Journal of Digital Contents Society Vol. 14 No. 3 Sep. 2013(pp. 303-311)
http://dx.doi.org/10.9728/dcs.2013.14.3.303

분할 차영상을 이용한 전시 자동 스크린세이버 구현 기법

안현수*, 김혜영**

요 약

기존의 정적인 전시의 관람객이 있든 없든 무한 반복 재생되는 전시물의 형태는 몰입감과 집중력을 떨어트린다. 따라서 본 논문에서는 기존의 일반 전시물에 분할 차영상을 이용한 전시 자동 스크린세이버를 도입함으로써, 체험형 전시물로 변환할 수 있는 기법을 새로이 제안하였다. 영상처리의 한 기술인 차영상을 개량한 분할 차영상을 이용해 움직임을 판단하여 상태를 유지 혹은 전이시키는 매커니즘을 제안하고, 이 제안기법을 적용한 전시 자동 스크린세이버를 구현하였으며, 테스트를 통하여 집중력 있는 체험형 전시물의 긍정적 측면에서의 사용 가능성을 보였다.

키워드 : 인간과 컴퓨터의 상호작용, 전시 기술, 체험형 전시 예술, 차영상, 분할 차영상, 스크린 세이버

A Study of Implementation Automatic Screen Saver using Division of Difference Image

Hyun-Soo An*, Hye-Young Kim**

Abstract

On original exhibits, which is passive and repeat itself endlessly irrelevant of the present of spectators, reduce its immersion. Therefore in this paper, we propose the scheme to recreate original exhibits be adding automatic screen saver by using division of different images to make experimental of an area of different images, to judge movement, maintain conditions or to propose the transition of mechanism. By implementing and analyzing this automatic screen saver, which was made through the scheme of proposition, we show the positive aspect of the use of centralized automatic screen saver.

Keywords : Human-Computer Interaction, Exhibition Technology, Interactive Media Art, Difference Image, Screen Saver

1. 서론

오늘날, 점점 발전하는 IT기술을 이용한 각 분야와의 접목이 활발해지고 있다. 이러한 접목은 전시관, 박물관 등 새롭고 참신한 것을 요구하는 전시분야에서도 두드러지고 있다. 예술과 첨단 테크놀로지 영역에서 이와 같은 융합현상이 연구되어 더 재미있고 대중적인 예술작품을 증가시키고 있는 것이다.[1] 과거의 전시공간이 전시물 위주의 공간이었다면, 현대의 전시공간은 관람객 지향의 공간으로 점점 변모하고 있다.[2] 기존의 단 방향의 정적인 전시에서 양 방향의 동적인(Interactive) 전시의'체험형 전시'가 각광을 받는 것이다.[3] 실제 체험형 전시가 관람객에게 긍정적인 영향을 미친다는 연구 결과도 다

※ 교신저자(Corresponding Author): Hye-Young Kim
접수일:2013년 07월 03일, 수정일:2013년 09월 05일
완료일:2013년 08월 15일
* 홍익대학교 게임학부 게임소프트웨어전공
email: ahs0311@naver.com
** 교신저자, 홍익대학교 게임학부 게임소프트웨어전공 부교수
Tel: +82-44-860-2683, Fax: +82-44-868-2710
email: hykim@hongik.ac.kr
▣ 본 연구는 정보통신산업진흥원의 IT/SW 창의 연구과정의 연구결과로 지식경제부와 (주)NHN에 의해 지원된 과제로 수행되었음"
(NIPA-2012-(H0506-12-101))
▣ 이 논문은 2013학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음

수 발표되고 있다.[4][5] 이러한 시점에서 전시관의 기존의 정적인 전시 형태인 관람객이 있든 없든 무한 반복 재생되는 전시물의 형태는 몰입감과 집중력을 떨어트려 효과적인 전시 환경을 위해 개선이 필요하다.[6]

이를 해결하기 위해서는, 평소에는 전시물이 작동하지 않다가 관람객이 도달했을 때, 전시물을 재생해줄 수 있는 환경이 필요하다. 하지만, 이러한 환경 조성을 위해 인력을 추가로 배치하거나 수동적으로 관람객이 직접 버튼을 눌러야 한다면, 장점이 반감될 것이다. 또, 체험형 전시물로 만들기 위해 기존의 작품을 수정하는 것은 큰 어려움일 수도 있을 뿐만 아니라, 맞지 않는 옷으로 작품의 전달 의도를 해할 수 도 있다.

따라서 본 논문에서는 효과적인 전시 환경을 위해, 관람객이 없을 시에는 전시물을 재생하고 있지 않다가, 관람객이 다가왔을 때 전시물이 재생하도록 반응하는 전시 자동 스크린세이버를 구현하는 기법을 제안한다. 또한 그 반대의 경우인, 관람하던 관람객이 나가면 전시물이 정지되고 전시 자동 스크린세이버가 작동하는 것도 포함한다. 구현에 이용할 기법으로는 웹캠을 사용하여 영상처리 기법의 하나인 '차영상' 기법을 통해 영역 내에 사람이 들어왔는지 판단하는 방법을 사용한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 기존의 체험형 전시의 종류와 특징, 차영상 기법에 대해 알아보며, 3장에서는 분할 차영상을 이용하여 전시 자동 스크린세이버를 구현하는 기법을 제안하고, 4장에서 실제 구현과 그 실행 결과를 보인다. 마지막으로 5장에서 결론을 기술한다.

2. 관련 연구

체험형 전시는 크게 참여형, 몰입형, 반응형, 감각형, 유도형의 유형으로 나눌 수 있다.[7] 참여형은 마우스, 키보드, 부착된 센서를 통해 작품영상에 인위적인 변화를 준다. 몰입형은 특수장비를 보조 장치로 활용하여 오감을 자극한다. 반응형은 센서 기술을 활용하여 반응하는 공간을 만들어 관람객을 모은다. 감각형은 시각 이외의 감각을 활용하여 감성을 자극하도록 구성한다. 유도형은 영상이나 전시물이 시간간격을 두고 정기적으로 나타나거나 움직이도록 한다.

이러한 기존 체험형 전시물은 제작 초기부터 체험형 전시를 고려하여 기획되어 제작하게 된다. 현재 기존의 정적인 전시물을 체험형 전시의 효과를 보이도록 하는 관련연구는 전무한 상태이며, 만약 체험형 전시물의 효과를 위해서는 기존 작품을 대폭 수정하여 체험형 전시물로 만드는 방법 밖에 없는 실정이다.

하지만 본 논문에서 제안하는 기법은 체험형 전시물이 아닌 일반 전시물에 전시 자동 스크린세이버를 도입하여 체험형 전시물의 효과를 보이도록 한다. 관람객이 없을 시에는 스크린세이버를 띄우고 있다가 관람객이 들어오면 이를 감지, 반응해서 전시물을 재생해주는 방식으로, 일반 전시물의 전달의도를 해하지 않는 체험형 전시의 반응형 효과를 기대할 수 있다.

이를 구현하기 위해서는 사람이 접근했는지, 자리를 떠났는지의 판단을 해야 한다. 이러한 사람의 움직임은 변화가 있는 동적인 요소로서, 해외에서 변화 탐지에 대한 연구로 활발한 영상처리의 '차영상' 기법을 사용해 판단을 할 수 있다.[8] 이는 별다른 장치 없이 웹캠 한 대만 있어도 구현이 가능하다는 큰 장점을 가지고 있다.

차영상은 약간의 시차를 두고 연속적으로 입력되는 동영상을 분석하여 물체의 이동 정보나 물체의 형태를 해석하는 정보를 추론하는 작업이다. 이동 물체를 탐지하고 탐지된 이동 물체의 동작 정보를 추출하는 작업을 위해 사용하기 때문에 군사 및 산업 분야 등 다양한 분야에 많이 적용되고 있는 기법이다.[9] 차영상은 두 영상 중 하나의 영상을 배경으로 삼아 또 다른 하나의 영상에서 그 값을 빼 계산을 하는 방식으로 구한다.

위의 방법은 일반적인 차영상의 구현 방법이다. 하지만 이 방법으로는 움직임을 판단하기 위해서는 화면 전체를 기준으로 변화된 픽셀의 수를 세는 방법밖에 없다. 따라서 해외 연구에서도 차영상 기법의 연구 방향은 화면 전체를 기준으로 변화된 픽셀의 수가 어느 정도의 Threshold를 만족할 지에 대한 연구의 방향으로 흐르고 있다.[8] 하지만, 기존의 화면 전체를 대상으로 변화한 픽셀의 수와 이를 체크할 Threshold 값에 의존하는 방법으로, 화면에 없던 사람이 나타

난 것을 체크하는 것과 화면에 있던 사람이 장소를 떠나는 것을 체크하는 것에 있어 큰 제약이 따른다. 영상처리에서는 노이즈가 고질적인 문제로서, 빛이나 기타 환경에 의한 노이즈가 발생 하는 경우가 많다. 따라서 노이즈 때문에 실제 사람이 접근하지도 않았지만 접근한 것으로 오판할 가능성이 크다. 혹은 그 반대의 경우도 농후하다.

본 논문에서는 이러한 노이즈의 영향을 최소화하기 위해 화면을 여러 노드로 분할해 각 노드의 차영상을 구함으로써, 분할한 각 노드의 Threshold를 각각 구하는 기법은 제안한다. 이는 노이즈에 의한 영향을 최소화하고 접근하는 사람의 위치도 알 수 있다는 장점으로, 사람의 등장과 퇴장을 파악하는데 있어 매우 적절한 방법이 된다.

본 논문은 2장에서 제시한 기법들로 기존의 일반 전시물을 체험형 전시물로 변환하기 위해 '분할 차영상을 이용한 전시 자동 스크린세이버 구현 기법'을 제안하여 효율적인 전시 환경에 대한 새로운 솔루션으로 기여하고자 한다.

3. 분할 차영상을 이용한 전시 자동 스크린세이버 구현 기법

본 논문에서는 사람을 감지하기 위한 방법으로 분할 차영상 기법을 제안한다. 보통 영상처리에서는 노이즈 발생 시, 넓은 영역에 산발적으로 분포가 되어 있는 경우가 많다. 일반적인 차영상 기법을 사용하여 사람을 감지하려면, 화면의 픽셀 수 대비 변화한 픽셀 수로 계산하기 때문에 사람이 아닌 노이즈인 경우에도 사람이 들어왔다고 판단될 수 있다. 따라서 노이즈의 영향을 덜 받을 수 있는 기법이 필요하다. 이러한 해결법으로 분할 차영상을 제안하고, 이를 이용한 전시 자동 스크린세이버를 구현하는 기법에 대해 제안한다.

3.1. 분할 차영상 기법

차영상은 A 영상과 바로 전 B 영상의 차이를 계산한 결과 영상이다. 바로 전 B 영상을 빈 도화지 삼아, B 영상과 달라진 A 영상(보통은 어떠한 움직임)의 픽셀을 표시해, 물체를 추출하거나 어떠한 움직임을 추적하기 위해 사용된다. 본 논문에서는 '움직임'을 감지하여 전시 자동 스크린세이버를 작동시키는 기법으로 차영상을 사용한다.

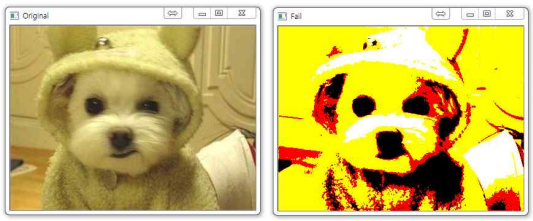
본 논문에서 제안하는 분할 차영상 기법은 기존 차영상 기법과 다르게 화면을 n(가로)*m(세로) 개로 분할한다. 이렇게 함으로써, 각각 분할된 노드 안의 노이즈 개수만을 판단하게 되어, 기존 단일 차영상에서 노이즈 개수에 의한 변화 판단으로 사람이라고 인식할 문제가 적어진다. 만약 사람이라면 분할된 노드의 픽셀을 대부분을 변화시킬 것이고, 노이즈라면 분할된 노드의 픽셀 일부분만을 변화시킬 것이다. 따라서 분할된 각 노드에서의 차영상을 구해 변화가 있는 노드의 수가 임의로 정한 k개 이상이 되면, 추후 전시 자동 스크린세이버 기법에서 이는 사람이 들어왔다고 판단하도록 한다.

분할 차영상의 과정은 다음과 같다.

1. 웹캠을 통해 영상을 받아온다.
2. RGB의 칼라로 표현되어 있는 영상을 Gray로 변환한다.
3. 화면을 임의의 개수 n(가로)*m(세로) 로 분할한다.
4. 각 분할한 노드에서의 현 A 영상과 전 B 영상의 차를 구한다.
5. 이진화하여 변화가 없는 부분은 검은색, 변화가 있는 부분은 흰색으로 표현한다.

2번 과정이 필요한 이유는 다음의 (그림 1)와 같은 경우를 막기 위해서다.

(그림 1) RGB 영상을 이진화한 영상

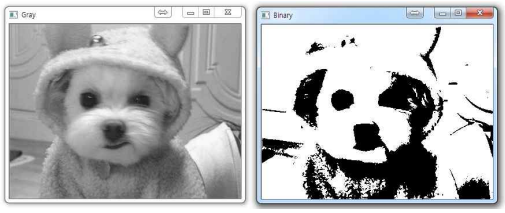


(Figure 1) Binary Image from RGB Image

컬러 영상은 RGB의 3가지 채널로 표현된다. RGB 영상을 바로 이진화 하게 되면, R과 G 그

리고 B의 채널 각각을 이진화하기 때문에, 의도치 않은 영상을 만들게 되어 차영상에 이용하기 부적절한 결과물이 된다. 따라서 2번 과정처럼 3개의 채널을 1개의 채널로 변환하는, RGB 컬러 영상을 Gray 영상으로 (그림 2)의 흑백 변환 과정이 필요하다.

(그림 2) Gray 영상을 이진화한 영상



(Figure 2) Binary Image from Gray Image

RGB 컬러 영상을 Gray 영상으로 변환하기 위해서 사용되는 행렬식은 (그림 3)과 같다.

(그림 3) YIQ와 RGB 간의 행렬식

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

(Figure 3) Matrix Equation between YIQ and RGB

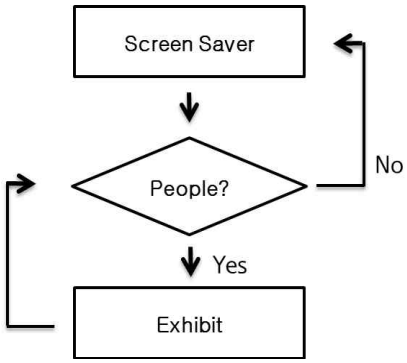
(그림 3)은 또 다른 색 표현 방식의 YIQ와 RGB 간의 행렬식을 나타내고 있다. YIQ의 Y는 밝기 정도를 나타낸다. 흑백 영상은 채도가 아닌 명도를 기준으로 만들어 지므로, Gray로 변환할 때 사용할 것도 바로 이 밝기 값인 Y이다. 따라서 Gray의 단일 채널은 RGB의 각 채널 픽셀 값을 더하여 Y = 0.299 * R + 0.587 * G + 0.114 * B 로 구한다.

5번 과정의 이진화는 값을 좀 더 명확하게 얻기 위해 필요하다. 임계치를 기준으로 값이 높은 픽셀은 흰색으로, 값이 낮은 픽셀은 검은색으로 대비되도록 표현한다. 표현이 단순하기 때문에, 영상 정보를 파악하기에 좋다. 이 때, 임계치는 각 환경에 따라, 원하는 추출 정도에 따라 상이하다. 따라서 경험 값으로 여러 번의 테스트를 거쳐 적절한 수(0 ~ 255 사이 값)를 사용한다.

3.2 전시 자동 스크린세이버 구현 기법

본 논문에서 제안하는 전시 자동 스크린세이버의 시나리오는 다음과 같다.

(그림 4) 전시 자동 스크린세이버 시나리오



(Figure 4) Scenario of Automatic Screen Saver

1. 스크린세이버 상태
   설치된 웹캠 내에 움직이는 대상이 없는 상태이다. 따라서 사람이 없다고 판단을 하여, 스크린세이버가 작동중인 상태이다.
2. 사람 판단 상태
   웹캠 내에 움직이는 대상이 있다고 판단하는 상태이다. 만약 사람이 있다고 판단되면, 스크린세이버 작동을 중지시키고, 3번 상태로 전이한다.
3. 전시물 재생 상태
   전시물을 재생하는 상태이다. 한편으로는 백그라운드에서 2번 상태를 수행하며, 사람이 있는지 판단을 하여 만약 없다면 4번 상태로 전이한다.
4. 전시물 정지 상태
   웹캠 내에 움직이는 대상이 없어졌다고 판단된 상태이다. 따라서 관람하던 사람이 나갔다는 판단을 하여, 전시물을 중지시키고, 스크린세이버를 재생시키는 1번 상태로 전이 한다.

위 시나리오의 상태 유지와 전이를 위해서는

사람이 들어왔는지, 나갔는지를 판단할 수 있어야 한다. 3.1장에서 구한 n*m개의 분할한 이진화된 차영상은 웹캠 영역 내에 사람이 들어왔는가를 판단해주는 역할을 한다. 분할한 각 노드의 이진화된 차영상 내 흰색 픽셀의 수를 세어 해당 비율이 지정한 백분율 C%(이하 C%라 한다) 이상일 경우, Count를 1 올린다. 모든 노드에 대해 판단을 하여 Count의 수 비율이 지정한 백분율 D%(이하 D%라 한다) 이상일 경우, 사람이 들어왔다고 판단할 수 있다. 단, C%와 D%는 환경에 따라 달라져야 한다. 환경에 따라 영상 내 사람의 크기가 다를 것이기 때문이다. 반대로 사람이 들어왔다가 나갔는지를 판단할 때에는 픽셀의 수를 세어 해당 비율이 이하일 경우, Count를 1 올린다. 마찬가지로 모든 노드에 대해 판단을 하여 Count 수의 비율이 지정한 백분율 E%(이하 E%라 한다) 이상일 경우, 사람이 나갔다고 판단할 수 있다. E% 또한 환경에 따라 달라져야 한다. 단, 2번 상태의 D%와 4번 상태의 E%의 값은 달라야 한다. 사람이 들어올 때는 차영상의 변화가 크지만, 머무를 때에는 변화가 작기 때문이다. 따라서 D%보다 E%는 값이 작아야 한다.

4. 구현 및 실행

본 논문에서는 화면을 분할하여 차영상을 구현하여 사람을 감지하는 분할 차영상 기법과 이를 이용하여 전시 자동 스크린세이버를 구현하는 기법을 위한 프레임워크를 설계하였다. 더불어 3장에서 제안한 구현 기법뿐만 아니라, 실제 구현에 있어 겪을 문제에 대한 해결 방법도 추가하여 작성하였다. 본 구현 및 실행 장에서는 먼저 분할 차영상 구현에 대해 보이고, 전시 자동 스크린세이버 구현, 실행의 순으로 보인다.

4.1 분할 차영상 구현

본 논문에서는 분할 차영상 구현을 위해 인텔에서 제작한 오픈 소스 컴퓨터 비전 C 라이브러리인 OpenCV를 사용하였다. 우선, 화면을 분할하는 분할 차영상을 구현하기에 앞서 차영상을 구현하였다. 다음 (그림 5~8)는 차영상을 구현한 과정 화면이다.

(그림 5) 원본영상

(Figure 5) Original Image

(그림 6) 현 흑백영상

(Figure 6) Current Gray Scale

(그림 7) 전 흑백영상

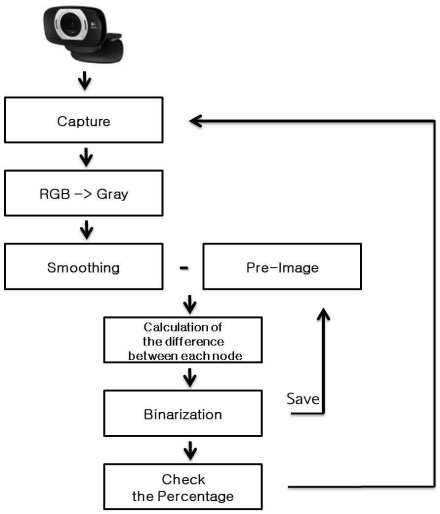
(Figure 7) Pre-Gray Scale

(그림 8) 차영상

(Figure 8) Difference Image

사람이 들어오게 되면 차영상에 의해, (그림 8)과 같이 변화에 대한 부분인 흰색 픽셀이 생기게 된다. 이 픽셀의 개수를 세어 기준 이상을 만족할 경우 사람이 들어왔다고 인식을 하게 된다. 이 차영상의 화면을 분할하여 각각 차영상을 구하는 분할 차영상 구현기법은 다음과 같다.

(그림 9) 분할 차영상 구현 기법



(Figure 9) The Method for Division of a Difference Image

1. 웹캠을 통해 영상을 받아온다.
```
m_lpCapture = QueryFrame(m_cvCapture);
```

2. RGB의 칼라로 표현되어 있는 영상을 Gray로 변환한다.
```
cvCvtColor(Image,
        NowImage, CV_BGR2GRAY);
```

3. Gaussian 필터를 적용하여 Smoothing으로 영상을 전체적으로 부드럽게 한다.
```
cvSmooth(AbsImage, AbsImage,
                CV_GAUSSIAN, 3, 3);
```

4. Count 변수를 0으로 초기화하고, 다음의 5~9과정을 n개의 가로 개수만큼 m개의 세로 개수만큼 반복한다.
```
m_iChangeCount = 0;
for(int i=0; i<n; ++i)
        for(int j=0; j<m; ++j)
```

5. 관심영역으로 설정하여 화면을 분할하여 해당 영역만 처리되도록 한다.
```
cvSetImageROI( NowImage,cvRect(width*i,
                height*j, width, height));
cvSetImageROI( PreImage,cvRect(width*i,
                height*j, width, height));
cvSetImageROI( AbsImage,cvRect(width*i,
                height*j, width, height));
cvSetImageROI( DstImage,cvRect(width*i,
                height*j, width, height));
```

6. 현 A 영상과 전 B 영상의 차를 구한다.
```
cvAbsDiff(NowImage, PreImage, AbsImage);
```

7. 이진화하여 변화가 없는 부분은 검은색, 변화가 있는 부분은 흰색으로 표현한다.
```
cvThreshold(AbsImage,AbsImage,
AbsThreshold, 255, CV_THRESH_BINARY);
```

8. 흰색으로 표현된 픽셀의 수를 세어 해당 비율이 지정한 백분율을 넘는지 판단한다.
```
int absNum = cvCountNonZero(AbsImage);
absPer = (double)absNum /
          (AbsImage->roi->width *
           AbsImage->roi->height) * 100;
if(absPer >= AbsPercent)
        ++m_iChangeCount;
```

9. 관심영역을 해제한다.
```
cvResetImageROI(NowImage);
cvResetImageROI(PreImage);
cvResetImageROI(AbsImage);
cvResetImageROI(DstImage);
```

영상처리는 빛과 그림자에 매우 민감하여, 노이즈가 발생하기 쉽다. 차영상에 있어 노이즈는 치명적인 존재이다. 구현에는 이러한 점을 고려하여 Smoothing을 통해 노이즈를 없애고 있다. Smoothing은 영상 내의 고주파 성분을 제거 해서 저주파 성분을 남기는 것이다. 고주파 성분이 영상에서 의미하는 것은 갑자기 변화하는 부분이다. 즉, Edge영역을 의미하며 이를 순화시키면 시킬수록 노이즈를 뭉개어 제거 할 수 있게 된다. Smoothing을 위한 필터로는 여러 필터가 있지만, 일반적으로 Gaussian 필터를 많이 사용한다. 이진화를 위한 임계치와 사람이 들어왔다고 판단할 흰색 픽셀의 백분율, 사람이 있다고 판단할 흰색 픽셀의 백분율은 철저히 환경에 맞추어야 한다. 때문의 여러 번의 테스트를 필요로 하며, 적절한 값을 찾기 위해 파일 입출력으로 해당 변수를 외부에서 수정 시 바로 적용할 수 있게끔 구현하였다.

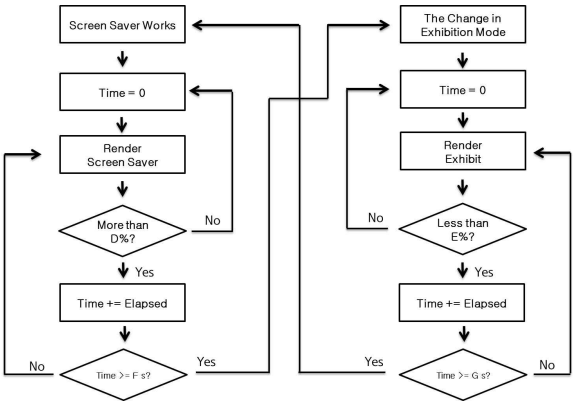
5번과 9번 과정의 관심영역은 어떠한 영역을 관심영역으로 설정하게 되면 이미지를 변경하였을 때나 값을 추출할 때 관심영역으로 설정한 영역에 대해서만 적용이 된다. 따라서 이를 이용하여 각 영역을 하나의 노드로 하여 차영상을 구현하였다.

4.2 전시 자동 스크린세이버 구현

본 논문에서는 전시 자동 스크린세이버 구현을 위해 Microsoft에서 윈도우용으로 개발한 멀티미디어 응용 프로그램 인터페이스(API)인 DirectX 9.0을 사용하여 렌더링을 하였다.

전시 자동 스크린세이버 구현기법은 다음과 같다.

(그림 10) 전시 자동 스크린세이버 구현 매커니즘



(Figure 10) Mechanism of Automatic Screen Saver

1. 스크린세이버를 작동한다.
```
m_pRender =
        new CRenderScreenSaver(d3ddev);
```

2. 변화된 시간을 0으로 초기화한다.
```
Time = 0;
```

3. 스크린세이버를 렌더 한다.
```
d3dspt->Draw(m_CamTexture, &part1,
&center1, &position1, D3DCOLOR_ARGB(
                        255, 255, 255, 255) );
d3dspt->Draw(m_texture, &part1, &center1,
          &position1, D3DCOLOR_ARGB(
                        255, 255, 255, 255) );
```
4. Count의 수 비율이 D% 이상인지 판단하여 이상이라면 5번 과정, 미만이라면 2번 과정을 수행한다.
```
int iMinCount = ( (m_iNumOfHeight *
              m_iNumOfWidth) * D ) / 100;
return m_iChangeCount >= iMinCount ? 1:0;
```

5. 변화된 시간에 한 프레임 실행 시간인 Elapsed Time을 더해준다.
```
Time += ElapsedTime;
```

6. 변화된 시간이 F초 이상이면 7번과정, 미만이면 3번 과정을 수행한다.
```
Return m_fTime >= m_fMinTime ? 1 : 0;
```

7. 전시물 모드로 변경한다.
```
m_pRender = new CRenderPiece(d3ddev);
```

8. 변화된 시간을 0으로 초기화한다.
```
Time = 0;
```

9. 전시물을 렌더 한다.
```
d3dspt->Draw(m_pieceTexture, &part1,
&center1, &position1, D3DCOLOR_ARGB(
                        255, 255, 255, 255) );
```

10. Count의 수 비율이 E% 이하인지 판단하여 이하라면 11번 과정, 초과라면 8번 과정을 수행한다.
```
int inCount = ( (m_iNumOfHeight *
                m_iNumOfWidth) * E) / 100;
return m_iChangeCount <= iMinCount ? 1:0;
```

11. 변화된 시간에 한 프레임 실행 시간인 Elapsed Time을 더해준다.
```
Time += ElapsedTime;
```

12. 변화된 시간이 G초 이상이면 1번과정, 미만이면 9번과정을 수행한다.
```
return m_fTime >= m_fMinTime ? 1 : 0;
```

구현에는 한 번의 값 변화로 무분별한 상태 전이가 되지 않도록, 변화된 시간을 체크하여 연속하여 판단을 만족했을 경우에 정해놓은 시간 이상일 때 상태 전이가 되도록 하였다. 처음의 상태는 1번 과정의 스크린세이버가 작동하고, 2번 과정에서 변화된 시간을 0으로 초기화한다. 3번 과정에서는 스크린세이버 렌더를 반복 한다. 4.1장의 분할 차영상의 계산 결과가 나오면, 이를 이용하여 4번 과정의 사람이 들어왔는지 판단을 한다. 사람이 들어왔다면 5번 과정으로 진행하여 변화된 시간에 한 프레임을 수행한 시간을 더하여, 6번 과정에서 변화된 시간이 F초 이상일 경우 7번 과정의 전시물 모드로 변경을 한다. 마찬가지로 9번 과정의 전시물 렌더를 반복하는 중에 분할 차영상의 계산 결과가 나오면, 이를 이용하여 10번 과정의 사람이 나갔는지 판단을 한다. 사람이 나갔다면 11번 과정으로 진행하여 변화된 시간에 한 프레임을 수행한 시간을 더하여, 12번 과정에서 변화된 시간이 G초 이상일 경우 1번 과정의 스크린세이버를 작동시킨다.

이 때, D%와 E%는 달라야 한다. 이는 매우 중요한 것으로, D%는 갑자기 화면의 변화가 많아짐에 따라 사람이 들어왔다고 판단을 내리게 되고, E%는 사람이 머무르면서 있던 약간의 화면의 변화가 미미해졌을 때 사람이 나갔다는 판단을 내리게 된다. 때문에 D%는 E%보다 상대적으로 커야 한다.

또, F초와 G초도 달라야 한다. F초는 사람이 들어올 경우이기 때문에 무분별한 상태 전이가 되지 않을 만한 시간 내에서 가장 빠른 시간이 좋다. G초는 사람이 나갈 것이기 때문에 F초보다는 상대적으로 긴 시간이 좋다. 본 논문의 구현에서는 각각 2초와 5초를 사용하였다.

4.3 실행

다음은 분할 차영상을 이용한 전시 자동 스크린세이버 프로그램의 실행 화면이다.

(그림 11) 스크린세이버 실행

(Figure 11) Screen Saver running

(그림 12) 사람 진입

(Figure 12) Person on Screen Saver

(그림 13) 전시물 재생

(Figure 13) Exhibition of running screen

(그림 11)은 처음 보여지는 Screen Saver의 실행 화면이다. 웹 캠이 찍고 있는 상태를 배경으로 보여주고, 아래에 'Screen Saver'라는 문구를 띄워줌으로써, 현재 Screen Saver 상태라는 것을 보여준다.

(그림 12)는 사람이 진입하였을 때의 실행 화면이다. 분할 차영상에 의해 지정한 백분율 이상의 변화를 감지한 노드의 색을 검게 변화시켜, 시각적으로 보였다. 전체 노드 개수와의 비율을 체크하여, 지정한 시간 이상이 흘렀을 시에 (그림 13)의 전시물 실행 화면으로 전환된다.

(그림 13)에서 전시물 재생을 하던 중에 사람이 나가면, 마찬가지로 지정한 시간이 흘렀을 때, (그림 11)로 전환되도록 하였다.

5. 결론

본 논문에서는 일반 전시물을 체험형 전시물로 변환하는 기법에 대한 연구가 전무한 현재, 기존의 일반 전시물에 분할 차영상을 이용한 전시 자동 스크린세이버를 도입함으로써, 체험형 전시물로 변환할 수 있는 기법을 새로이 제안하였다. 이는 일반 전시물의 전달의도를 방해하지 않는 선에서, 체험형 전시물의 긍정적인 효과를 가미한다는 점에서 기존의 전시 방법보다 전시 효과가 클 것으로 기대한다.

기존의 관람객이 있든 없든 무한 반복 재생되는 전시물의 방식이 아닌 관람객이 없을 시에는 전시물을 재생하고 있지 않다가, 관람객이 다가왔을 때 전시물이 재생하도록 하는 스크린세이버는 관람객에게 있어 나로 인해 반응하는 듯한 느낌을 주어 한층 더 몰입감 있는 전시 효과를 기대할 수 있을 것으로 보인다.

또 본 논문은, 영상처리의 한 분야인 차영상을 개량한 분할 차영상을 이용해 움직임을 판단하여 상태를 유지 혹은 전이시키는 기법도 제안하였으며, 이는 노이즈의 영향을 덜 받으면서도 영상 내 변화의 위치를 체크할 수 있게 되어 다른 분야에서도 응용, 활용 가능성이 매우 클 것으로 보인다.

본 논문에서는 기대되는 효과에 대한 검증을 수행하지 못한 상태라 이에 대한 연구가 필요할 것으로 보인다. 따라서 향후 연구로, 결과 검증을 위한 툴을 제작하고, 실제 전시 환경에서 전시 자동 스크린세이버를 적용 했을 시와 적용하지 않았을 시의 경우를 조사하여 비교 분석해 이에 따른 철저한 검증을 하고자 한다.

References

[1] K. Kim, and Y. Lim, “Production of an Interactive Artwork through Analysis on the Expression Method for Subject Area on Screen (Focused on Color, Face and Brightness),” Journal of Digital Contents Society, Vol.13, No.3, pp.439-449, Sep. 2012.

[2] S. Oh, “A Study on the Classifying the Design of Immersion Exhiition in the Children’s Museums,” Journal of Korea Design Knowledge, Vol.19, Aug. 2011.

[3] K. Doo, and S. Kim, “The Study on the Interactive Display Video Activating Plan in Experience-Type Media Space,” Journal of Korea Design Knowledge, Vol.24, Dec. 2012.

[4] Y. Song, S. Kim, and J. Lee, “Analyses on Satisfaction and Perception for Space according to Experiencing Interactive Media in Exhibition Halls,” Korean Journal of Air-Conditioning and Refrigeration Engineering, Vol.24, pp.271-280, Feb. 2012.

[5] S. Lim, S. Kwak, I. Park, J. Park, and K. Bae, “The Measurement of the Subjective Experience for Analysing the Flow Experience Degree in the Interactive Exhibit Contents,” Journal of Korean Society of Design Science, Vol.22, No.4, Jun. 2009.

[6] J. Shin, K. Wohn, and M. An, “Human - Content Interaction for Interactive Media Art Exhibition Technology,” HCI 2010, Jan. 2010.

[7] S. Kwon, “A Typological Approach of the Digital Interactive Exhibition pavilion applying Ubiquitous Concept,” Journal of Korea Design Knowledge, Vol.10, No.2, Apr. 2010.

[8] P. Rosin, and E. Ioannidis, “Evaluation of global image thresholding for change detection,” Pattern Recognition Letters 24, Jan. 2003.

[9] G. Kim, H. Choi, and E. Lee, “Extraction of Motion Information by Analyzing Difference Images,” Journal of The Korea Information Science Society, Vol.21, No.8, Aug. 1994.

안 현 수

2008년 ~ 현재: 홍익대학교 게임학부 게임소프트웨어전공 학사과정

2012년~현재: 삼성 소프트웨어 멤버십 22-1기

관심분야 : 게임 서버(Game Server), 컴퓨터 그래픽스(Computer Graphics), 프레임워크 설계(Architecture Framework), 인공지능(AI), 영상처리(Image Processing) 등

김 혜 영

2005년: 고려대학교 대학원 (이학박사-컴퓨터학)

2007년~현재: 홍익대학 게임학부 게임소프트웨어전공 부교수

관심분야 : 게임 엔진(Game Engine), 게임서버(Game Server), 모바일 게임(Mobile Game), 인터렉티브 디바이스(Interactive Device), HCI, 위치관리 기법(Location Management Technique), 부하분산처리(Load Balancing Processing) 등