셋째, 실행파일을 다시 컴파일하고 빌드하는 시간을 줄일 수 있다는 것이다. 그리고 마지막으로 시스템과 스크립트가 명확하게 분리된 경우, 스크립트 코드로 인해서 시스템에 치명적인 문제를 야기할 가능성이 낮아짐으로써 프로그램의 안정성이 높아질 수 있다. 즉, C++로 코드를 작성했을 경우에는 개발자의 실수로 치명적인 오류를 유발할 수 있지만, 스크립트로 개발하는 경우에는 그러한 가능성이 원천적을 차단될 수 있다는 것이다.

사실 스크립트에는 하나로 규정하기 곤란할 정도로 다양한 형태가 있다. 단순히 데이타를 외부로 추출하고, 쉽게 변경하기 위해서 txt 파일이나 xml 파일 형태의 스크립트를 사용하는 경우도 있을 수 있고, UI 의 기능을 정의하기 위해서 로직을 lua 스크립트와 같은 기존의 언어로 구현하는 경우도 있다. 한편으론 게임 엔진에서 렌더링에 필요한 머트리얼을 기술하거나 쉐이더를 기술하는 것도 크게 볼 때 스크립트라고 볼 수 있다.

게임에 스크립트를 활용하는 경우 크게 기존에 존재하는 스크립트 언어를 사용하는 경우와 자체 스크립트 언어를 만들어서 사용하는 경우가 있다. 기존에 존재하는 언어를 활용하는 경우에는 Lua 나 Python 이 대표적이다. Python 은 언어 자체가

가지는 유연성과 객체지향성과 같은 막강한 특성 그리고 C++과 쉽게 연동이 가능하다는 장점으로 인해서 스크립트 언어를 넘어서 개발을 위한 대안 언어로까지 거론되고 있는 언어이니만큼, 게임계에서도 많이 사용된다. Python 은 막강하다는 장점이 있는 반면 포함해야 하는 파일의 수도 많고 불필요한 기능이 많아서 게임에 적용하기에는 다소 무겁고 느리다는 평가가 있다. 그래서 최근에는 가벼움을 모토로 내건 Lua 언어가 게임 개발에 폭넓게 활용되고 있는 추세이다. 일례로 최근 월드 오브 워크래프트(World Of Warcraft)에서도 유저가 Lua를 이용하여 각종 유틸리티나 UI 요소를 Add-On으로 개발하여 게임에 적용할 수 있도록 하고 있다[10].

어떤 게임들은 직접 스크립트 언어를 만들어서 사용하기도 하는데 이러한 예에는 Unreal 이 있다. Unreal 은 Unreal Engine 을 이용해서 만들어졌는데, Unreal Engine 은 UnrealScript 라는 자체 제작 스크립트를 언어를 제공한다[11]. 실제로 Unreal 은 스크립트 언어가 단순히 몇 개의 정해진 기능만을 수행하는 것이 아니라 게임의 전체 코드가 UnrealScript로 만들어졌다고 할 정도로 광범위하게 사용되었으며, 스크립트가 C++ 객체와 자유롭게 연동이 가능하고 기능도 많고 강력하다는 특징이 있다.

4. 애자일 소프트웨어 개발

애자일 소프트웨어 개발은 특정 방법론 가리키는 것이 아니라, 문자 그대로 Agile(기민한, 좋은 것을 빠르고 낭비 없게 만드는) 개발을 가능하게 해주는 다양한 방법론들을 지칭하는 개념이다. 비슷한 가치를 추구하는 방법론을 기존에는 경량 방법론(lightweight method)이라고 부르곤 했으나, 2001년 애자일 동맹(Agile Alliance)의 구성원들이 애자일 방법론이라는 용어를 채택하면서 많이 사용되기 시작했다[12]. 애자일로 불리우는 방법론에는 익스트림 프로그래밍(Extreme Programming 또는 XP)과 스크럼(Scrum) 등이 있다[13]. 또한 테스트 주도 개발(TDD)과 같은 기법은 익스트림 프로그래밍에 속한 개념이기도 하면서 한편 독립적인 방법론으로 강조되기도 한다.

익스트림 프로그래밍(Extreme Programming 또는 XP)은 1990년대 후반 Kent Beck 에 의해서 제창된 이래로 관심을 받고 있는 개발방법론의 하나인데, 최근 게임업계에서도 XP에 대한 관심과 도입이 늘어나고 있다. XP에서는 개발팀이 공유해야만 하는 4가지 가치가 제시되어 있다. 그 4가지는 고객과 개발자의 또는 개발자간의 원만한 커뮤니케이션, 필요한 최소한의 설계만 하는 심플함, 빈번한 테스트에 의한 피드백, 대담한 설계 변경에 맞설 용기이다. 여기에다 경험에 의해서 제시한 구체적인 실천항목(practice)이 12개 제시되어 있다. 실천항목에는 불필요한 복잡함을 배제하는 심플 디자인, 동작을 변경하지 않고 프로그램을 다시 작성하는 리팩토링, 2명이 공동으로 코드를 작성하는 페어 프로그래밍, 소규모 배포를 계속 행하는 스몰 릴리즈, 테스트 케이스에 의해서 개발이 주도되는 테스트 주도 개발 등이 포함되어 있다. XP는 모든 항목을 반드시 지켜야 하는 것이 아니라 주어진 가치를 추구하면서 자신의 상태와 조직에 따라서 적절하게 적용하면 된다.

TDD, 즉 테스트 주도 개발의 중요성이 대두되면서 게임 개발자들도 개발 단계에서 테스트 주도 개발을 도입하거나 최소한 유닛 테스트를 작성하는 경향이 많다. 그런데, 게임 개발에서의 TDD를 적용하는 것은, 비즈니스 로직이나 여타의 분야에 비해서 까다롭다. 특히 게임 클라이언트 개발의 경우 그래픽스 하드웨어나 사운드 하드웨어와 같

이 하드웨어 장치를 사용하는 경우가 많고, 복잡한 그래픽 리소스를 사용해서 다양한 연산이 결합되어 최종 결과를 만들어 내는 경우가 많기 때문에 테스트 케이스를 만드는 것이 까다로우며, 게임 개발에 TDD를 보다 잘 적용하기 위해서는 이러한 점들을 고려해야 한다.

익스트림 프로그래밍이 원칙과 그 원칙을 적용하기 위해 실행하는 실천항목을 강조한 방법론이라면, 스크럼은 반복적인 프로세스를 강조한 방법론이다. 스크럼에서는 소규모의 스크럼팀을 구성하고, 스프린트라고 불리우는 대략 30일간의 개발 주기 동안 동작 가능한 제품을 제공하는 반복적인 프로세스를 기본으로 하며, 그외에 매일 15분 정도 회의를 한다거나, 구분없는 오픈 스페이스를 유지한다거나 하는 실천항목들을 포함한다. 스크럼은 빠른 주기로 개발을 반복적으로 수행해나가는 프로세스에 초점을 맞춘 방법론이기 때문에, 익스트림 프로그래밍의 실천 항목이나 원칙들과 결합되어 적용될 수도 있다. 즉, 기본적인 프로세스는 스크럼을 따르되 구체적인 실천 항목들은 익스트림 프로그래밍의 항목을 따르는 식으로 애자일이라는 목적을 달성하기 위해서 유연하게 적용 가능하다.

5. 결 론

본고에서는 온라인 게임 클라이언트 기술 동향을 게임 엔진에 필요한 3D 기술을 중심으로 알아보았다. 3D 기술에서 그래픽 품질을 높이기 위한 요소 기술로서 픽셀 라이팅, HDR 렌더링, 그리고 실시간 래디오시티에 대해서 요약하였고, 래디오시티 노말 매핑과 지연 쉐이딩에 대해서도 언급하였다. 3D 기술 외에 게임 콘텐츠 개발에 사용되는 스크립트 기술에 대해서도 설명하였다. 이러한 게임을 개발하는 데에는 요소 기술도 중요하지만 더욱 중요한 것은 주어진 목적을 달성하기 위해서 여러 기술들을 효과적이고 적절하게 사용하여 프로그램 전체의 성능과 품질을 높이는 것이라는 것을 잊지 말아야 할 것이다. 또한, 게임 프로그램을 개발하기 위해 활용되는 개발 방법론의 하나로 애자일 소프트웨어 개발에 대해서 소개하였다. 애자일 소프트웨어 개발은 중소규모의 팀에서 그리고, 요구사항이 자주 변경되는 경우에 기존의 무거운 방법론에 비해서 효과적이라고 알려져 있다. 이러한 특징은 게임 개발이 갖는 특징과도 잘 맞는 부분이 있어서 최근 게임 업계에서도 관심을 가지고 도입하고 있는 상황이다. 앞으로도 게임 개발에 있어서 요소 기술 자체에 대한 연구와 발전 뿐만 아니라 이러한 개발 방법론에 대해서도 지속적으로 관심을 가지고 발전시킬 필요가 있다고 본다.

참 고 문 헌

[1] http://www.idsoftware.com

[2] http://www.unrealtechnology.com

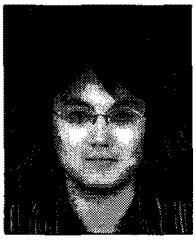
[3] Erik Reinhard, Michael Stark, Peter Shirley and James Ferwerda, "Photographic Tone Reproduction for Digital Images," SIGGRAPH 2002.

[4] Brenan, C., "Diffuse Cube Mapping," Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks, Wolfgang Engel, ed., Wordware Publishing, 2002.

[5] Green, R. Spherical Harmonics Lighting: The Gritty Details. 2003.

[6] Sloan, P., Kautz, J.,Snyder, J., Precomputed Radiance Transfer for Realtime Rendering in Dynamic, Low-frequency Lighting Environments, SIGGRAPH 2002.

[7] http://www.valvesoftware.com
[8] McTaggart, G. Half-Life 2 Soure Shading, GDC 2004.
[9] Shawn Hargreaves. Deferred Shading, GDC 2004.
[10] http://unreal.epicgames.com/UnrealScript. htm
[11] http://www.worldofwarcraft.com
[12] http://www.agilealliance.org
[13] http://www.extremeprogramming.org



김 성 용

• 1993. 3~1998. 2 KAIST 전산학과(학사)
• 1998. 3~2000. 2 KAIST 전산학과(석사)
• 2000. 2~2001. 7 (주)나래디지털엔터테인먼트 팀장
• 2001. 8~2002. 12 (주)디지털매직엔터테인먼트팀장
• 2003. 1~현재 (주)제이씨엔터테인먼트 게임개발실장
• 관심 분야 : 실시간 컴퓨터 그래픽스 기술, 개발방법론