모바일 + 3D 게임 API(MEGA)

안 정 환 삼성종합기술원 Computing Technology Lab 3D Graphics팀

모바일 + 애플리케이션 컨버전스 표준화 특집

모바일 + RFID
모바일 + 콘텐츠
모바일 + 웹(모바일 웹 2.0 포커싱)
모바일 + DMB
모바일 + TPEG
모바일 + 3D 게임
모바일 + 3D 게임 API(MEGA)

1. 서론

모바일 게임은 3차원 가속장치의 등장으로 기존의 2차원 게임 위주에서 3차원 게임으로 발전해가고 있다. 모바일용 3차원 게임제작은 2차원 게임 제작에 비해 더욱 많은 비용과 제작시간을 필요로 하며, 모바일 기기의 플랫폼이 다양화됨에 따라 특정 플랫폼을 목표로 개발된 3차원 콘텐츠를 이종의 플랫폼으로 단기간에 이식할 필요성이 커지고 있다.

모바일용 3차원 게임에서 필요로 하는 기술을 엔진화하고 표준화하면, 게임 제작사들은 이를 활용하여 쉽게 3차원 게임제작이 가능하고, 많은 모바일 게임들을 단기간에 제작할 수 있다. 또한, 서로 다른 모바일 단말기기 상에 동일한 게임 콘텐츠에 대한 호환성을 최소한의 노력으로 확보할 수 있다. 게임 데이터 포맷, API 등 다양한 수준에서의 표준화는 기존 게임 콘텐츠를 하나의 제작사 내 뿐만 아니라, 여러 제작사들 간에 공유할 수 있게 한다. 이에 따라 개발자는 게임 콘텐츠 개발 시간을 크게 단축시킴과 동시에 콘텐츠 자체의 수준을 향상시키는데에 보다 많은 시간을 할애할 수 있다. 이처럼 고품질 3D 게임 콘텐츠로 사용자를 만족시킴으로써 국내 모바일 게임의 시장 확대 및 세계 시장 확대를 기대해 볼 수 있다.

이에 모바일 3D 표준화 포럼(M3DSF)의 3D 게임엔진 워킹그룹에서는 2004년과 2005년의 표준화 요구사항 도출과 준비기간을 거쳐 2006년도에 본격적인 표준화 활동을 수행하였다. 그 결과, 모바일 3D 게임엔진 API(MEGA: Mobile 3D Game API) 1.0, MEGA 적합성 시험도구 표준 등 포럼 표준안을 도출하였으며, TTA에 제안하였다. 그리고, MEGA 1.0은 WIPI 표준으로 제안할 예정이다.

이 작업에 주로 참여한 기관으로는 한국전자통신연구원(ETRI), 삼성종합기술원(SAIT), (주)이머시스, (주)넥서스

칩스, (주)놀이즌, (주)신지소프트, (주)레인콤, (주)펫헤머 등이 있다.

2. Mobile 3D Game API

MEGA 1.0은 크게 장면 부(scene part), 렌더링 부(rendering part), 애니메이션 부(animation part)의 3가지 부분으로 이루어져 있다. 장면 부는 하나의 장면을 가지고 있는 월드 시스템(world system)과 월드를 구성하는 클럼프 시스템(clump system)으로 구성되며, 3D 데이터를 그리는 기능을 제공하는 렌더링 부는 3D 객체인 쉐이프(Shape), 기하정보(Geometry), 텍스처(Texture), 라이트(Light) 등의 구조를 가지고 있다. 애니메이션 부는 Skinning, 키 프레임 애니메이션, 모핑(Morphing), 애니메이션 블렌딩을 지원한다. 그리고, 3D 모델을 전송 및 저장하는데 있어 압축 표준 기술인 MPEG-4 AFX (Animated Framework eXtension)에 정의된 Core 3D Compression Profile을 만족하는 '.m3d' 포맷의 압축 파일을 읽을 수 있다.

그림 1은 MEGA API 구성도이다. 게임을 구성하는 장면 부, 애니메이션에 관련된 애니메이션 부, 렌더링에 관련된 그래픽스 부로 구성된다. 그림에서 회색으로 표현된 부분은 MEGA 1.0에서 정의하지 않은 부분으로 차기 버전에서 지원할 예정이다.

장면은 크게 하나의 장면을 가지고 있는 월드와 월드를 구성하는 클럼프들로 구성된다. 클럼프는 기하(geometrical) 데이터의 계층적인 구조를 표현하고 월드는 위치(locality)나 가시성(visibility)을 최적화하는 구조를 가진다.

애니메이션은 클럼프가 애니메이션할 수 있도록 애니메이션의 키 프레임을 관리하고, 서로 다른 키 사이의 값을 보간하거나 블렌딩(blend)한다.

렌더링은 3D 데이터를 그리는 기능을 제공하는 시스템

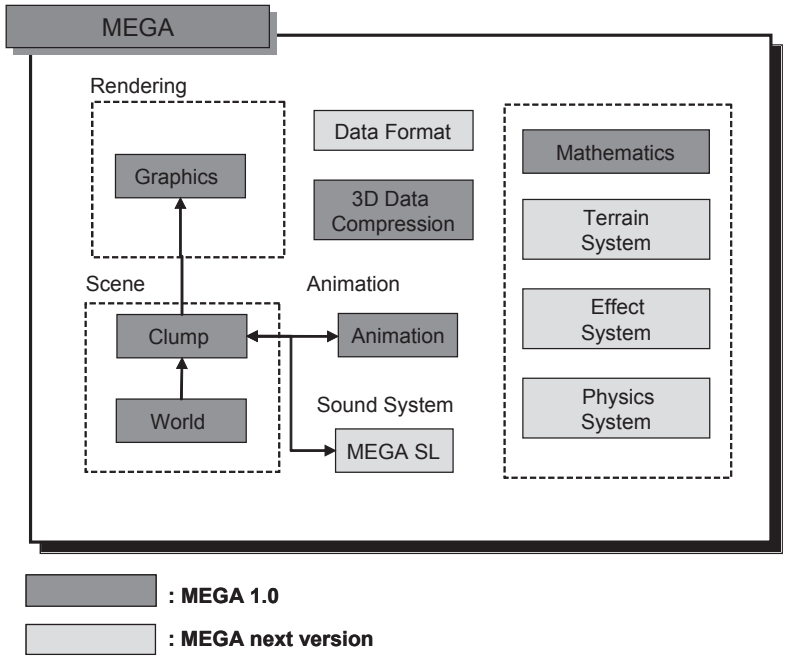


그림 1. MEGA 1.0 구조

으로 그래픽스 하드웨어나 OpenGL®, OpenGL ES®, Microsoft® Direct3D®, Direct 3D Mobile(D3DM)®과 같은 Low-level graphics API와는 독립적인 구조이다. 여기에는 3D 객체인 쉐이프(Shape), 기하(Geometry), 텍스처(Texture), 라이트(Light) 등과 같은 구조를 가지고 있다.

2.1 장면 부

장변 부는 크게 2가지 부분으로 구성된다.

- 월드 시스템: 게임과 같은 애플리케이션에서는 큰 규모의 배경을 바탕으로 많은 객체들로 구성되는 것이 일반적이다. 게임의 성능을 높이면서 다수의 기하 모델을 효과적으로 관리하기 위한 시스템으로, 주로 큰 규모의 월드 상에서 게임 진행 및 화면 출력을 위해서 필요로 하는 쿼리(Query) 처리를 빠르게 하기 위한 시스템이다.
- 클럼프 시스템: 클럼프란 계층적 기하모델을 만들거나 처리하기 위한 일종의 컨테이너로 애니메이션이나 다른 클럼프와의 연결에도 사용된다.

2.1.1 월드 시스템

월드는 게임에 사용되는 객체들을 가지고 있는 구조로 빠른 쿼리를 위해 특별한 데이터를 가지고 있다. 보통 쿼리에는 가시성 쿼리와 위치 쿼리가 있다. 가시성 쿼리는 현재 카메라 시점에서 보이는 객체에 대한 리스트(list)를 얻기 위한 것이고, 위치 쿼리는 현재 임의의 위치에서 관심있는 주변 물체에 대한 리스트(list)를 얻기 위한 것이다. 이렇게 월드와 객체로만 구성되어 있는 대규모의 게임이나 일반 3D 응용에서 언급한 쿼리 처리를 위해서는 많은 연산량이 필요하다. 따라서, 월드를 몇 개의 구역(world sector)으로 나누어 관리하고, 일차적으로 그 영역 자체에 대해서 쿼리를 수행한다면 연산량을 상당히 줄일 수 일다.

- 월드는 어떤 장면을 표현하는 주 클래스이고 이것은 여러 개의 객체들을 가지고 있다. 월드를 이루는 객체는 다음과 같다.
- ClumpObject – 기하(geometrical) 객체
- DynamicClumpObject – update() 함수가 있는 기하 객체
- LightObject – 광원(light source) 객체
- CameraObject – 카메라(camera) 객체

MEGA 1.0에서 월드는 2D 혹은 3D의 복셀 격자(voxel-grid)의 구조를 가지고 있다. 그림 2는 월드와 객체의 관계를 그림으로 표현한 것이다

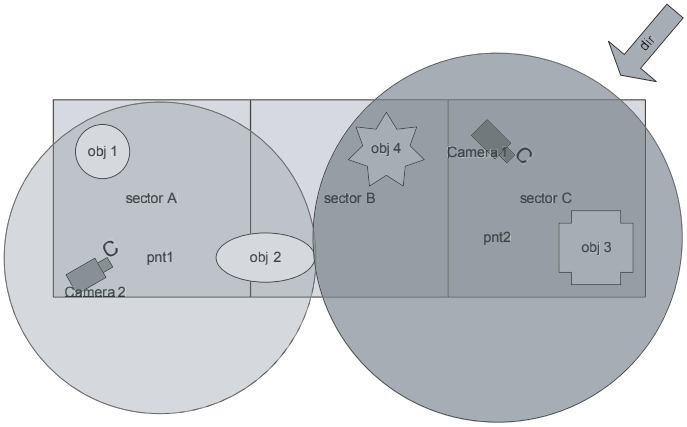


그림 2. 월드와 객체의 관계

월드는 크게 3개의 구역(A, B, C)로 나눠지며, 2개의 카메라(camera1, camera 2), 4개의 기하 객체(obj 1, obj 2, obj 3, obj 4), 2개의 포인트(point) 라이트(pnt1, pnt2), 한 개의 디렉셔널(directional) 라이트(dir)로 구성되어 있다. 이런 경우에 대한 월드의 구조는 다음과 같다.

```
World
{
    WorldSectors [sector A, sector B, sector C]
    Lights [dir]
}

WorldSector “sector A”
{
    WorldObject [obj 1, obj 2, Camera 2]
    Lights [pnt1]
}

WorldSector “sector B”
{
    WorldObject [obj 2, obj 4]
    Lights [pnt1, pnt2]
}

WorldSector “sector C”
{
    WorldObject [obj 3, Camera 1]
    Lights [pnt1, pnt2]
}
```

월드는 크게 리스트 월드(List world)와 복셀(Voxel) 월드로 나뉜다. 리스트 월드는 단순하게 객체들을 저장하고 있는 구조로, 쿼리를 위한 특별히 최적화된 알고리즘이 없다. 복셀 월드는 쿼리를 위한 복셀로된 구조를 가지고 있다. 즉, 객체에 대한 복셀 월드를 생성하고 라이트를 위한 복셀 월드를 생성할 수도 있다. 일반적으로 라이트를 위한 복셀은 객체를 위한 복셀보다 크기 때문에 라이트에 대한 복셀을 생성하면 쿼리를 더 빠르게 할 수 있다.

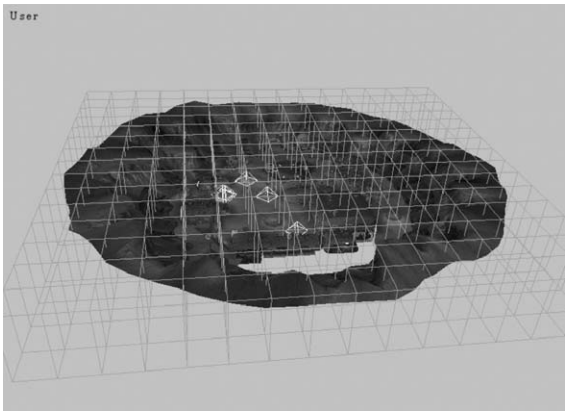


그림 3. 복셀 월드

2.1.2 클럼프 시스템

클럼프는 게임을 구성하는 기본 객체이다. 즉, 계층적 구조를 갖는 기하 모델을 만들거나 처리하기 위한 구조로, 그림 2에서 월드를 구성하는 WorldObject 중 기하정보 객체의 그룹 개념이 클럼프다. 또한, 애니메이션이나 다른 클럼프와의 연결에 사용되는 기능을 지원한다.

클럼프는 Transform과 Bounding Volume을 가지고 있다. Transform은 그림 4와 같이 계층(Hierarchy) 구조를 표현할 수 있으며, 3가지 종류가 있다.

- ShapeTransform: 기하값과 바운딩 볼륨(Bounding Volume)을 갖는다.
- BoneTransform: Skinned 애니메이션을 위해 사용되는 Bone 정보를 갖는다.
- HelperTransform: 클럼프 계층 구조 내 행렬에 접근할 수 있다. 만약, 사람이라는 클럼프 구조에서 오른쪽 손에 카메라 객체를 연결하는 경우 Helper Transform을 사용한다.

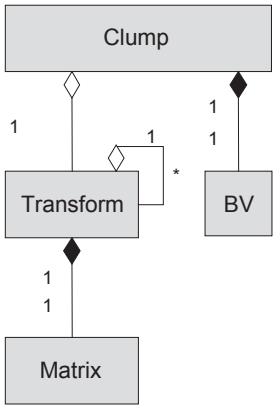


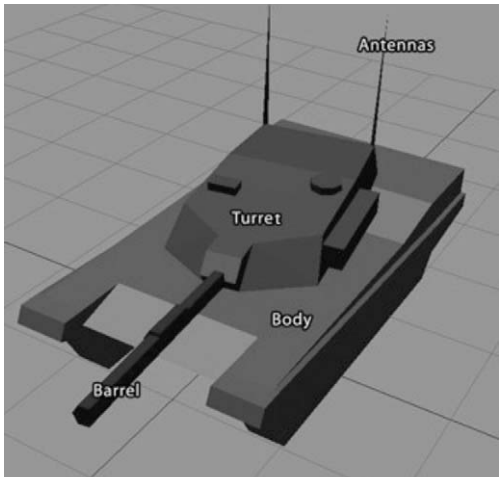
그림 4. Clump의 구성

그림 5는 Tank라는 클럼프를 구성한 예이다.

Clump는 그림 6과 같이 AnimatedClump, Skinned Clump로 확장될 수 있으며, AnimatedClump는 키 프레임 애니메이션이나 모핑(morphing)을 위해 사용되며 SkinnedClump는 Skinned 애니메이션을 지원하기 위해 사용된다.

2.2 렌더링 부

쉐이프는 렌더링되는 값을 가지고 있으며, Geometry



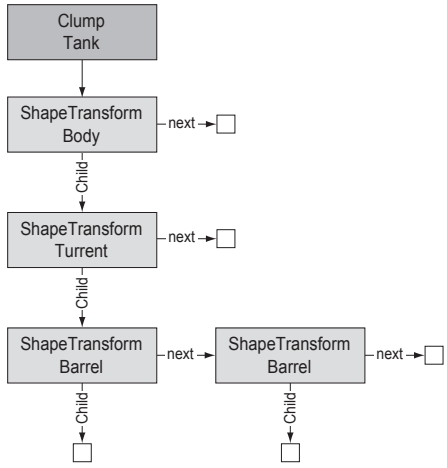


그림 5. Tank Clump의 예

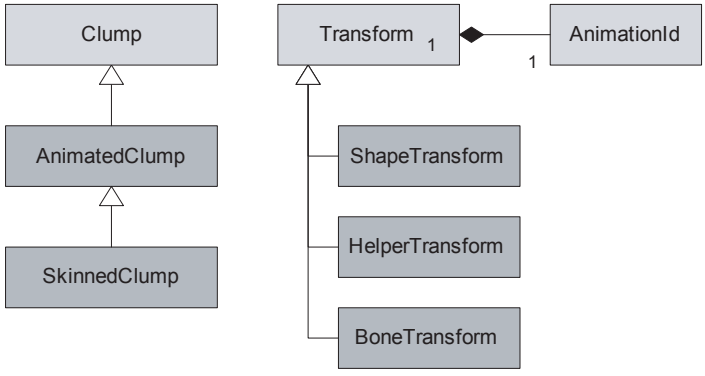


그림 6. Clump의 종류

와 Appearance의 포인터를 저장한다. 따라서, 2개의 쉐이프가 서로 다른 Appearance를 갖고 있지만 하나의 Geometry를 갖고 있는 모델에 대한 표현도 가능하다.

Geometry는 Vertex들의 집합으로 Appearance와 함께 쉐이프를 만들며, 게임 엔진에서 렌더링 단위가 된다.

Appearance는 렌더링되는 동안 화면에서 geometry가 어떻게 보이는지에 대한 정보 material, texture 등을 가지고 있다. Material은 diffuse reflection, ambient reflection, light emission, specular highlight 특성을 포함하는 표면의 빛에 대한 속성값을 가지고 있다. Appearance에 사용되는 texture를 더하거나 변경하기 위해서는 setTexture()를 사용한다.

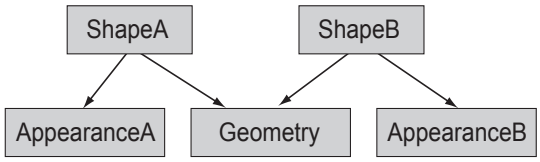


그림 7. 쉐이프 구성

쉐이프를 렌더링하기 위해 월드 좌표계 상에서 쉐이프에 대한 position과 orientation을 지정하기 위한 행렬을 지정한다. 그리고, Graphics instance의 renderShape을 호출하여 렌더링을 수행한다.

LightPack은 실제로 쉐이프에 영향을 주는 라이트의 묶음(pack)이다. 이것은 createFrameLightPack에 의해서 생성된다.

Light는 Light 클래스로 구현되며, directional, point, spot을 지원한다.

Viewport는 다음의 특성을 갖고 있다.

- Viewport의 Position과 크기
- color buffer, z-buffer, stencil buffer의 clear를 위한 파라미터
- viewport를 사용하는 카메라

Viewport의 주 기능은 Viewport::internalRenderWorld로 mgGraphics3D::beginScene와 mgGraphics3D::endScene 사이에 Viewport::renderWorld를 호출한다. 그러면 Viewport::renderWorld는 다음의 일을 수행한다.

- Clear 파라미터를 이용하여 viewport를 clear한다.
- 카메라를 그래픽스에 부여한다.
- 월드에 있는 모든 객체를 렌더링한다.

2.3 애니메이션 부

애니메이션 시스템은 프레임 관리자에 기반을 두고 있다. 프레임 관리자는 애니메이션할 데이터를 저장하고 있어, 키 프레임 애니메이션을 관리하거나, 키 사이의 보간, 블렌딩(blending) 등을 처리할 수 있는 구조로, 애니메이션 데이터를 내부에 갖고 있다.

보간을 위한 애니메이션 데이터 구조는 그림 8로 클럼프에 대한 애니메이션 시퀀스 L개가 있고, 각 애니메이션 시퀀스를 이루는 Transform이 K개 있을 때 하나의 Transform에 대한 animation key값들은 N개가 존재하는 예를 보여준다. 이렇게 프레임 관리자는 애니메이션에 필요한 데이터를 가지고 애니메이션 값을 계산하여 전달해 준다.

따라서, 서로 다른 애니메이션 값의 블렌딩, 모핑 애니메이션, Skinned 애니메이션을 지원한다.

3. 결론

모바일 3D 표준화 포럼의 3D 게임엔진 분과에서는 국내 3D 게임의 시장 활성화와 세계 시장 선도를 목적으로 MEGA(mobile 3D game API) 1.0을 개발하였다. 이 작업에는 다수의 국내 3D 게임개발사 및 삼성종합기술원(SAIT)과 한국전자통신연구원(ETRI) 등의 연구기관이 참여하였

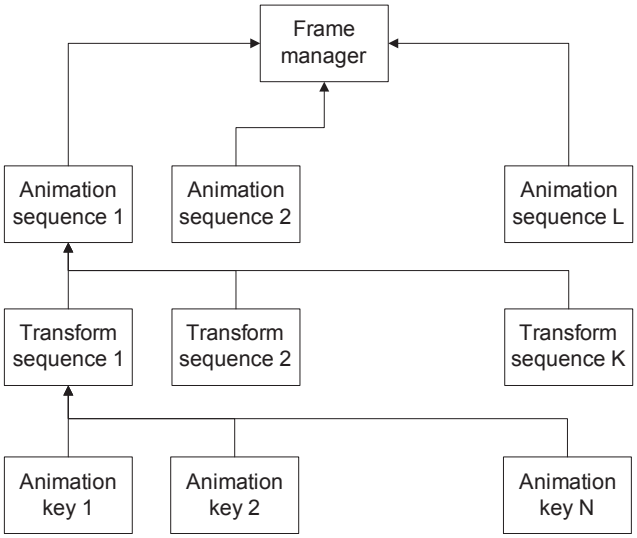
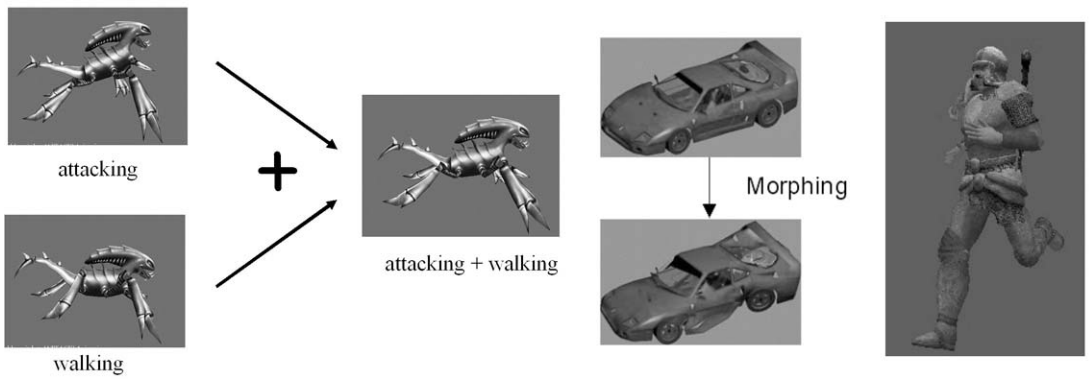


그림 8. 애니메이션 데이터 구조



(a) 애니메이션 블렌딩 (b) 모핑 애니메이션 (c) Skinned 애니메이션

그림 9. 지원하는 애니메이션

다. 3D 게임엔진 분과의 TFT(task force team)는 MEGA의 향후 버전에 네트워크 엔진, 물리엔진, 인공지능 엔진 등을 점차적으로 추가하여 개발자들이 사용하기 편리한 국내 표준 게임엔진 API로서 MEGA를 발전시킬 계획이다. TTA