그래픽 기술 기초 / 색상 모델

Graphic technology basic

Color model

그래픽 기술 기초

그래픽스 기본 개념과 용어

- 3차원 그래픽 객체 생성과정

  1. 대상체 또는 객체의 기하학적인 현상을 모델링
     > 객체 정의
     > 수치 값
     > 다각형 모델

  2. 3차원 객체를 2 차원 평면에 투영

  3. 3차원 / 2차원 그래픽 객체의 색상과 명암을 추가 / 화면 출력
     > 와이어프레임 렌더링, 솔리드 렌더링

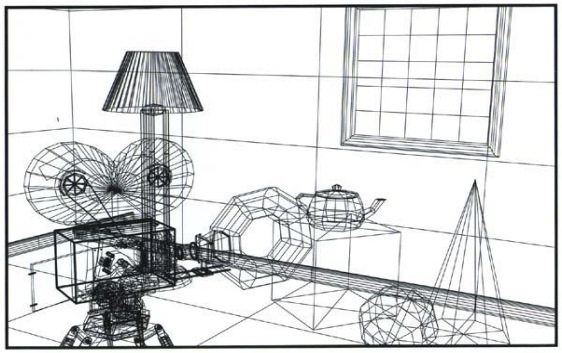
그래픽 기술 기초

Q) 3D Cube를 모델링 하시오

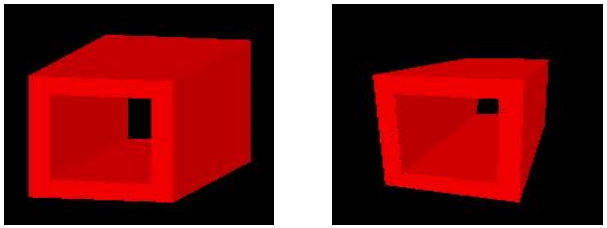
그래픽 기술 기초

그래픽 처리 과정과 용어

3차원 물체의 모델링

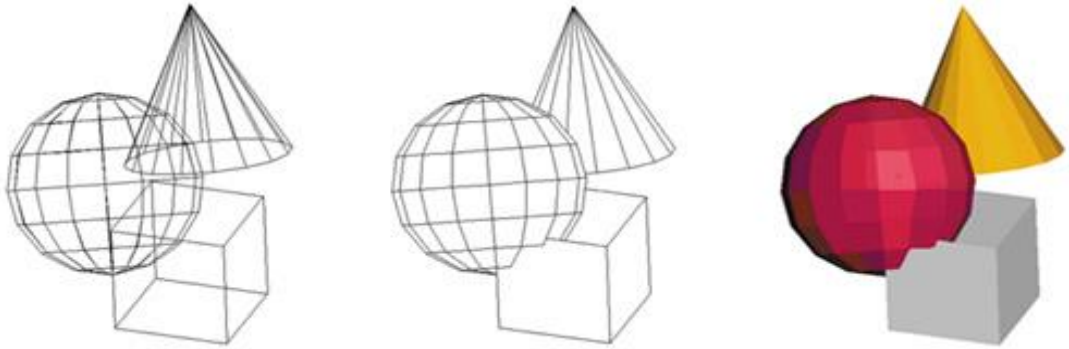


3차원 물체를 2차원
평면에 투영(Projection)



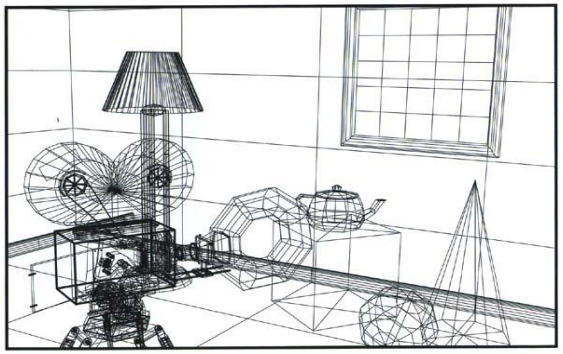
생성된 3차원 물체색상과
명암을 추가(Rendering)

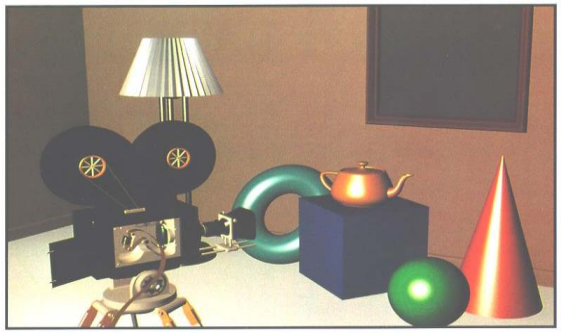
은면 (Hidden Surface)와 음영 처리 (Shading)



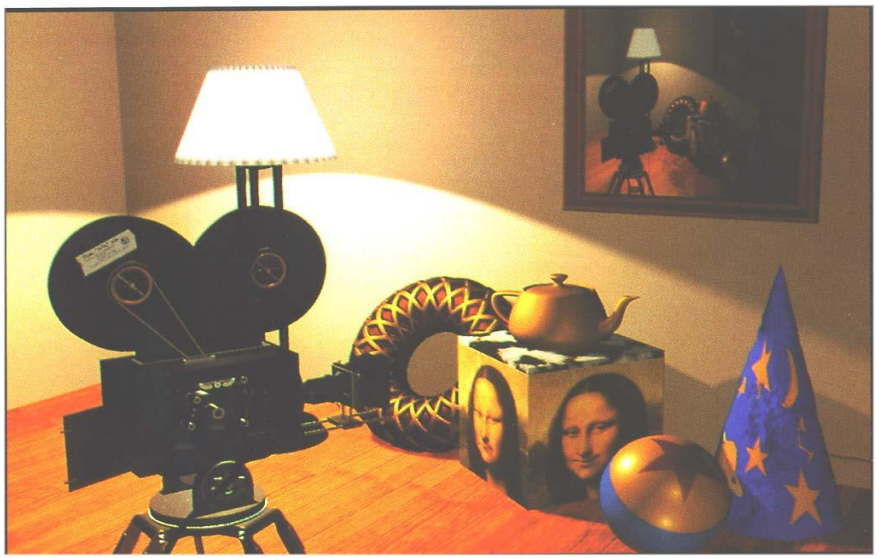
그래픽 기술 기초

그래픽 처리 과정과 용어





텍스처 매핑



그래픽 기술 기초

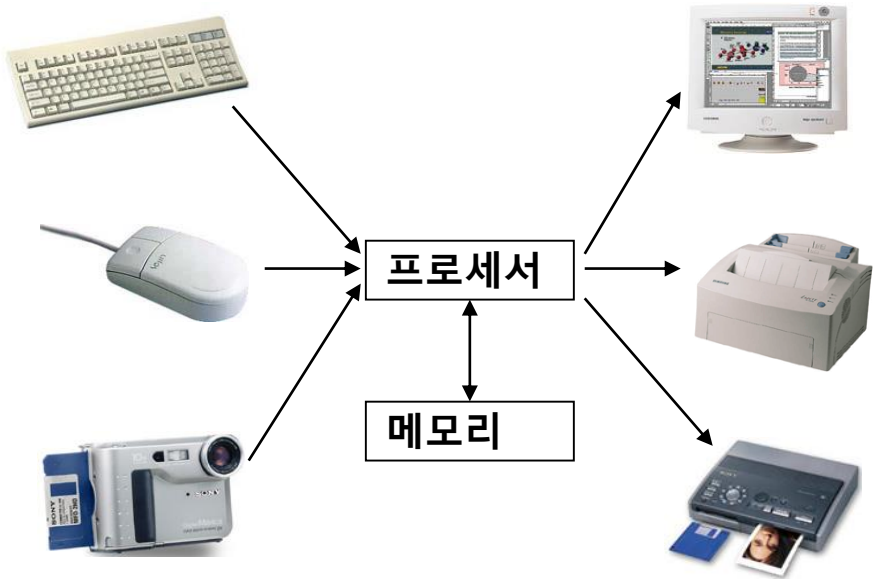
그래픽 시스템 구성요소

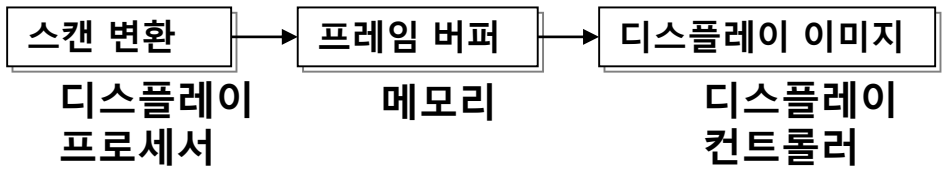
- 프로세서 (processor)
  > 그림 형성 (picture formation)
  > 그림 표시 (display of the picture)

- 메모리 (memory)
  > 디스플레이 메모리 (display memory)
  > 프레임 버퍼 (frame buffer)

- 출력 장치 (output device)
  > 래스터 스캔
  > 플로터 / 프린터 등

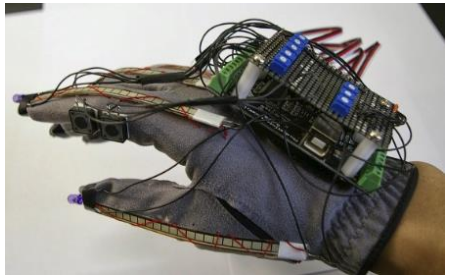
- 입력 장치 (input device)
  >마우스, 키보드 등





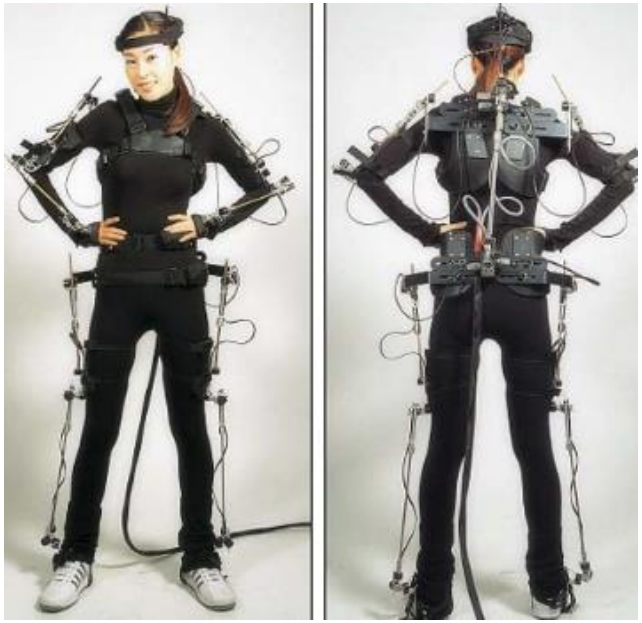
그래픽 기술 기초

그래픽 입력 장치와 출력 장치

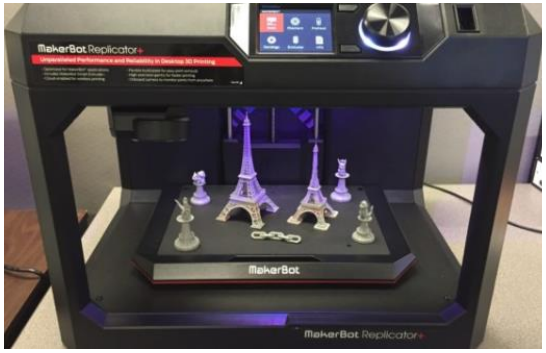


Digital Glove

HMD (Head Mounted Device)



Motion Capture



3D Printer

Hologram

그래픽 처리 장치

그래픽처리장치

[graphics processing unit]

요약 컴퓨터의 영상정보를 처리하거나 화면 출력을 담당하는 연산처리장치. 중앙처리장치의 그래픽 처리 작업을 돕기 위해 만들었으며 그래픽카드 또는 마더보드에 들어있다. 그래픽 프로세서 또는 간단히 GPU(graphics processing unit)이라고도 한다.

그래픽카드 또는 마더보드에서 그래픽에 관련된 연산을 처리하는 반도체 칩을 말한다. 중앙처리장치(CPU)의 그래픽 작업으로 인해 생기는 병목 현상을 해결하기 위해 만들었으며 2D 또는 3D 그래픽을 CPU가 처리하는 것보다 빠르게 처리할 수 있다. 이러한 그래픽 가속기능때문에 그래픽카드를 그래픽가속기(graphics accelerator)라고도 하며, 그래픽카드의 성능은 그래픽처리장치(GPU)와 비디오 램(RAM)에 따라 달라진다.

GPU(Graphics Processing Unit)라는 용어는 1999년 8월 엔비디아(NVIDIA)사에서 새로운 그래픽카드를 출시하면서 처음 사용했다. CPU의 부담을 줄이고 다른 작업에 자유롭게 사용하기 위해, CPU가 처리하던 트랜스폼(Transform)과 라이트닝(Lighting) 등의 작업을 그래픽처리장치가 대신 처리하게 하였다. CPU의 처리량이 줄어드는 대신 GPU의 처리량은 계속 증가함에 따라 많은 열이 발생하여 CPU처럼 방열판이나 냉각팬이 설치되는 경우가 많다.

컴퓨터가 사양이 다양해지고 고속, 고성능화되면서 그래픽처리장치의 중요성은 더 높아졌다. 특수효과를 내는 필터를 사용하는 프로그램과 점차 DSP적인 프로세서로 변해감에 따라 GPU의 처리량도 늘고 있다.

그래픽 기술 기초

그래픽 처리 장치

프레임 버퍼

[frame buffer]

래스터 주사 방식에서 화면에 나타날 영상 정보를 일시적으로 저장하는 기억 장치. 그래픽 프로세서가 중앙 처리 장치로부터 도형을 표현하는 디스플레이 리스트를 받아 변환하여 프레임 버퍼에 기록한다. 프레임 버퍼의 각 기억 단위는 화면의 픽셀에 하나씩 대응하며 화면에 그대로 반영된다. 즉, 화면 각 점의 온(on)/오프(off)나 색깔을 비트맵으로 기억하고 있으며 이 기억 장치에 어떤 내용을 써넣으면 그것이 화면에 표시된다. 대개 시스템의 주기억 장치와는 별도로 분리되어 있으며, 특별한 기술을 사용하여 속도를 빠르게 하기도 한다.

출처 **전자용어사전**

컴퓨터에서 화상을 기억하기 위한 기억 장치. 중앙 처리 장치에서 나온 표시 데이터의 신호는 일단 여기에 기록된 다음 디스플레이 화면에 보내져서 표시된다. 비디오 RAM의 일종이지만, 화상을 풀 컬러로 기억할 수 있는 것 등을 특별히 프레임 버퍼라고 해서 구별한다.

요약 RAM(Random Access Memory)은 기억된 정보를 읽어내기도 하고 다른 정보를 기억시킬 수도 있는 메모리로서, 컴퓨터의 주기억장치, 응용 프로그램의 일시적 로딩(loading), 데이터의 일시적 저장 등에 사용된다.

출처 **농촌진흥청**

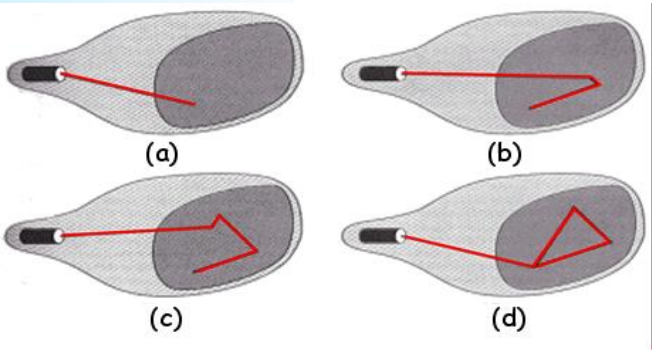
픽셀(pixel) 단위로 영상의 내용을 저장하기 위한 기억장치.

출처 **컴퓨터인터넷IT용어대사전**

픽셀 단위로 영상의 내용을 저장하는 기억 장치.

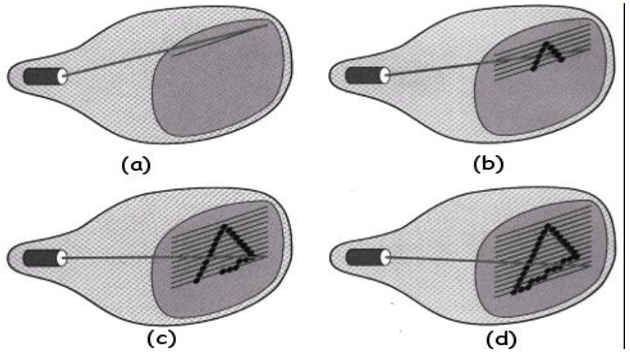
그래픽 기술 기초

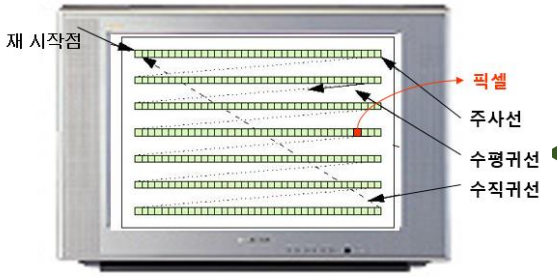
랜덤 스캔 / 래스터 스캔 모니터



- 랜덤 스캔 또는 벡터 재생 디스플레이라고도 함

- 래스터 스캔 디스플레이
  > 주사전환 : 픽셀 정보를 처리하여 프레임 버퍼에 저장
  > 주사선: 픽셀로 이루어진 수평선
  > 프레임 버퍼, 재생버퍼, 래스터, 재생기억장치
  > 프레임버퍼에서 화면 픽셀에 대한 값 저장
  > 픽셀: 프레임 버퍼내의 각 점
  > 화면 출력은 프레임 버퍼의 정보를 한번에 한 줄씩, 위에서 아래로 읽으면서 스크린상에 색칠 하는 방식과 유사
  > 래스터 픽셀 수: 디스플레이 해상도 또는 프레임버퍼의 해상도





그래픽 기술 기초

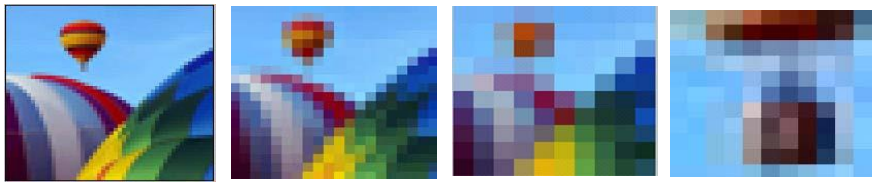
해상도

- 장치 해상도: 출력 장치 픽셀의 수
- 이미지 해상도: 장치와 무관한 이미지 자체의 해상도

- 해상도가 높으면 선명도 증가
- 해상도가 높을 수록 더 많은 정보 포함

화면의 크기를 고정한 후, 해상도를 낮춘 경우

고해상도 ← → 저해상도

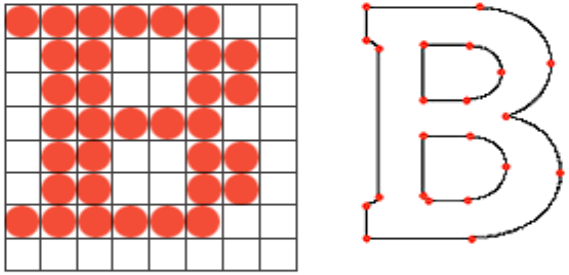


그래픽 기술 기초

래스터 / 벡터 그래픽

- 래스터(Raster) 그래픽
  > 2차원 픽셀(pixel) 집합으로 그래픽 처리 대상을 표현
  > 대용량 저장장치가 필요하지만 이미지의 복잡도에 관계없이 일정한 저장공간 적용
  > 파일 크기는 해상도에 비례, 화면을 확대 시 화질 감소

- 벡터(Vector) 그래픽
  > 점, 선, 면 같은 기하 정보와 꼭지점, 모서리, 면 같은 위상 정보의 조합으로 대상을 표현
  > 기본적으로 적은 저장용량을 차지하지만, 이미지가 복잡할수록 저장공간이 커지고 방식이 복잡
  > 벡터 그래픽은 점, 선, 곡선, 원등의 기하적 객체로 표현되므로, 화면 확대 시 화질 변화가 없음

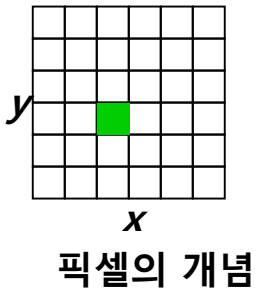


그래픽 기술 기초

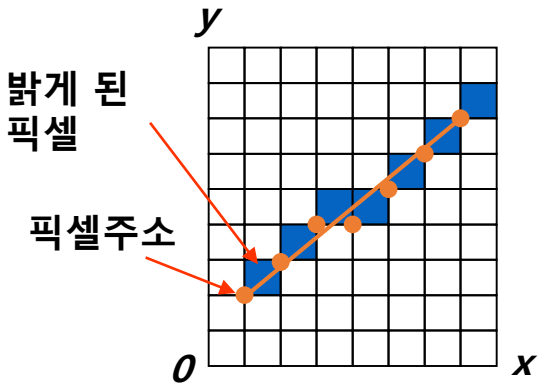
래스터 그래픽 단위 픽셀

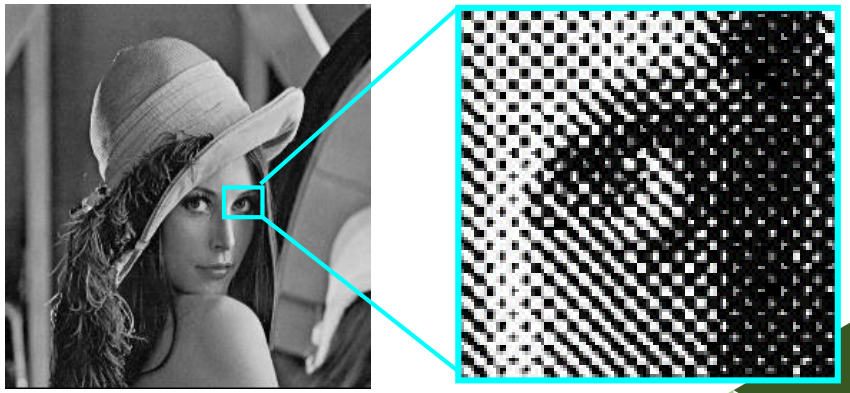
- 디지털 이미지의 기본 단위 즉, 화면상의 최소 단위
  > 그림 요소의 배열
  > 픽셀 하나 당 비트 정보는 색상 수를 결정
  > 그림을 여러 개의 점 즉, 픽셀들로 구성
  > 픽셀들은 프레임 버퍼에 저장

- 이미지는 화면 상 픽셀들에 명암과 색을 지정함으로써 구현
- 정확도와 질은 디스플레이 장치의 해상도에 따라 달라짐



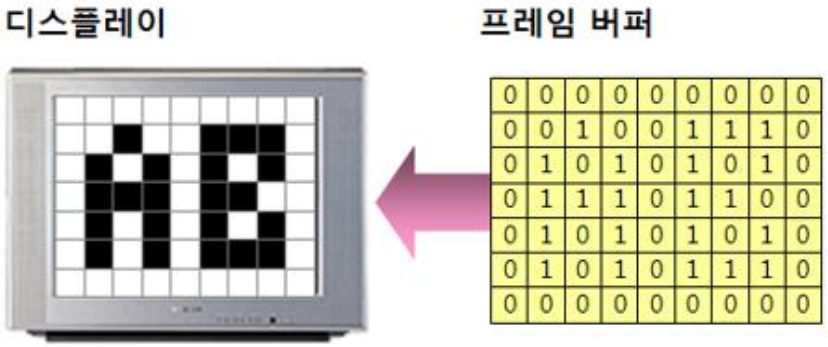
픽셀의 개념





그래픽 기술 기초

래스터 그래픽 단위 픽셀



색상수와 프레임 버퍼에서의 공간

비트	색상수	픽셀당 차지하는 공간
1	2	1 픽셀(1 비트)
2	4	1 픽셀(2 비트)
4	16	1 픽셀(4 비트)
8	256	1 픽셀(8 비트)
16	65,536	5 비트 / 6 비트 / 5 비트 — 1 픽셀(16 비트)
24	16,777,216	8 비트 / 8 비트 / 8 비트 — 1 픽셀(24 비트)

-128 ~ 127
0 ~ 255 (0xff)

이진 영상 (Binary Image)

그레이 스케일
영상 (Grey Scale Image)

컬러 영상 (Color Image)

그래픽 기술 기초

해상도

비디오 램에 따른 색상수와 해상도

1280 X 1024 최대 장치 해상도 경우

비디오램	16색	256색	하이칼라	트루칼라
1 MB	1280 ×1024	1024 × 768	800 × 600	640 × 480
2 MB	1280 × 1024	1280 × 1024	1024 × 768	800 × 600
4 MB	1280 × 1024	1280 × 1024	1280 × 1024	1024 × 768

1 byte=8 bit

화면 하나의 용량 예

화면 해상도: 640 × 480
픽셀당 비트 수: 24 비트(트루칼라)

640 × 480 × 24=7,372,800[bit]/8
=921,600[byte] ≅ 0.9 MB

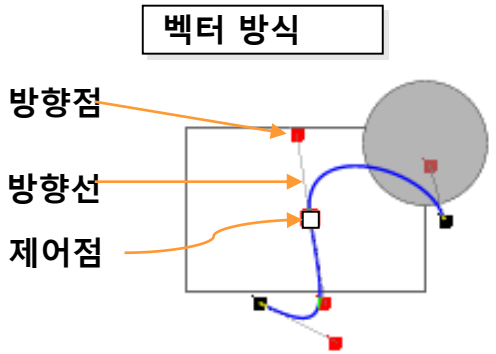
-128 ~ 127

0 ~ 255 (0xff)

그래픽 기술 기초

벡터 그래픽 단위

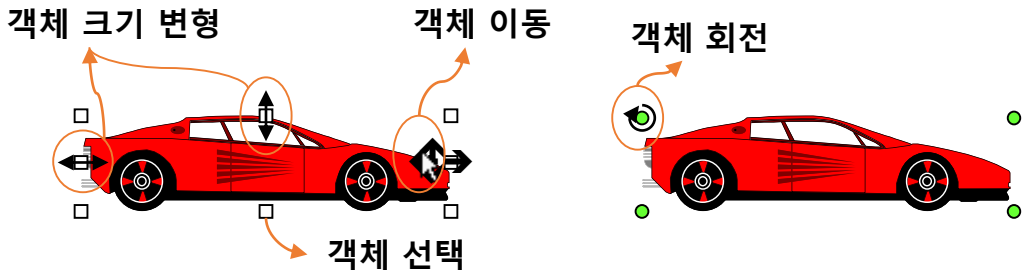
- 벡터 그래픽은 그래픽스의 기하학적 정보 및 위상 정보를 저장하고 있다가 출력시 수학적 연산에 의해 그림을 표현
  > 프리미티브: 점, 선, 면, 곡선 등



연속적이고 정확한 값을 저장

- 장점
  > 수학적 계산에 의해 디스플레이 하므로 파일 크기가 작음
  > 편집 작업이 쉬움
  > 변형에도 불구하고 화질 보존

- 단점
  > 그래픽으로 바꾸어 디스플레이 해야하므로 작업 속도가 느림
  > 색 변화가 크면 계산이 복잡해짐



그래픽 기술 기초

더블 버퍼링(Double Buffering)

- 프레임 버퍼 메모리 내에서 이미지 또는 영상정보를 구성하는 색상이나 밝기 값이 채워지고 이 메모리로부터 데이터를 가지고 와서 화면 출력

- 한 개 이상 버퍼 필요
  > 첫 번째 버퍼 이미지 ①에 대한 출력 값을 채움
  > 이미지 ① 출력 값을 채우는 동시, 이미지 ② 값을 채우기 위한 그래픽 처리 수행
  > 첫 번째 버퍼에 있는 이미지 ①가 화면에 출력됨과 동시, 두 번째 버퍼에서는 이미지 ②를 위한 출력 값을 채움
  > 이미지 ③을 화면에 출력하기 위해 첫 번째 버퍼 영역에 있는 이미 소모된 이미지 ① 값을 비우고 이미지 ③ 값을 채움

그래픽 기술 기초

래스터 영상정보 저장 방식

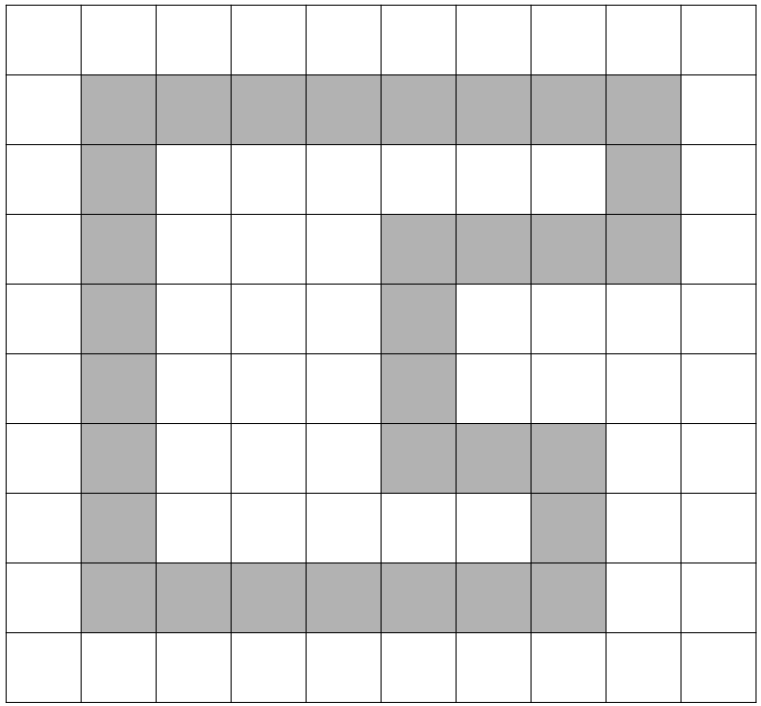
- 런렝스 엔코딩 (Run-Length Encoding)
  > 메모리 요구량을 줄이기 위한 밝기 값을 코드화
  > 각 주사선에 대하여 하나의 정수 쌍으로 저장
  > 한 숫자는 밝기 값을 나타내고 다른 정수 값은 같은 밝기를 가진 주사선상의 이웃한 픽셀 수를 나타냄

- 셀 부호화 (Cell encoding)
  > 래스터를 하나의 사각형 지역으로 부호화
  > 밝기에 변화를 주기가 어렵고 저장 용량이 커지는 경우도 있음

- 쿼드 트리 (Quad Tree)
  > 4분할 저장 구조

그래픽 기술 기초

래스터 영상정보 저장 방식 (런-렝스 예제)



- 각 셀들을 각 행마다 왼쪽에서 오른쪽으로 진행하면서 셀 값의 변경이 발생하는 경계선 부분을 찾아내 저장하는 방법
- 좌측-상단에서부터 시작하여 우측-하단으로 진행

런-렝스 코드 예시

2행 : [2, 8] -> [2,2,8]
3행 : [2, 1], [9, 1] -> [3,2,1],[3,9,1]
4행 : [2, 1], [6, 4] -> [4,2,1],[4,6,4]
5행 : [2, 1], [6, 1] -> ...
6행 : [2, 1], [6, 1] -> ...
7행 : [2, 1], [6, 3] -> ...
8행 : [2, 1], [8, 1] -> ...
9행 : [2, 7] -> ...

- [a, b, c] a=행, 번호, b=열 번호, c= 셀 수

래스터 영상정보 저장 방식

런 렝스 부호화

위키백과, 우리 모두의 백과사전.

런 렝스 부호화(Run-length encoding, RLE) 또는 **런 길이 부호화**는 매우 간단한 비손실 압축 방법으로, 데이터에서 같은 값이 연속해서 나타나는 것을 그 개수와 반복되는 값만으로 표현하는 방법이다. 이 방법은 아이콘 등의 간단한 이미지와 같이 연속된 값이 많이 있는 데이터에 효과적이다. 런 렝스 부호화는 만화나 애니메이션 등과 같이 배경의 변화가 없는 영상에 적합한 방식이다.[1] 그러나 실제적으로 잘 적용되지 못한다. 이 방식은 3번 이상 반복되는 문자들에 대해 이용되기 때문에 영어 문장에서는 잘 수행되기 어렵다.[1]

예를 들어서, 흰 바탕에 검은 글자가 나오는 스크린을 생각하면 이 스크린에는 연속된 흰 픽셀이 많이 나타날 것이다. 이러한 스크린의 한 스캔 라인이 다음과 같다고 가정하자. (흰 픽셀을 W로 표시하고 검은 픽셀을 B로 표시한다.)

```
WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW
```

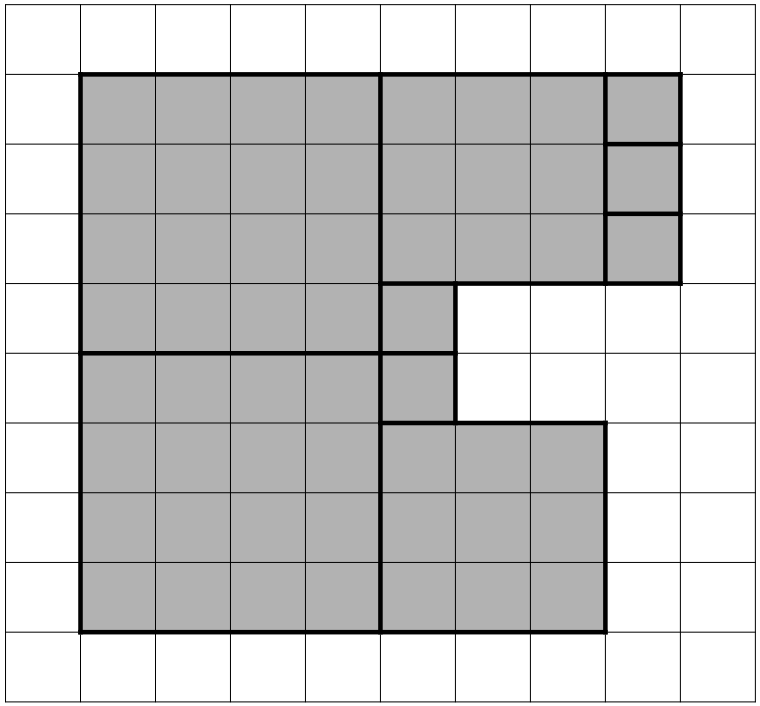
위의 데이터를 간단한 반복 길이 부호를 사용해서 압축하면 다음과 같은 결과를 얻을 수 있다.

```
12WB12W3B24WB14W
```

이는 '12개의 W, (한 개의) B, 12개의 W, 3개의 B, 24개의 W, (한 개의) B, 14개의 W'로 해석한다. 위의 예제의 경우, 압축 전에는 67글자였으나 압축 후에는 단지 16글자만으로 표현할 수 있다. 물론 실제로 이러한 이미지는 바이너리 포맷으로 저장되며 반복되는 길이를 저장하는 방법도 다양하지만, 기본적인 개념은 동일하다. 이때 데이터의 크기는 67바이트에서 16바이트로 줄었으므로 압축률은 약 4.18이다.

그래픽 기술 기초

래스터 영상정보 저장 방식(셀 부호화=블록 코딩)



- 일련의 0 또는 1의 값을 가지는 셀들을 나열하는 대신 대상이 되는 영역을 정사각형 영역으로 분할하여 해당 정사각형의 시작 좌표와 사각형의 크기로 저장

- 블록 코드로 변환하면 16단위 정사각형 2개와 9단위 정사각형 2개, 1단위 정사각형 5개로 표현

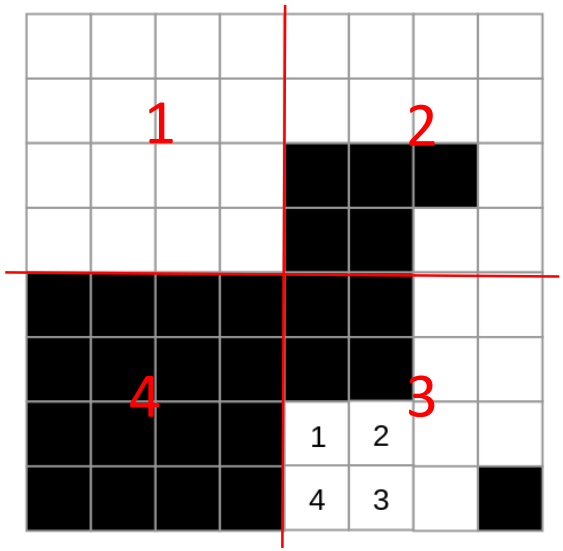
  [2,2,4], [6,2,3], [6,5,1], [9,2,1], [9,3,1], [9,4,1], [2,6,4], [6,6,1], [6,7,3]

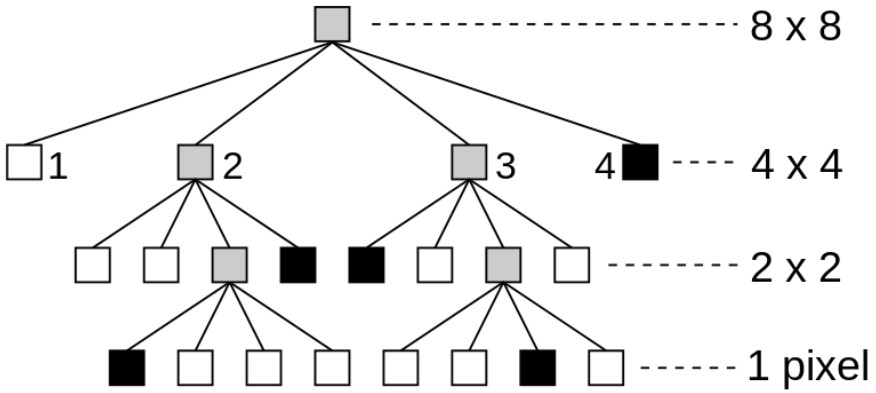
- [a, b, c]
  a = 열 번호
  b = 행 번호
  c = 정사각형 범위

그래픽 기술 기초

래스터 영상정보 저장 방식(쿼드 트리)

- 2차원 지역을 4분할(4 구역, 4 상한)하여 영상정보 저장
  > 만약 하나 상한내 픽셀들이 모두 같은 색을 지니면, 노드 내 해당 데이터 요소에 해당 색에 대한 정보 저장
  > 하나의 상한 내 픽셀들이 모두 같은 색인 경우가 아니라면, 계속 4분할 작업을 진행, 이 때에는 다음 노드를 가리키는 포인터 값 저장

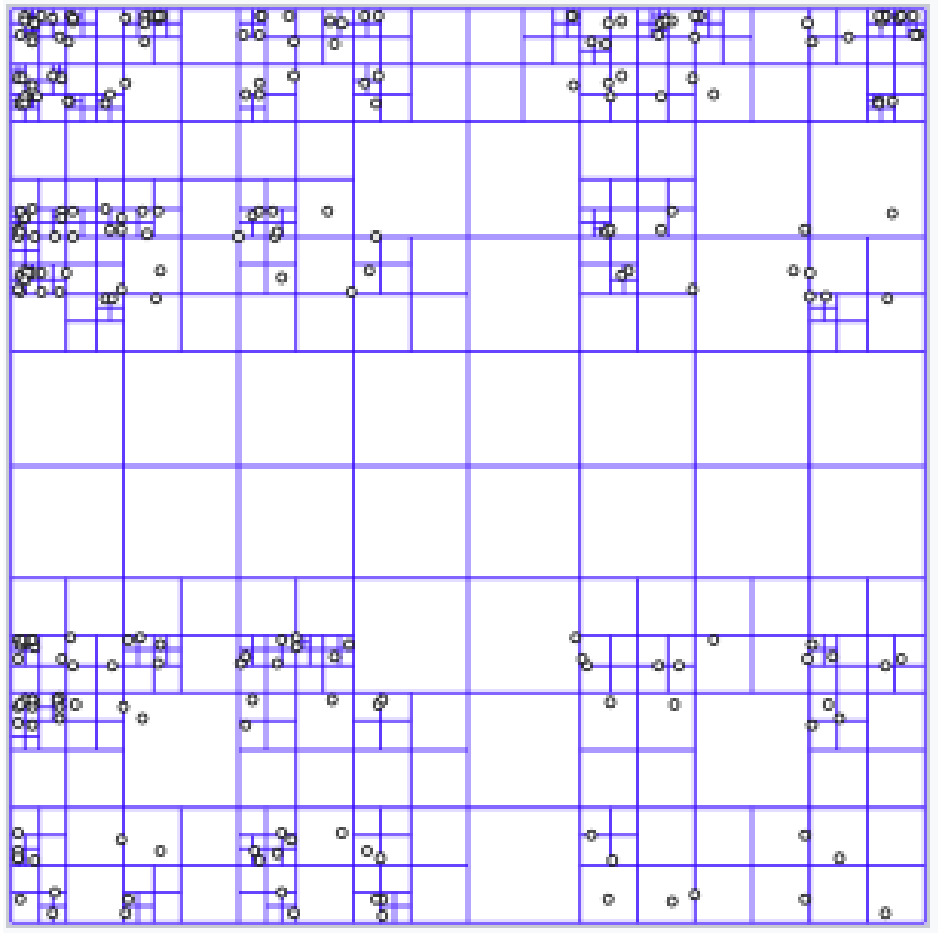




그래픽 기술 기초

래스터 영상정보 저장 방식(쿼드 트리)

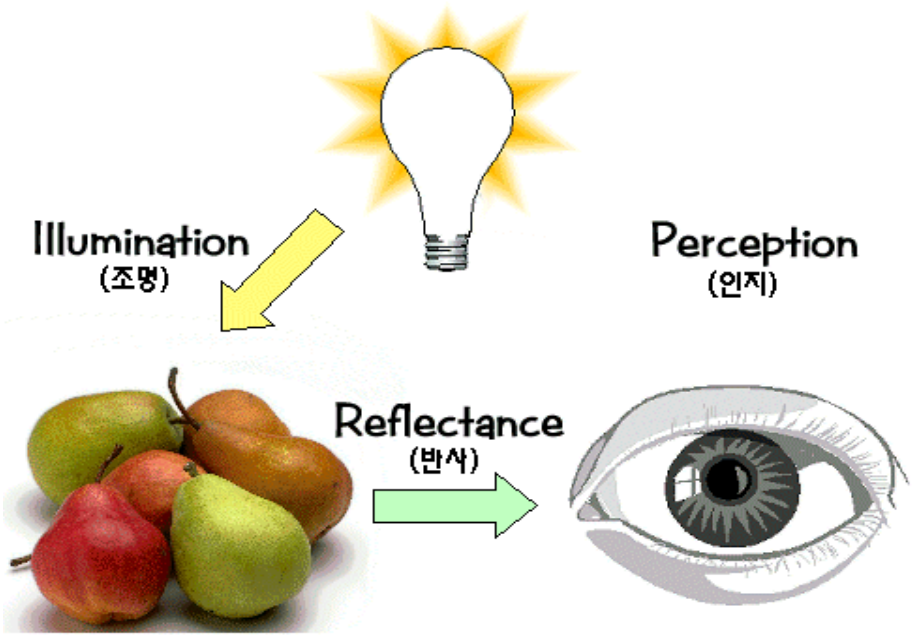
A point-region quadtree with point data.

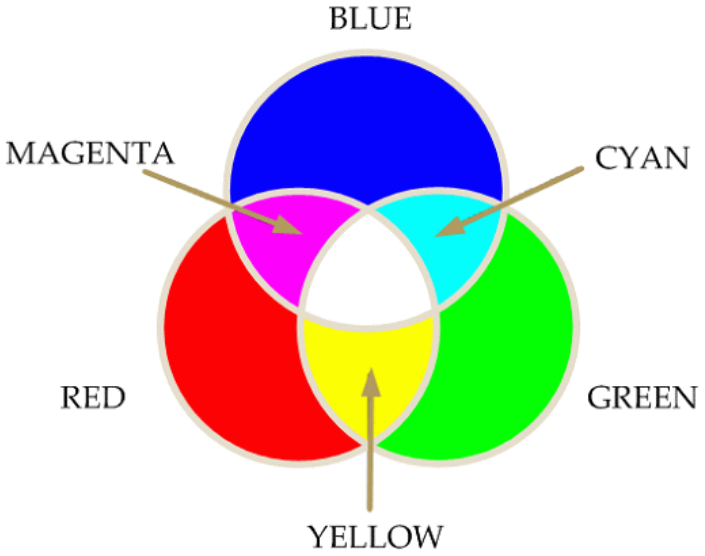


컬러 모델

원리

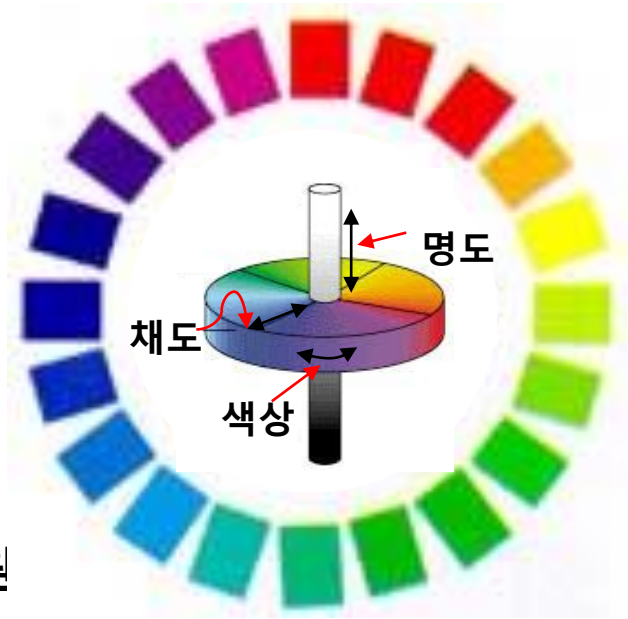
- 빛이 존재하기 때문에 색상 구분 가능
- 삼원색 (Primary Color) : 적색, 녹색, 청색





컬러 모델

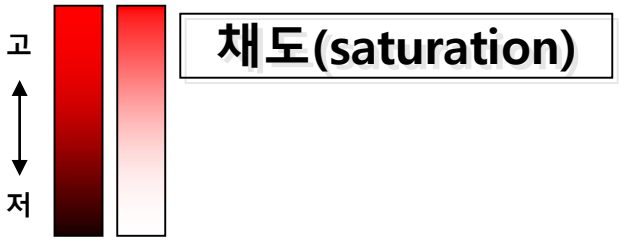
빛의 성질



- 태양이나 전구와 같은 광원은 가시영역에 있는 모든 주파수를 발산 (백색 광원을 생성)

- 빛의 밝기 (Brightness) :
  > 빛의 강도와 관계 있음
  > 빛의 강도(에너지)가 높으면 높을수록 밝은 광원

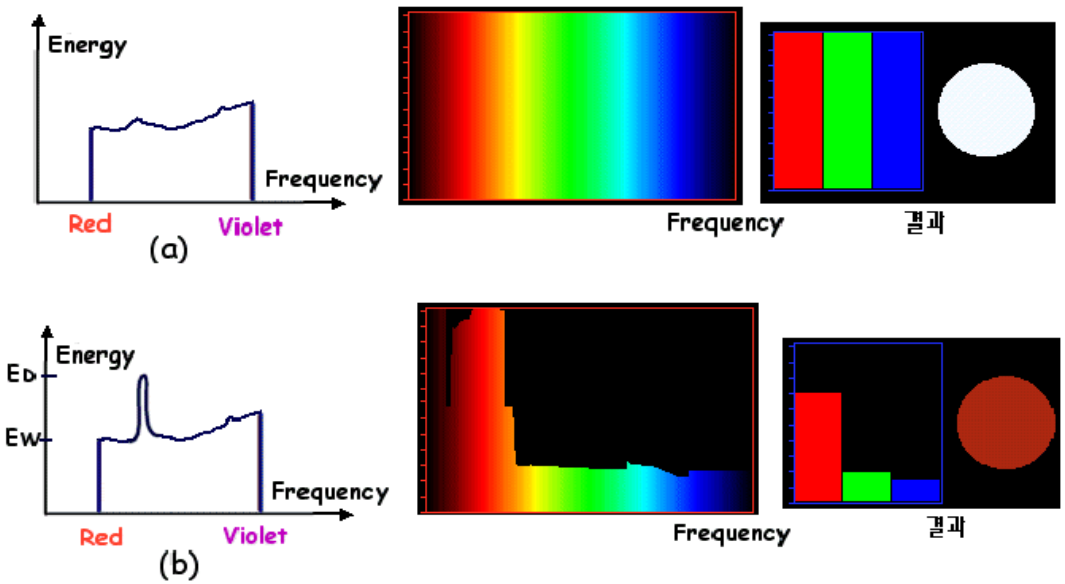
- 채도
  > 빛의 순수함 or 농도
  > 파스텔 색조나 회색 빛이 우세한 색은 채도가 낮다고 말함

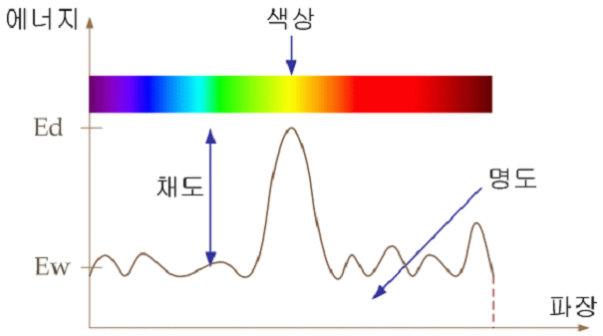


컬러 모델

백색광선에서 나오는 에너지

- 붉은 색(Red)에서 보라색(Violet) 범위 내 주파수 부문에서 나오는 에너지가 균등할 때와 우월 주파수가 있는 경우





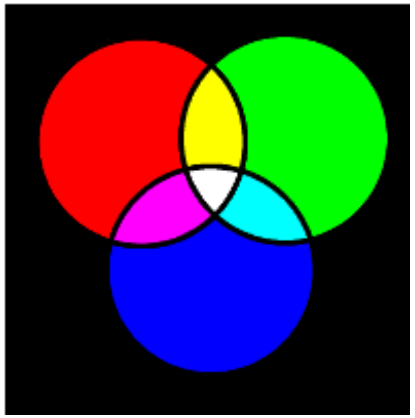
ED > EW 빛은 맑아짐
EW = 0 일 때 채도 100%
EW = ED 일 때 채도 0%

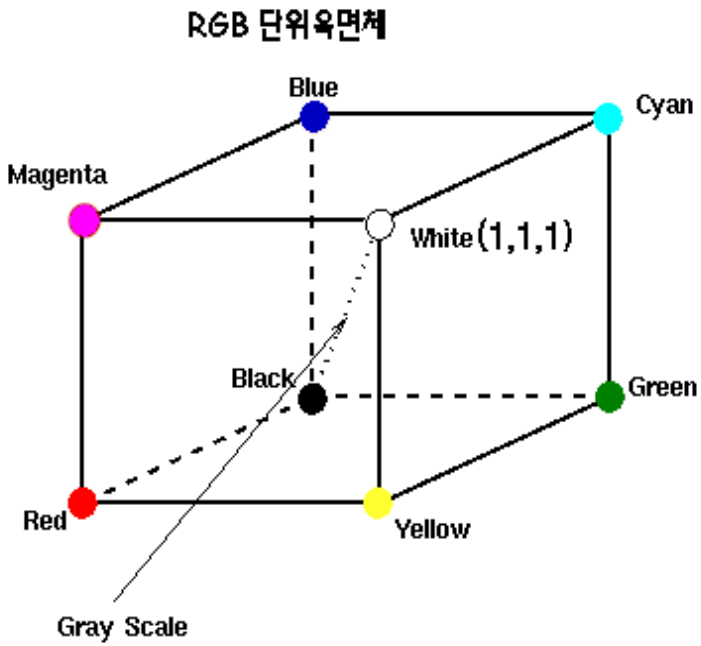
- ED : 우월 주파수에서 나오는 에너지
- EW: 다른 주파수 영역에서 흰 빛을 내는데 기여하는 에너지
- ED / EW 차이에 따라 채도 결정

컬러 모델

RGB 칼라 모델 (가산 혼합)

- R, G, B 축 상 단위 육면체로써 모델을 표현
  > 원점 좌표(0, 0, 0) – 검은색 , (1, 1, 1) – 흰색
  > 단위 육면체의 축 상에 위치한 꼭지점은 기본 색,
  나머지 꼭지점은 각 기본 색에 대한 보색

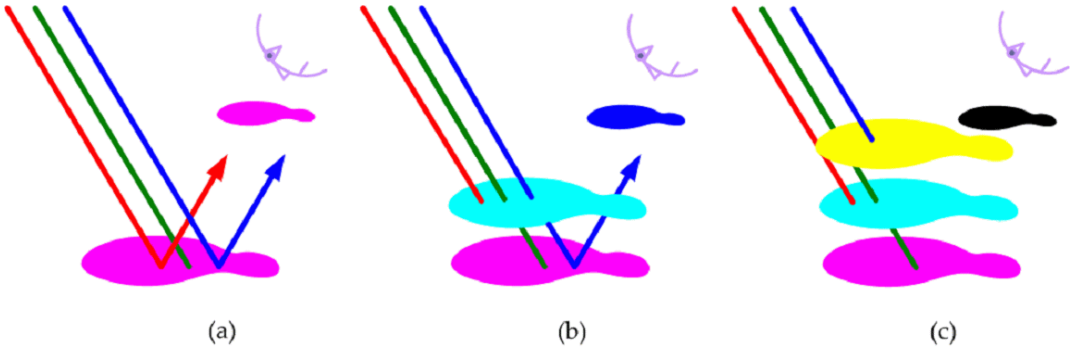




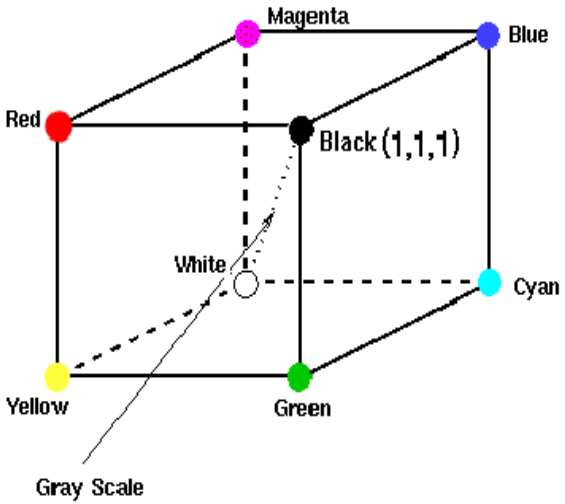
컬러 모델

CMY 칼라 모델 (감산 혼합)

- Cyan, Magenta, Yellow 사용
- 반사되는 빛에 의하여 색을 인식



W - G(Complement of Magenta) = R + B = Magenta
(W - G) - R(Complement of Cyan) = Blue
(W - G - R) - B(Complement of Yellow) = Black

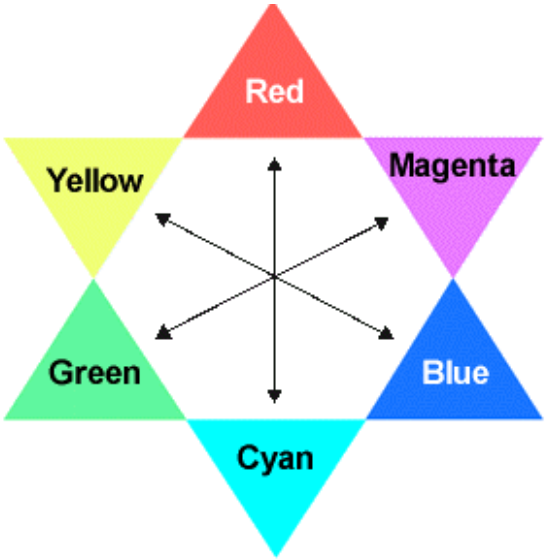


컬러 모델

각 모델 변환

- RGB 값을 CMY 모델 값으로 전환도 가능하고
  CMY 값을 RGB 값으로 전환 가능함

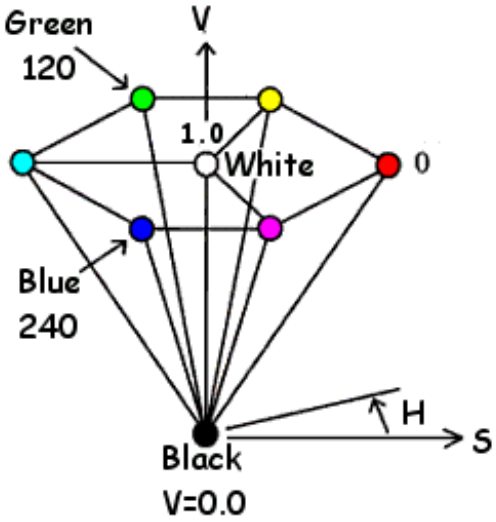
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} W \\ W \\ W \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} B \\ B \\ B \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

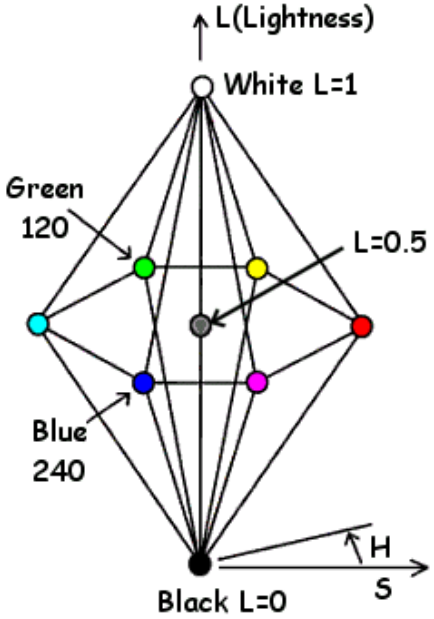


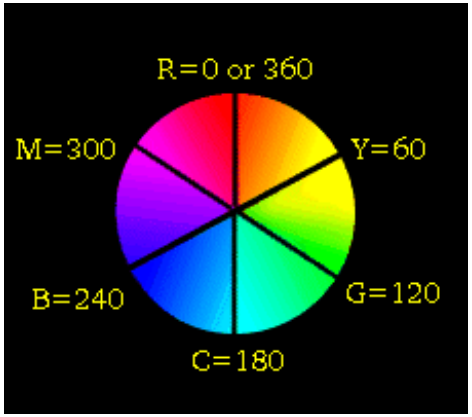
컬러 모델

HSV 컬러 모델

- 단위 육면체를 흰색 꼭지점을 잇는 주 대각선을 따라서 바라볼 때 단위 육면체의 윤곽이 육각형 모양으로 나타남
- 육각형 경계선이 각 색을 나타냄
- 농도: 수평축에 의하여 측정
- 값 : 육면체 뿔의 중심을 지나는 수직 뿔에 의하여 측정







컬러 모델

HSV 컬러 모델은 왜?

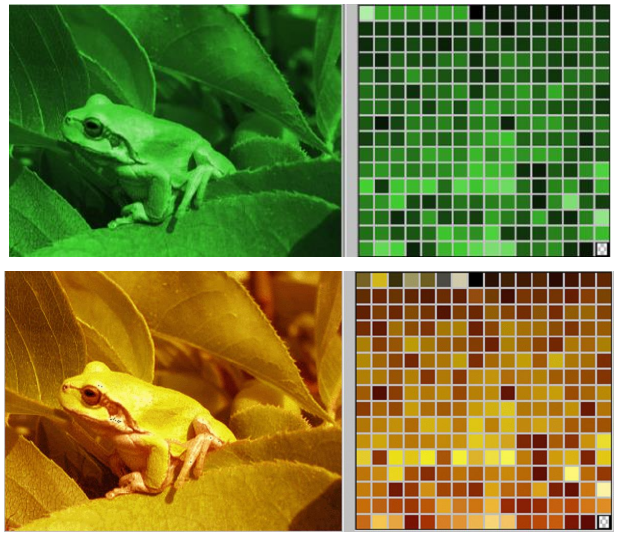
- RGB 모델은 직관적이지 않음. 보라색을 만들기 위해 R, G, B 값을 어느정도 적용을 해야하는가?

- HSV(Hue, Saturation, Value)는 색상, 채도, 명도에 대한 값을 이용
- Hue : 수직 축을 기준으로 0도인 적색부터 360도까지의 회전각도 표시
  육각형의 각 변은 60도 간격으로 두고, 황색 60도, 녹색 120도, 보색 관계는 180도씩 간격
- Saturation: 0 ~ 1
- Value : 0 ~ 1
- v와 s가 모두 1이면 가장 순수한 색이됨
- 흰색은 v=1, s=0일 경우

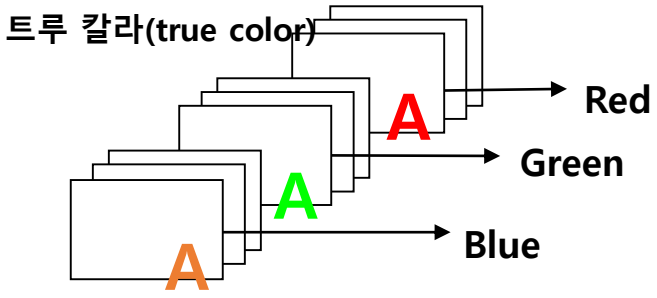
컬러 모델

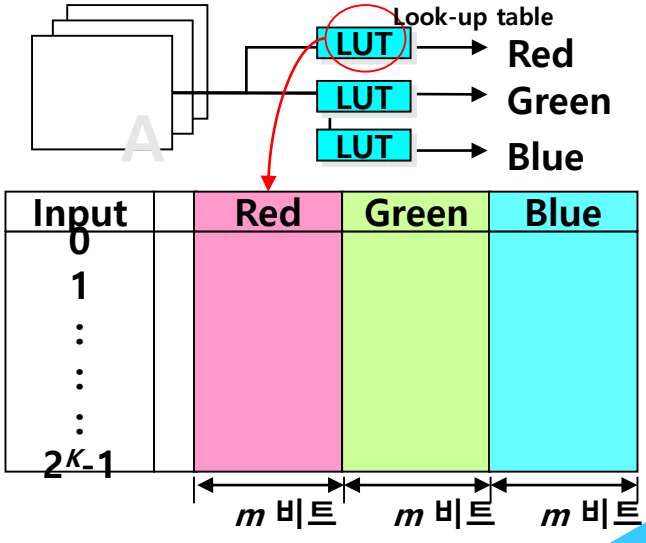
Index 컬러 체계

- 컬러 수를 줄이기 위한 가장 대표성이 있는 컬러를 선정하여 사용



프레임버퍼 크기 또는 요구되는 메모리가 너무 크다.





CULT(color look-up table)

사용한 색 또는 사용 가능한 색을 색인화(indexed)하여 테이블 형태로 모아 놓은 것

컬러 모델

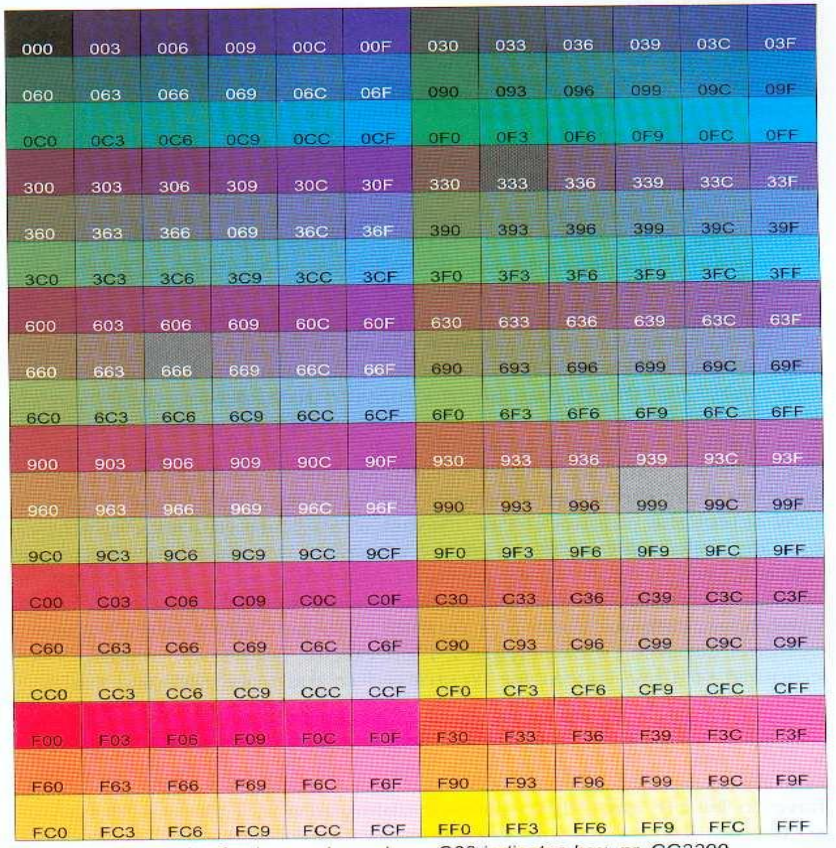
Hexadecimal 컬러

10진수	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	…
16진수	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	…

- 0-255를 2개의 digit으로 표현

- 컬러 코드 의미? (#CCFF33)
  CC = 16*12+12 = 204
  FF = 16*15+15 = 255
  33 = 16*3 + 3 = 51
  RGB = (204, 255, 51)
  #CCFF33 -> CF3

- 만약 00, 33, 66, 99, CC, FF 만으로 Color 팔레트를 만드는 경우
  216 가지의 색상만 가능



Example of colour code: code nr. C39 indicates hex. nr. CC3399