High-Resolution Image viewing On Projection-based Tiled Display Wall

Abstract

본 논문에서는 프로젝션 기반의 벽면 타일 디스플레이에서 고해상의 위성, 천문학 사진을 확대, 축소, 이동 등의 조작에 할 수 있는 Viewer 구성을 위하여 중요한 이슈에 대하여 언급하였다.

1. Segmentation and Combination
2. Limit of the Maximum Texture Size
3. Distributed Data Share
4. Synchronization

1. Segmentation and Combination(계속)

기본적으로 전체 벽면디스플레이는 Monitor-PC의 Monitor와 맵핑된다.

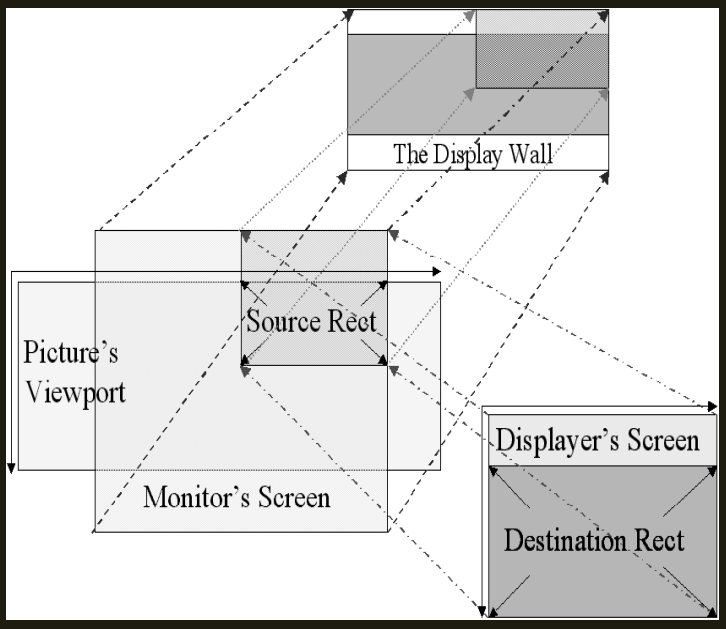
1. 기하학적 좌표와 관계 없이 각각의 Displayer Screen의 영역이 곧 Display-Wall와 Monitor Screen의 일부가 되기 때문에 각각의 Displyer는 영역의 교차를 사용하여 맵핑된다.

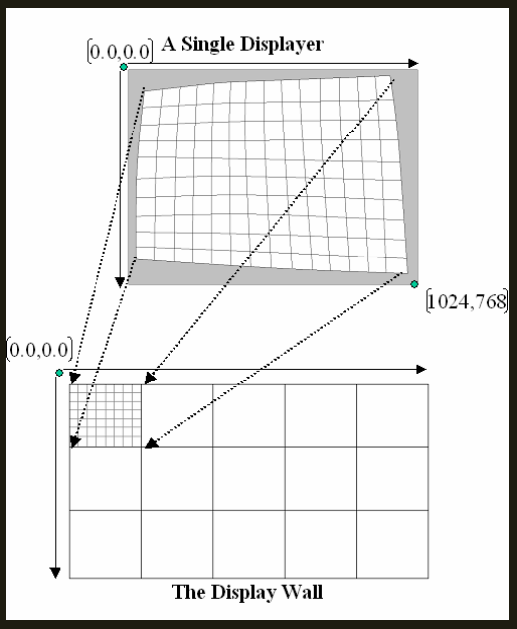
1. Segmentation and Combination(계속)

2. 1에서의 결과로 만들어진 각각의 Displayer영역에 기하학적 좌표에 의한 격자와 해당 Display-Wall의 일부 영역의 격자와 서로 맵핑된다.

이 두 가지의 과정을 거쳐서 image data가 각각의 Displayer에 보여질 수 있다.

1. Segmentation and Combination





First pass

Second pass

2. Limit of the Maximum Texture Size(계속)

Monitor Screen에 표현되는 고해상도의 이미지의 일부 또는 각각의 Displayer에 표현 되어지는 이미지의 일부가 GPU가 처리 할 수 있는 한계를 초과하는 경우가 생기게 되므로 Maximum Texture Size를 극복 할 수 있는 다양한 해상도의 텍스쳐 계층을 동적으로 선택 할 수 있는 방법에 대해 설명하고자 한다.

1. 원본이미지의 ¼크기만큼씩 줄여가면서 최대 텍스쳐 크키보다 작은지 검사한다.

2. Limit of the Maximum Texture Size(계속)

2. 최대 텍스쳐 사이즈보다 작은 Layer를 찾았다면 선택하게 된다.

If 0 ≤ SrcRectW < TexW1 :
Layer1 is selected,
0 ≤ SrcRectW1 < TexW1
TexW/2 ≤ SrcRectW1 < TexW
Else if TexW1 ≤ SrcRectW < TexW2:
Layer2 is selected,
TexW1/2 ≤ SrcRectW2 < TexW2/2
TexW/2 ≤ SrcRectW2 < TexW
Else if TexWN−1 < SrcRectW ≤ TexWN:
LayerN is selected,
TexWN−1/2N−1≤ SrcRectWN< TexWN/2N−1
TexW/2≤ SrcRectWN < TexW

2. Limit of the Maximum Texture Size

- 텍스쳐의 해상도 대신에 스크린의 해상도를 사용하게 된 조건은 최대 텍스쳐 크기가 스크린의 해상도의 두배보다 크거나 같아지기 때문이다.
- 고해상도의 이미지는 최대 텍스쳐의 크기도 초과하게 되고 Monitor−Screen의 해상도를 초과하게 되기때문에 최대 텍스쳐 크기에 맞도록 조정하고 해당 screen의 해상도에 맞도록 해야 하기 때문이다.

3. Distributed Data Share(계속)

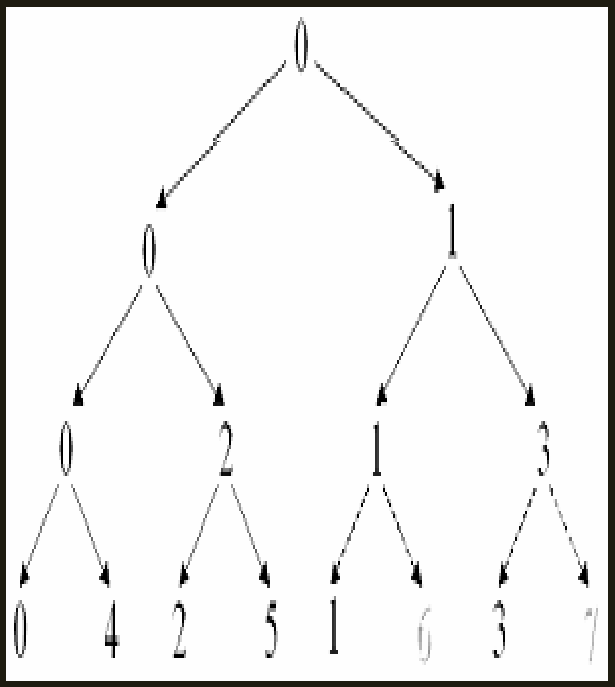
PC cluster환경에서 메모리를 관리하고 정보를 전송하기 위해서 송신측이 수신측에게 1:다의 전송을 1:1로 하게 된다면 시간복잡도는 O(N)이 되게 된다. 하지만 cluster의 크기가 매우 커진다면 기분좋지 않은 상황이 벌어지게 된다. 불필요한 데이터를 만들어내지 않고 최대한 전송시간을 단축시킬 수 있는 방법을 설명하고자 한다.

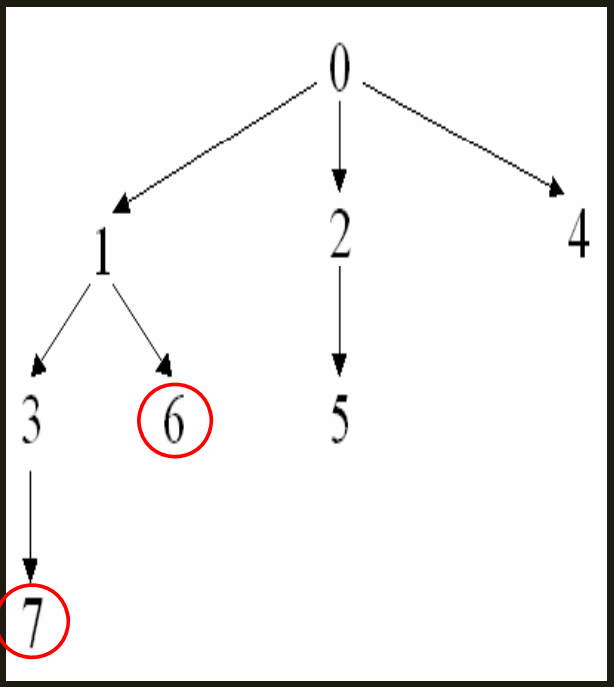
1. 전송에 대해 이진트리 구조를 사용하고 각 노드(PC)에 Monitor PC는 0번 노드로 놓는다..

3. Distributed Data Share(계속)

2. 0번 노드에서는 0번과 1번 노드에 전송하고 0번과 1번 노드는 다시 0,2,1,3번 노드에 전송하게 된다.
3. 이러한 식으로 이진트리를 만들고 전송하게 하는 방식이 있는데, 사실 실제 동작하는 PC보다 leaf 노드가 많아지게 된다.
4. 결국 동작하지 않는 노드나 존재하지 않는 노드에 대해 표현의 대책이 필요하게 된다. 존재 하지 않는 노드는 노드에는 있으나 Invalid의 mark를 해서 존재하지 않는 노드임을 명시한다.

3. Distributed Data Share





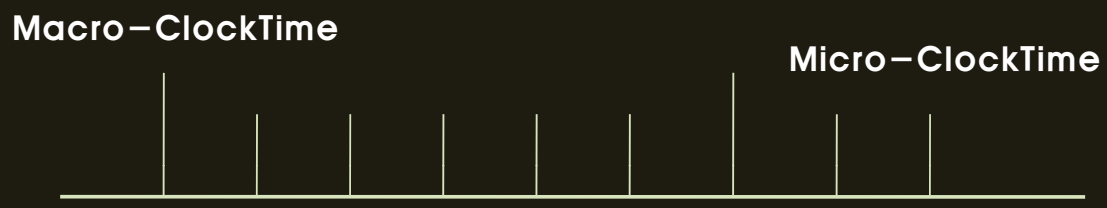
4. Synchonization(계속)

PC cluster환경에서 이미지의 이동 및 확대, 축소에서 오는 각 화면간의 동기화는 매우 중요하다. 변화에 따라 전송하고 수신하는 메시지에 따라서 다수의 Displayer가 동기화 되어야 자연스러운 이미지의 결과를 얻을 수 있기 때문이다.

단순히 Monitor-PC의 메시지 전송에 따른 동기화를 하려고 한다면 매우 느린 결과를 볼 수 있게 된다.

이러한 문제를 해결하고자 Macro Micro clock-sets을 사용하여 해결하고자 하였다.

4. Synchonization



간단히 설명하자면 Monitor PC에서 Macro clock을 각 Displayer에 전송하면 받은 Displayer는 Monitor PC에서 받은 offset값과 결합하여 새로운 offset이 만들어지게 되고, offset의 결과는 각 Displayer사이의 최대 랜더링 시간으로 생성된 Micro-Clocktime에 맞추어서 동작하게 된다.

5. Conclusion

본 논문에서 강조하고자 하는 내용은 텍스쳐 맵핑, 분산 데이터 접근, 동기화였다.

최대 텍스쳐보다 큰 고해상 이미지에 대한 처리, 이진트리를 이용한 분산 데이터의 전송, Macro-Micro clock을 사용한 동기화를 사용하여 실시간 이미지에 대한 변화에 대하여 로딩시 필요한 시간복잡도를 줄이는 효과를 볼 수 있었다.