



미니 프로젝트(Ver 3.0)

# Server-Side Programming

Digital Image Processing

박영호

([blog.naver.com/hkpyh](http://blog.naver.com/hkpyh))

# Contents

01

Project 개발 환경 및 범위

02

Server-Side Programming 모식도

03

JSP 주요 소스코드 및 활용 예시

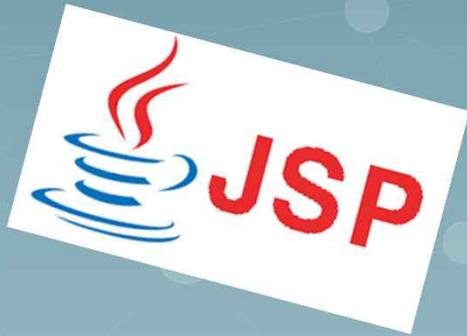
04

JavaScript → Java 소스코드 변환

05

결론 & 추후 계획

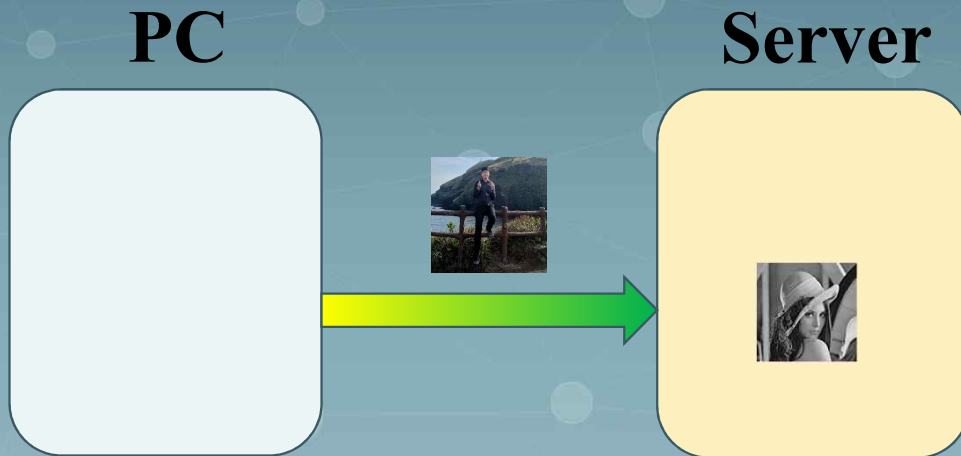
# Project 개발 환경 및 범위



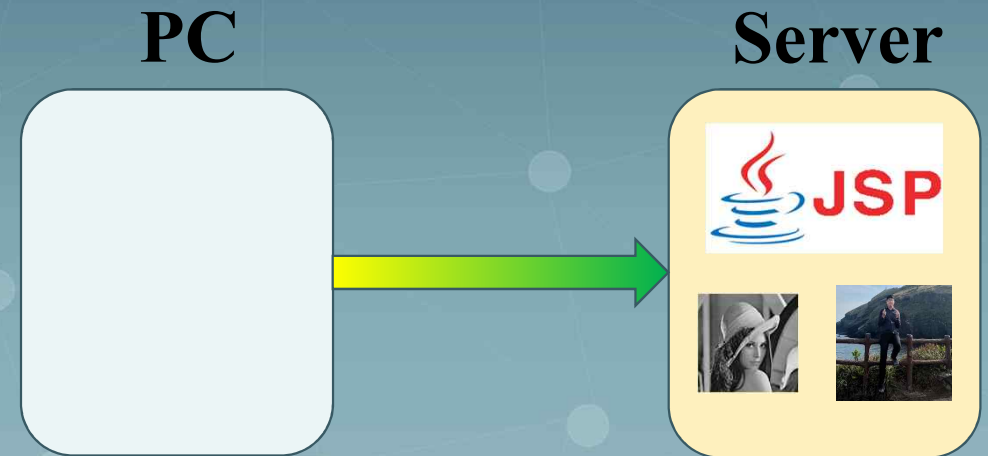


# Server-Side Programming 모식도

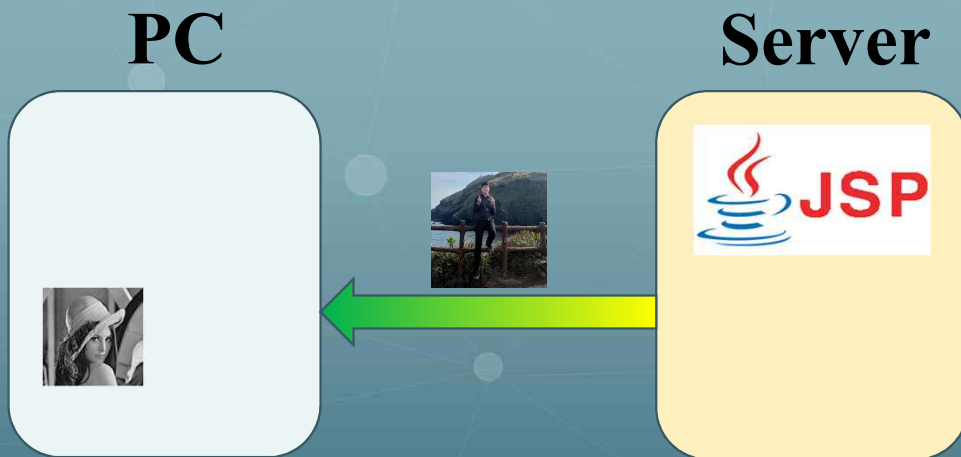
## 1) (PC → Server) File Upload



## 2) Server 영상 처리



## 3) (Server → PC) File Download



## \* Server 영상 처리 메뉴 \*

### Digital Image Processing(Color)

----- Algorithm ----- ▾

Parameter1 :

Parameter2 :

Upload File :  선택된 파일 없음

[서버 흑백\(RAW\)](#)

picture28.jpg 영상 처리 완료 !!

[!! 다운로드 !!](#)



# JSP 주요 소스코드 및 S/W 활용 예시

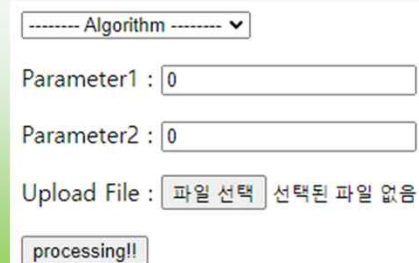
# Image Processing Code - JSP

```
// (0) 파라미터 넘겨 받기
MultipartRequest multi = new MultipartRequest
    (request, "c:/upload", *1024*1024, "utf-8",
     new DefaultFileRenamePolicy());

// (주의) 파라미터 순서가 반대
String tmp;
Enumeration params = multi.getParameterNames();
tmp = (String) params.nextElement();
para2 = multi.getParameter(tmp);
tmp = (String) params.nextElement();
para1 = multi.getParameter(tmp);
tmp = (String) params.nextElement();
algo = multi.getParameter(tmp);

// File // 여러개 파일 가능
Enumeration files = multi.getFileNames();
//첫 파일 1개
tmp = (String) files.nextElement();
//파일명 추출
String filename = multi.getFilesystemName(tmp);
```

PC → Server 파일 Upload 위한 파라미터 선언



사용자(PC)의 입력을  
서버로 전송하기 위한  
변수 선언

Upload 후 파일명을 가져오기 위한 코드



# Image Processing Code - JSP

```
// (1) 입력 영상 파일 처리
inFp = new File("c:/upload/" + filename);
BufferedImage bImage = ImageIO.read(inFp);

// (2) 파일 --> 메모리
// (중요!) 입력 영상의 폭과 높이를 알아내야 함!
long len = inFp.length();
inW = bImage.getHeight();
inH = bImage.getWidth();

// 메모리 할당
inImage = new int[3][inH][inW];
```

- upload 파일 이름 선언
- 'Image' Class 활용, 파일 內 사진 정보 입력

사진 파일의 폭, 높이를 변수(inW, inH)에 입력

칼라 영상을 위한 3차원 메모리 할당(RGB, 가로, 세로)



# Image Processing Code - JSP

```
// 읽어오기
for(int i=0; i<inH; i++) {
    for (int k=0; k<inW; k++) {
        // ex) F377D6 --> 0000F3
        int rgb = bImage.getRGB(i, k);

        // F377D6 ---> 0000F3 & 0000FF ==> F3
        int r = (rgb >> 16) & 0xFF;

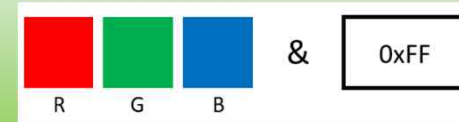
        // F377D6 ---> 00F377 & 0000FF ==> 77
        int g = (rgb >> 8) & 0xFF;

        // F377D6 ---> F377D6 & 0000FF ==> D6
        int b = (rgb >> 0) & 0xFF;

        inImage[0][i][k] = r;
        inImage[1][i][k] = g;
        inImage[2][i][k] = b;
    }
}
```

- 각 좌표에서 값 추출 후 변수에 입력(getRGB)

- 각 좌표의 값을 r, g, b로 나누어서 입력
- r, g, b로 값을 나누기 위해 0xFF와 & 연산 수행



- r, g, b 값을 입력 영상 배열(inImage)에 입력

# Image Processing Code - JSP

```
// (4) 결과를 파일로 저장하기
outFp = new File("c:/out/out_" + filename);
BufferedImage oImage // Empty Image
= new BufferedImage(outH, outW, BufferedImage.TYPE_INT_RGB);

//Memory --> File
for (int i=0; i< outH; i++) {
    for (int k=0; k<outW; k++) {
        int r = outImage[0][i][k]; // F3
        int g = outImage[1][i][k]; // 77
        int b = outImage[2][i][k]; // D6
        int px = 0;
        px = px | (r << 16); // 000000 | F30000 => F30000
        px = px | (g << 8); // F30000 | 007700 => F37700
        px = px | (b << 0); // F37700 | 0000D6 => F37766

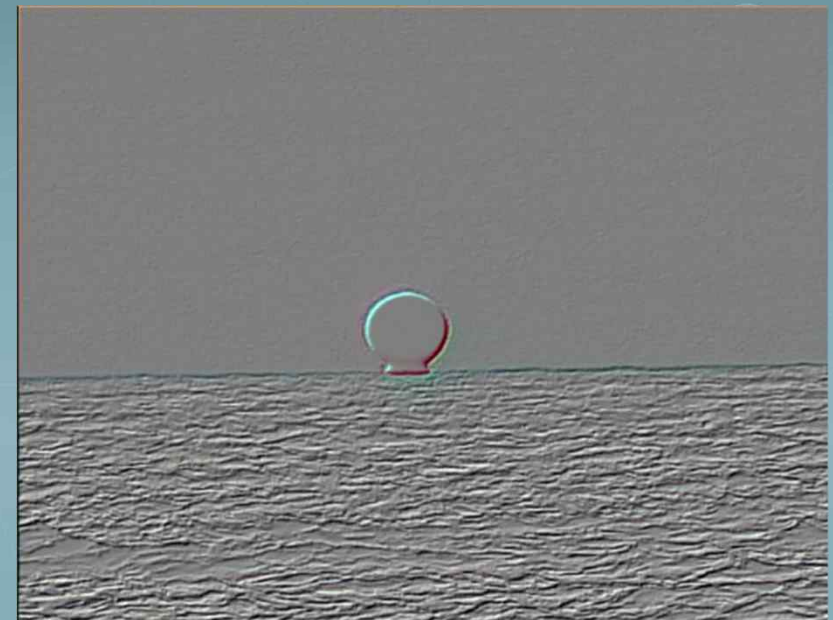
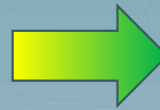
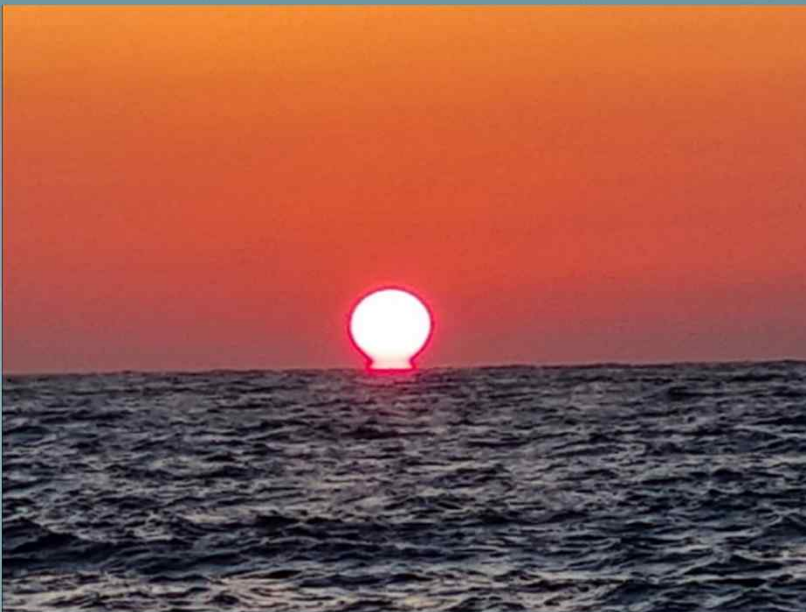
        oImage.setRGB(i, k, px);
    }
}
ImageIO.write(oImage, "jpg", outFp);
```

영상처리 한 결과를 서버에 저장하기 위한  
BufferedImage 설정

oImage에 setRGB(Image class) 활용,  
각 좌표의 값 저장

oImage 로 저장된 파일을 jpg 형식으로 저장

# Image Processing 결과 - 엠보싱



- Mini\_Project 1,2와 동일한 알고리즘 활용
- 출력 영상 = 입력 영상 x 엠보싱 마스크(3x3)  
곱의 합으로 표현

-1	0	0
0	0	0
0	0	1

$$Output\_pixel[x,y] = \sum_{m=(x-k)}^{x+k} \sum_{n=(y-k)}^{y+k} (I[m,n] \times M[m,n])$$

- Output\_pixel[x, y]: 회선 처리로 출력한 화소
- I[m, n]: 입력 영상의 화소
- M[m, n]: 입력 영상의 화소에 대응하는 가중치

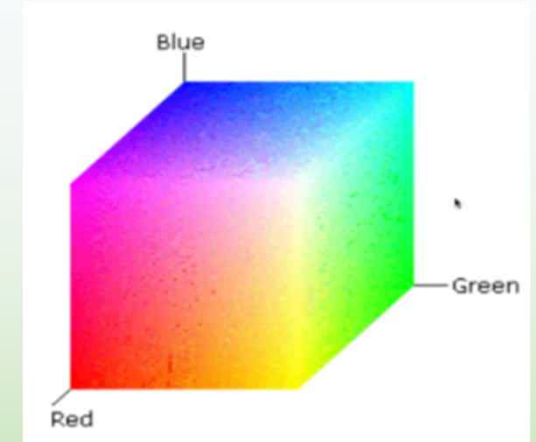


JavaScript → Java  
소스코드 변환

# \*\* RGB vs HSV \*\*

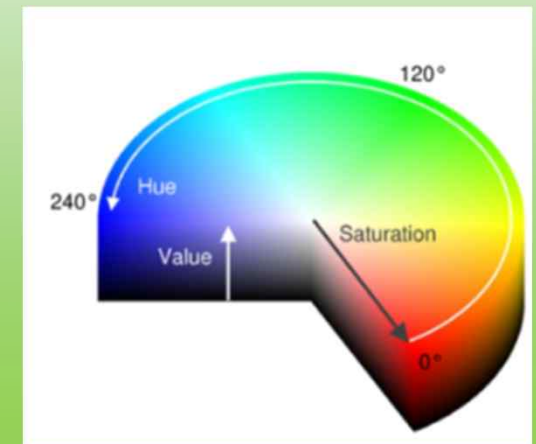
- RGB : Red(빨강), Green(초록), Blue(파랑)

- 빛의 3원색. 색의 혼합으로 표시
- 모니터, TV 등에 표시할 때 사용
- \* 인쇄 : RGB를 CMYK로 변환하는 과정 필요



- HSV : Hue(색), Saturation(진함), Value(밝기)

- 빛의 3원색. 색의 혼합으로 표시
- 본 프로젝트에서 색을 추출할 때 H값 활용
- \* 0: 빨강 ~ 360: 보라



# RGB to HSV

```
public float[] rgb2hsv
(float r, float g, float b) {

    float max1 = Math.max(r,g);
    float max2 = Math.max(g,b);
    float max = Math.max(max1,max2);
    float min1 = Math.min(r,g);
    float min2 = Math.min(g,b);
    float min = Math.min(min1, min2);
    float d = max - min;
    float h=0, s;
    float v = max / 255;

    if (max==0)
        s = 0;
    else
        s = d/max;

    if(max==min){
        h = 0; }
}
```

```
else if (max==r) {
    h = (g - b) + d *
        (g < b ? 6: 0); h /= 6 * d; }
else if (max==g) {
    h = (b - r) + d * 2; h /= 6 * d; }
else{
    h = (r - g) + d * 4; h /= 6 * d; }

hsv[0] = (float) (h);
hsv[1] = (float) (s);
hsv[2] = (float) (v);

return hsv;
}
```

< JavaScript로 작성된 RGB to HSV 함수를 JSP로 변환 >

구 분	JavaScript	JSP
변 수	let, var로 선언 가능	float, int 등 변수에 맞게 선언
return 값	여러 개 가능	1개만 가능(배열 활용)



# HSV to RGB

```
public float[] hsv2rgb(float h, float s, float v){
    float r=0, g=0, b=0, f, p, q, t;
    h=h*360; s=s*100; v=v*100;

    h = Math.max(0, Math.min(360, h));
    s = Math.max(0, Math.min(100, s));
    v = Math.max(0, Math.min(100, v));

    h /= 360;    s /= 100;    v /= 100;
    int i = (int) Math.floor(h * 6);
    f = h * 6 - i;
    p = v * (1 - s);
    q = v * (1 - f * s);
    t = v * (1 - (1 - f) * s);
```

< JavaScript로 작성된 HSV to RGB 함수를 JSP로 변환>

\* 주의사항

- JavaScript에서 switch~case 문으로 작성 → if 문으로 변경

▶ switch~case은 '정수'만 가능하므로 값에 오류 발생

```
if(i%6==0){
    r = v; g = t; b = p; }
else if(i%6==1){
    r = q; g = v; b = p; }
else if(i%6==2){
    r = p; g = v; b = t; }
else if(i%6==3){
    r = p; g = q; b = v; }
else if(i%6==4){
    r = t; g = p; b = v; }
else if(i%6==5){
    r = v; g = p; b = q; }

rgb[0] = (float) r*255;
rgb[1] = (float) g*255;
rgb[2] = (float) b*255;

return rgb;
```





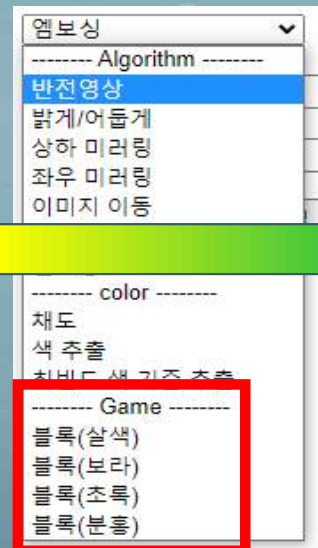
이미지 영상처리  
실제 활용 예시

# 게임 – 각 블록의 색깔 인식 및 추출

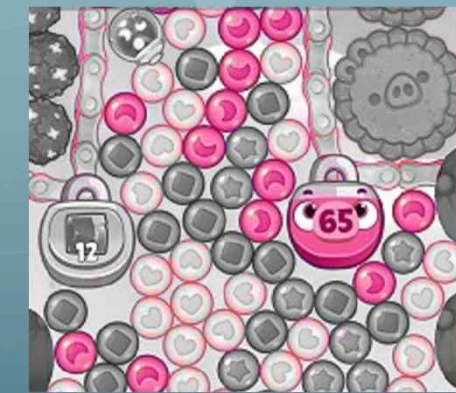
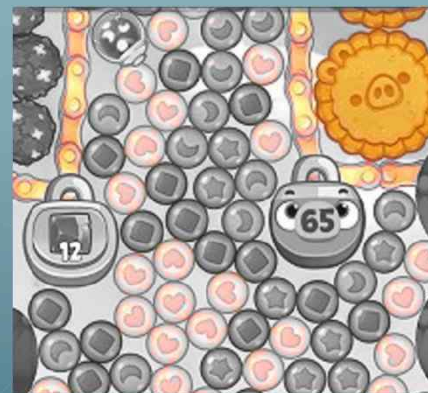
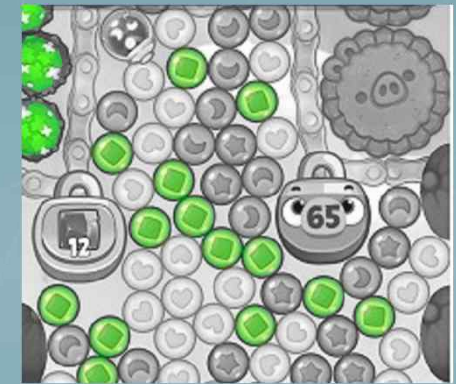
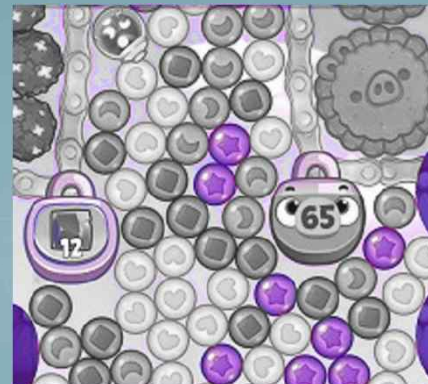
- 게임명 : Dream Blast
- S/W 적용 결과 : 우측 그림과 같이 색깔별 블록 추출 가능(Mini Project 1,2와 동일)



실제 게임 화면



## 각 블록의 색깔만 인식 및 추출





# 결론 & 추후 계획

# 결론 & 추후 계획

'22. 9월

JavaScript  
환경 구성  
(완료)

'22. 9월 **현재 시점**

Server-Side  
Programming  
(완료)

(추후 예정)

Web에 구현 및  
상용화

'22. 9월

흑백, 칼라 영상  
Code 작성(완료)

'22. 11월

동영상에  
각종 효과 구현

## <개선 필요 사항>

- 상용화를 위한 추가 기능 개발(추출 영상 자동 터치 S/W 개발 및 구현)
- 홈페이지 개설 및 사용자 편의를 위한 화면 구성



THANK YOU