한국컴퓨터정보학회 하계학술대회 논문집 제31권 제2호 (2023. 7)

운영체제 시각화를 위한 GUI 구현

김동휘, 박연택, 정해람, 양길모, 주용완, 이준동
강릉원주대학교
e-mail: awesome_devnet@outlook.com, parkyt123@naver.com, jhr83199@gmail.com,
kiyawo0210@naver.com, gmyang@gwnu.ac.kr, ywju@gwnu.ac.kr, jlee@gwnu.ac.kr

GUI Implementation for operating system visualization

DongHwi Kim, YeonTaek Park, HaeRam Jung, Gilmo Yang, YongWan Ju, JunDong Lee
GangNeungWonju National University

● 요 약 ●

운영체제(Operating System)는 사용자의 하드웨어, 시스템 자원(System Resources)을 제어하고 프로그램에 대한 일반적 서비스를 지원하는 시스템 소프트웨어(System Software)이다. 시스템 하드웨어를 관리할 뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공한다. 최근에는 가상화 기술의 발전에 힘입어 실제 하드웨어가 아닌 가상 머신(HyperVisor) 위에서 실행되기도 한다.

본 연구에서는 다중 코어 프로세서를 타겟으로 한 소규모 운영체제 개발 프로젝트의 일환으로 화면 모드를 전환해 주고, 화면을 그리는 기능을 작성해 주었다. 이를 잘 활용하면 고해상도 그래픽모드에서의 보다 심도 있는 그래픽의 구현, 나아가 임베디드 시스템, IOT 등 다양한 분야에 이용할 수 있다.

키워드: 운영체제(Operating System), 시스템 소프트웨어(System Software), GUI

I. Introduction

텍스트 모드는 이전 그래픽 컨트롤러와 호환성 유지한다. 그러나 그래픽 모드는 호환성을 유지하지 못한다.

그래픽 컨트롤러마다 지원할 수 있는 해상도와 색의 수가 다르기 때문이다.따라서 OS가 정보를 컨트롤러에서 찾아 처리를 해야한다.

VESA(video Electronics Standards Association)에서 BIOS를 통해 그래픽 모드로 전환할 수 있는 방법을 규정하여 표준화 되어있다. 따라서 BIOS 인터럽트 서비스를 통해 그래픽 모드로 전환하여 이미지를 출력할 수 있다.

운영체제를 개발함에 있어 그래픽 모드로 화면을 전환하는 기능을 작성하기 위해 고려해야 하는 문제가 있다. 바로 실제로 운영체제에 대해 사용할 수 있는 메모리 공간이 제한적이라는 점이다. 가령 1280 X 1024의 해상도에 32비트 색상을 지원하는 화면 모드의 경우, 1280 * 1024 * 32BIT(4BYTE) = 5MB의 메모리 공간이 필요하다는 것이다.

이렇듯 몇 가지 실제 기능의 구현에 앞서 사전에 유의해야 하는 사항들이 있다.

II. Preliminaries

2.1 VBE 구현 서론

VBE에서의 핵심은 바로 선형 프레임 버퍼이다. 선형 프레임 버퍼는 비디오 메모리 전체를 하나로 통합하여 연속된 메모리 영역으로 사용하는 기법이다. 운영체제에서 화면 모드는 크게 텍스트 모드와 그래픽 모드로 나누어진다.

III. The Proposed Scheme

3.1 VBE를 활용해 화면 모드를 전환

운영체제에서 화면 모드를 전환하기 위해서는 BIOS 서비스를 사용하게 되는데, 화면 모드를 전환하는 인터럽트인 0x10를 사용하며, 적절한 기능 번호와 플래그 값을 일부 레지스터에 등록해 주어야 한다.

```
MOV AX, 0x4F02   ; VBE 기능 번호
MOV BX, 0x4117   ; 1024 * 768 * 16BITS

INT 0x10
```

VBE 모드에 대한 세부적인 정보는 0x7E00 위치에 저장되어 있으므로, 해당 주솟값을 참조하여 해상도, 색상 등의 정보 확인이 가능하다.

```
MOV AX, 0x4F01    ; VBE 기능 번호
MOV CX, 0x117     ;  1024 * 768 * 16BITS
MOV BX, 0x07E0    ;

INT 0x10
...
```

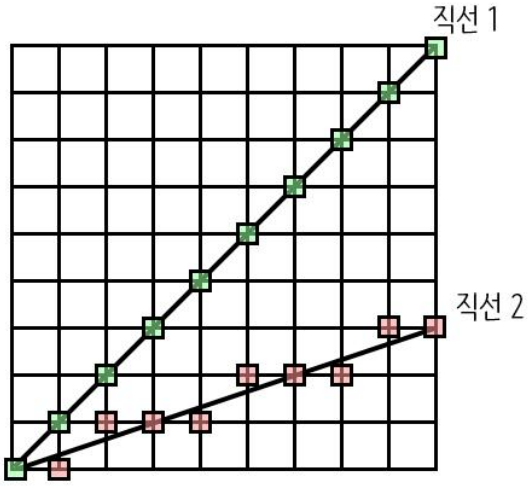
3.2 화면에 그리기

운영체제에서 화면 모드를 전환하기 위해서는 BIOS 서비스를 사용하게 되는데, 화면 모드를 전환하는 인터럽트인 0x10를 사용하며, 적절한 기능 번호와 플래그 값을 일부 레지스터에 등록해 주어야 한다.

모든 그리기 동작은 PIXEL, 즉 점을 화면에 표시하여 모양을 어떻게 조합하느냐에서 시작된다. PIXEL은 비디오 메모리로 다룰 수 있는 가장 작은 표시 단위이다.

```
PIXEL : (X축 해상도 * 색상 수) * Y축 방향의 OFFSET + X축 방향의 OFFSET * 색상 수
```

3.2.1 선 그리기



직선1과 같은 경우 픽셀로 충분히 직선을 표현이 가능하다.
직선2와 같은 경우는 직선이 픽셀 사이로 통과하고 있기 때문에 픽셀로 직선을 표현하기 어렵다.

직선이 연속적이지 않은 불연속 좌표계로 옮겼기 때문에 발생하는 문제이다.

이를 해결하기 위해서는 최대한 직선에 가깝게 픽셀을 표시하여 픽셀이 보이지 않을 정도의 높은 해상도를 사용하여야 한다. 그러나 이러한 방법도 계단 현상으로 인하여 완전한 직선을 표현하는 것은 어렵다.

최대한 직선에 가깝게 픽셀을 표시하는 방법 :
1. DDA(Digital Differential Analyzer) 알고리즘
2. 브레슨햄(Bresenham)의 직선 알고리즘
3. 샤오린(Wu Xiaolin)의 직선 알고리즘

브레슨햄 알고리즘은 곱셈 또는 나눗셈 연산을 사용하지 않고 덧셈과 뺄셈만으로 직선을 그려 다른 알고리즘에 비해 속도가 빠르고 비교적 간단하다.

3.2.2 사각형 그리기

사각형을 그릴 때는 속이 빈 사각형과 속을 채운 사각형의 두 가지 형태로 표현. 빈 사각형은 선 네개로 구성할 수 있고, 찬 사각형은 해당하는 메모리 영역에 COLOR 데이터를 채워 준다. 빈 사각형에 해당하는 왼쪽 위(x1, y1)의 좌표와 오른쪽 아래 끝 좌표(x2, y2)를 지정하여 대칭 관계를 이용해 나머지 좌표를 구한다. 내부를 채울 때에는 Y 좌표를 고정하고 X 좌표를 이동시키면서 사각형의 길이만큼만 픽셀을 출력하여 처리한다.

3.2.3 원 그리기

'중심점 원 알고리즘(MIDPOINT CIRCLE ALGORITHM)'을 활용하는데, 이 알고리즘은 현재 픽셀의 위치를 (X, Y)라 할 때, 이에 인접한 두 픽셀(X + 1, Y), (X + 1, Y − 1)의 중심점이 원의 내부에 있는지, 혹은 외부에 있는지를 확인하여 그릴 픽셀을 선택하는 알고리즘이다. 중심점이 원에 포함되는지 여부는 원의 중심에서 중심점까지 떨어진 거리와 원의 반지름을 비교하여 확인할 수 있다.

IV. Conclusions

운영체제(Operating System)는 사용자의 하드웨어, 시스템 자원(System Resources)을 제어하고 프로그램에 대한 일반적 서비스를 지원하는 시스템 소프트웨어(System Software)이다. 따라서 운영체제의 본질적인 기능에 대한 접근성을 보장하기 위해서는 한 방법으로 GUI 환경을 제공하는 것이다.

본 연구에서는 다중 코어 프로세서를 타겟으로 한 소규모 운영체제 개발 프로젝트의 일환으로 화면 모드를 전환해 주고, 화면을 그리는 기능을 작성해 주었다.

이는 가장 기본적인 시각화의 표현이며, 이를 이용하면 다양한 데이터의 시각화 표현도 가능하다. 또한, 고해상도 그래픽모드에서의 보다 심도 있는 그래픽의 구현, 나아가 임베디드 시스템, IOT 등 다양한 분야에 이용할 수 있다.

ACKNOWLEDGEMENT

이 논문은 2023년도 정부(개인정보보호위원회)의 재원으로 한국인터넷진흥원의 지원을 받아 수행된 연구임 (1781000005, 반정형 트랜잭션 및 실시간 수집 정형 데이터에서의 개인정보 가명·익명 처리 기술 개발)

REFERENCES

[1] SeungHoon Han, “IT EXPERT, 64Bit MultiCore OS Principal and Structure”, HanbitMedia, 2014.

[2] Kawai Hidemi, “OS structure and principle”, HanbitMedia, 2007

[3] Harvey M.Deitel, Paul J. Deitel, David R.Choffnes, “Operating Systems“, HanbitMedia, 2009

[4] Hyun-Hwoi Ku, “Operating System”, HanbitMedia, 2016

[5] https://vesa.org/