텍스처 분석 테이블을 이용한 3D 지형 객체 자동 생성

선영범, 김태용, 이원형
중앙대학교 첨단영상대학원 영상공학과

Automatic 3D Map-Object Generation Using Texture Analysis Table

Young-Bum Sun, Tae-Yong Kim, Won-Hyung Lee
Dept. of Image Engineering
Graduate School of Advanced Imaging Science, Multimedia & Film
Chung-Ang University

요 약

본 논문은 지형중심 게임에서 깊이레벨에 기반한 텍스처 분석 테이블(TAT)을 이용하여 높이에 따라 정의된 지형 객체들을 효율적으로 생성 시킬 수 있는 알고리즘을 제안한다. 기존의 방법에서는 맵에디터 상에서 지형의 텍스처와 지형의 사실적 표현을 위해 나무나 바위 등의 지형 객체를 수작업으로 편집하였는데 제안한 알고리즘을 적용하면 깊이 단계별 최소의 지형 텍스처만을 사용하여 매우 다양한 종류의 지형 텍스처를 생성해 낼 수 있으며, TAT로부터 깊이 정보값을 활용하여 자연-객체들(Natural Object)을 자동으로 생성시킬 수 있다. 이로써 게임 지형을 제작하는데 불필요한 작업량을 줄일 수 있으며, 그만큼 인공-객체들(Artificial Obejct)을 생성하는데 많은 시간을 투입할 수 있다.

1. 서론

게임은 많은 사람들에게 오락의 하나로 발전해 왔으며 현대에 이르러서는 컴퓨터를 이용한 게임의 등장과 더불어 컴퓨터와 사람의 대결이라는 시나리오로 발전해 왔다. 또한 21세기 미래 핵심산업의 중심이 될 것이라고 예견하고 있으며, Microsoft, SGI, AT&T, IBM 등 과거에는 게임과 아무런 관련이 없던 거대 통신분야의 회사들이 전략적 제휴를 통해 게임 산업에 속속 참여하고 있다. 한편 헐리우드의 제작자들도 실리콘밸리의 회사들과 연계를 통하여 미래의 상호작용 게임(Interactive Game) 및 가상현실 기술을 응용한 첨단 게임을 준비하고 있다.

현재 온라인 게임은 상호 작용성, 익명성, 오프라인에 비해 높은 접근성, 내용 확장성등 다른 장르에 비해 게임 콘텐츠가 고급화 되고 있으며, 멀티플레이 사용자 수의 대규모화로 인해 가상세계의 규모도 매우 커지고 있다.

본 연구는 교육부의 BK21 사업의 지원으로 수행된 결과의 일부임.

초기 온라인 게임은 16명 이하로 제한될 정도로 동시 참여율이 매우 떨어졌으나 네트워크의 환경과 게임 개발 기술의 발전으로 지금은 대규모 멀티플레이 온라인 게임(MMPOG)의 형태를 취하고 있다. 이러한 MMPOG 형태의 게임은 거대한 가상 세계의 대륙을 기반으로 많은 게임 유저들이 활동하고 있으며 그 곳에서 단지 적을 무찌르는 즐거움만을 얻는 것이 아닌 현실세계에서 느낄 수 있는 소속감과 전투를 제외하고도 연애나 혈맹, 상업, 정치 활동을 통해 다양한 희노애락을 느낄 수 있도록 구현되어 있다. 이러한 가상환경을 구축하기 위해서는 네트워크 기술과 거대한 대륙의 표현 및 분할 방법이 필요하다. 하지만 아직 많은 부분이 해결되지 않고 있다. 기존의 지형 생성 방법을 보면 2D의 경우, 순환 텍스처를 그래픽 디자이너가 직접 제작하고 있으며, 이러한 수동식 제작방법으로 인해 텍스처 추가시마다 8방향을 고려하여 최대 8^n 개의 텍스처를 제작해야 하는 어려움을 안고 있다. 이것은 여전히 해결되지 않은 부분이며, 예외없이 3D 상에서도 이러한 문제가 적용되고 있다.

본 논문에서 제안한 방법은 이러한 근본적인 지형 텍스

처 표현 방법을 바꾼 것으로 단순히 만들어진 지형 텍스처와 보간 텍스처를 이용하여 지형의 재질을 표현하는 것이 아니라 자동으로 같은 텍스처를 출력하는 각 타일도 서로 다르게 표현할 수 있으며, 서로 다른 텍스처를 가진 타일간에도 보간된 형태의 순환타일을 자동으로 생성하고 깊이레벨 정보를 활용하여 동일텍스처의 반복사용을 피할 수 있다. 또한 TAT에서 깊이정보를 추출하여 텍스처 종류에 따라 자동으로 자연 객체를 생성할 수 있도록 제안하였다. 본 논문의 구성은 다음과 같다. 먼저, 2장에서는 기존 지형 객체 생성 방법의 문제점을 제시하며, 3장에서는 텍스처 분석 테이블을 이용한 지형 객체 자동생성 방법을 제안한다. 마지막으로 4장에서는 결론 및 향후의 연구과제를 제시한다.

2. 기존 3D 지형 객체 생성방법의 문제점

2.1 BSP/CSG 기반 방식

공간 분할 방법의 대표적인 게임은 퀘이크로써 매우 효율적인 지형 분할 및 랜더링기법을 제공하고 있다. 하지만 이러한 방법은 실내나 던전과 같은 시야가 많이 가려질 수 있는 곳에서만 통용되는 방법이며 온라인 게임과 같은 실외가 주가 되는 게임에서는 부적당하다.

2.2 Height-Map 기반 방식

하이트맵 기반 방식은 높이값을 가진 2차원 배열정보를 응용한 것으로 하이트맵 방식의 몇가지 문제점을 보안한 Multi-Height-Map방식도 많이 사용되고 있다. 이러한 방식은 실내보다는 실외 맵을 생성하는데 더욱 높은 효율을 제공한다. 기존의 방법에서는 그리드에 하나의 텍스처를 매핑시키기 때문에 일정 높이에서 다른 텍스처를 매핑하는 과정에서 연결이 자연스럽지 못하며 격자 형태의 지형 텍스처를 얻게된다.

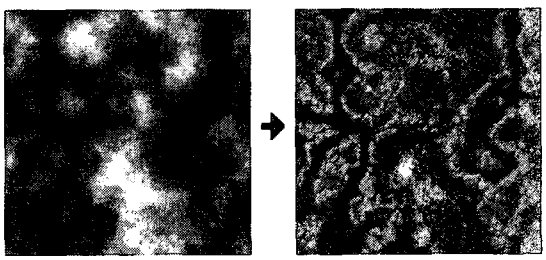

그림 1. Procedural Texture Generation을 이용한 텍스처맵

그림 1은 높이에 따라 모래(0~64), 초원(64-128), 암벽(128~192), 눈(192-256)의 텍스처를 선정하여 하이트맵을 참조하여 오버랩이 될 수 있도록 생성한 맵이다. 그림 3은 이러한 맵을 Height-Map에 Planar Mapping을 하여 다른각도로 본 영상이다.

그림 2. Planer Mapping 결과이미지

이러한 매핑 방법의 단점은 단계별로 정해진 텍스처의 이미지를 사용하므로 텍스처의 개수를 제한했을 경우 텍스처의 연속적 사용이 일어날 수 밖에 없으며, 이러한 반복 사용으로 사실감 감소 및 플레이어에게 지루함을 줄 수 있게 된다. 만약 이러한 순환타일 사용에서 그리드의 크기가 작을 경우 타일형태가 쉽게 드러나므로 지형텍스처의 패턴화 현상이 발생할 수 있으며 플레이어에게 몰입감을 떨어뜨리는 요인을 줄 수 있다. 반대로 그리드가 너무 커지게 되면 텍스처가 블랜딩되어 지형 재질이 뚜렷히 보이지 않는 결과를 얻게 된다.

3. TAT를 이용한 지형객체 자동생성 제안

3.1 높이값에 따라 각각의 텍스처 획득

하이트 맵 기반에서 단계별로 텍스처가 크게 바뀌는 부분에 대해 여러개의 텍스처를 준비한다.

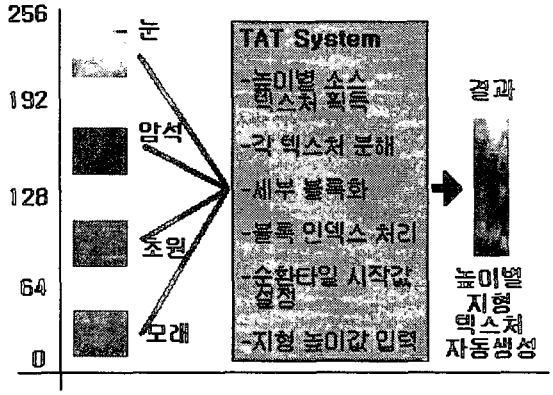

그림 3. 제안한 방법의 전체 흐름도

본 논문에서는 그림 3와 같이 단계별로 텍스처로 설정하였다. 기존 방법과는 다르게 소스 텍스처들은 디자이너의 별도의 수작업이 필요없이 자연영상의 일부를 취득한 것으로 순환 텍스처로 만들 필요가 없다.

3.2 각 텍스처별 순환 초기값 화소 설정

순환 텍스처의 특징인 Left X 화소들과 Right X 화소들이 같다는 특징을 활용한다. Y축 화소들 역시 마찬가지다. 따라서 각 높이별 소스 텍스처들의 시작 X, Y축 화소들 값을 고정시켜서 마지막 블록은 첫 블록 텍스처와 유사도값이 가장 높을 블록으로 선정하면 순환 텍스처를 생성 시킬 수 있다.

3.3 Texture Analysis Table 생성

TAT는 각 소스 텍스처를 다시 작은 블록으로 분해해서 서로간의 연결성이 높은 블록간의 정보를 인덱스한 테이블이다. TAT 생성은 그림 4와 같이 소스 텍스처에 임의의 블록을 선정하여 i, j로 1 Pixel 씩 증가하여 TAT_{DB}를 구축한다.

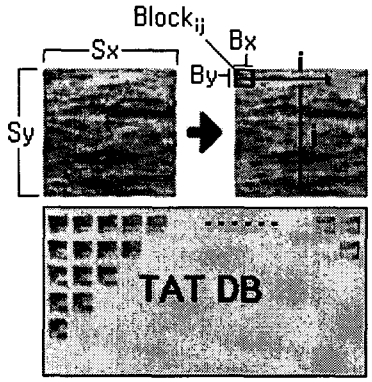

그림 4. TAT DB 생성

TAT_{xy}에는 B_{xy} 정보가 매핑되며, B_{xy}의 범위는 (1)식과 같다.

$$TAT_{DB} = \{ TAT_{ij} = Block_{ij} \mid 0 \le i < Sx-Bx,\ 0 \le j < Sy-By \}$$
$$i,j \in \text{Integer}$$

여기서 TAT_{ij} = *Texture Analysis Table Array,*
$Block_{ij}$ = *Block Texture Start Point*
Sx,Sy = *Texture Size*
Bx,By = *Block Size* (1)

(1)식에 의해 TAT DB는 한 텍스처에 대해 (SX-BX) * (SY-BY)개의 작은 블록텍스처를 포함하게 되며, n개의 소스 텍스처 개수 만큼 수행하므로 총 (SX-BX) * (SY-BY) * n * BY * BX 의 텍스처 블록 저장 메모리가 필요하다.

3.4 Block Texture Index 생성

분해된 텍스처들은 자신을 중심으로 4방향(동,서,남,북)에 대해 연결성이 가장 높은 텍스처 정보를 찾아 블록 텍스처간의 연결정보를 TAT DB에 추가한다. Index 정보는 서로간의 Linked List로 구성하면 시스템의 Loop 간소화로 속도향상을 얻을 수 있다.

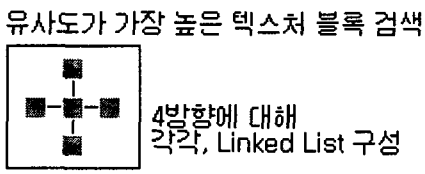

그림 5. 한 블록텍스처의 연결구성

그림 5와 같이 유사도 측정은 TAT DB에 있는 모든 블록 텍스처들에 대해서 수행되며, 각 블록 텍스처들은 모든 방향에 대해 유사도가 높은 인덱스 정보를 포함한다. 왼쪽 텍스처 매칭 누적값 T_{Dest}와 매칭이 이루어질 본 블록 화소 누적값 T_{Sour}는 다음식을 통해 유사도 T값을 얻을 수 있다.

$$T_{Sour} = \sum_{y=by*j}^{by*j+bx} P_{bx*(i-1)+bx-1,\ y} - P_{bx*i,\ y}$$

$$T_{Dest} = \sum_{y=by*j}^{by*j+bx} P_{bx*+bx,\ y} - P_{bx*i,\ y}$$

여기서 $P_{x,y}$ = *Source Texture Image*

$$T = |T_{Sour} - T_{Dest}| \qquad (2)$$

(2)식에서 계산에 사용된 블록과 매칭이 이루어질 블록은 각각의 화소에 대해 차분을 통해 유사도를 측정할 수 있다. 유사도를 측정하는 과정에서 한개의 블록 텍스처가 모든 블록 텍스처에 대해 4방향으로 유사도 측정이 일어나므로 많은 양의 계산시간이 소요된다. 하지만 이렇게 해서 TAT 테이블이 만들어지면 이를 참조하여 같은 지형 텍스처의 순환 타일을 무한대로 만들어 낼 수 있다. 블록 텍스처 선정 과정에서 각 방향에 대해 블록 텍스처를 선정할 때 Random Weight를 주어 한번 선정된 블록

텍스처는 연속해서 선정 되지 않도록 한다.

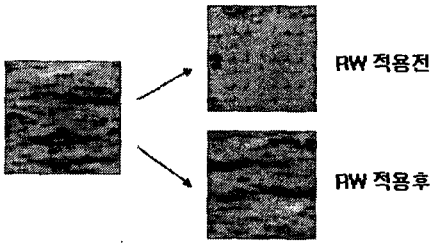

그림 6 Random Weight 적용전과 적용후

3.4 깊이값을 이용한 자동 텍스처 맵 생성

Height-Map으로부터 깊이값을 입력받으면 그림7의 함수를 통해 서로 다른 두개의 텍스처의 색상 혼합비율을 결정할 수 있다. 높이가 224인곳은 눈 텍스처가 50%, 암석 텍스처가 50%가 포함되어 밝은 회색빛이 도는 텍스처가 생성이 된다.

```
float texfactor(float h1, float h2)
{
  float percent;
  percent = (64.0f - (h1 - h2)) / 64.0f;

  if(percent < 0.0f) percent = 0.0f;
  else if(percent > 1.0f) percent = 1.0f;

  return percent;
}
```

그림 7 두개의 텍스처 색상혼합결정 함수

3.5 깊이정보에 기반한 자동 자연-객체 생성

자연-객체가 존재하는 텍스처 종류와 높이값을 선행적으로 설정해 두어야 하는 불편함이 있지만 이러한 설정을 각 자연-객체에 대해 한번만 해두면 그림8과 같이 거대한 대륙에 대해서 자동으로 객체들을 생성시킬 수 있다.

그림 8 깊이정보를 이용한 나무객체 출력

4. 결론

본 논문은 기존의 맵 제작 어려움을 개선시킬 수 있는 방법을 제안한 것으로 제안한 알고리즘의 적용으로 지형에 사용되는 텍스처를 순환 텍스처로 생성시켜줄 수 있게 되었으며, 하나의 소스 텍스처를 이용하여 세부 블록화 및 블록매칭을 위한 TAT 생성을 통해 하나의 텍스처를 여러종류의 순환텍스처로 만들어 줄 수 있게 되었다. 또한 이러한 텍스처를 다단계 링크 인덱스화로 통해 하이트맵의 3D 지형에 텍스처맵 및 이미 만들어둔 자연-객체들도 자동으로 해당 지형에 생성 될 수 있게 되었다. 이로 인해 지형 텍스처가 상당부분 자동화되면서 맵 디자인의 제작시간을 줄일 수 있었으며 기존의 방법에 비해 사실적인 지형영상을 만들어 줄 수 있게 되었다. 앞으로의 연구과제는 제안한 본 알고리즘의 최적화 및 적용된 알고리즘의 게임을 개발하는 것이다.

[참고문헌]

[1] 2002 게임백서, 한국게임산업개발원, 2002.

[2] Kokaram, A., "Parametric texture synthesis for filling holes in pictures", Image Processing. 2002. Proceedings. 2002 International Conference on, Vol. 1, pp. 325 -328, 2002

[3] Tonietto, L., Walter, M., "Towards local control for image-based texture synthesis", Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on, pp. 252 -258, 2002

[4] L.Y.Wei and M.Levoy, "Fast texture synthesis using tree-structured vector quantization", ACM Proceedings of SIGGRAPH, 2000

[5] L. Liang, C. Liu, Y. Xu, B. Guo and H. Y. Shum, "Real-time texture synthesis by patch-based sampling", Micro-Soft Research, Microsoft Corp., Mar 2001

[6] Andre LaMothe "Tricks of the Windows Game Programming", MIN Press

[7] http://leechen.wzsoft.com

[8] http://supercms.woweb.net/