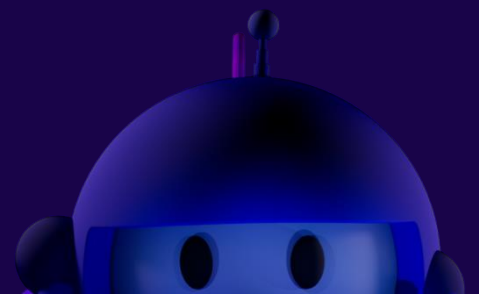
GPU를 이용한 공간 보간 Heat map 구현

김준형

Agenda

1. 구현 배경
2. Heat map 구현
3. GPU를 이용한 성능 개선
4. C# 언어 기반 GPU 연산
5. 결과 분석 및 마무리



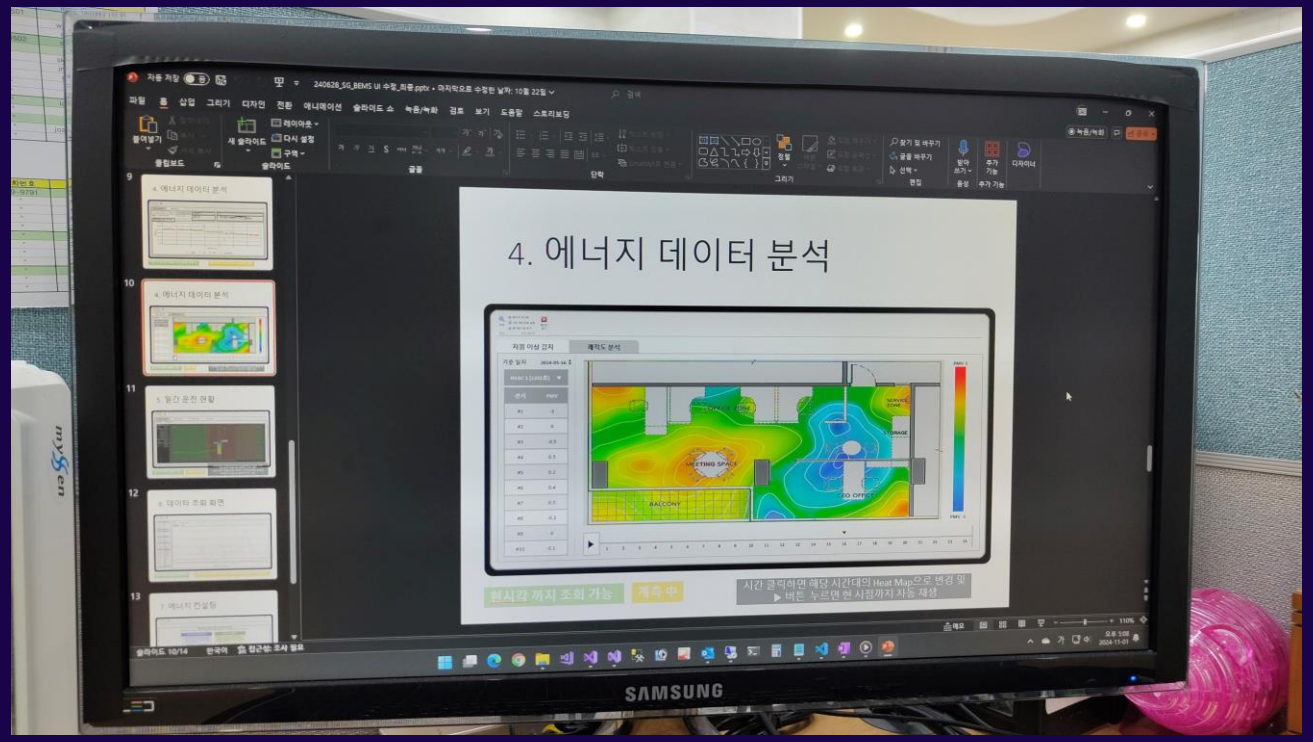
01

구현 배경

구현하게 된 계기

고객사 요구사항

- 연구실에 온/습도 센서가 7개 설치되어 있음
- 온/습도를 이용한 PMV를 시계열 DB에 기록
- PMV를 연구실 평면도에 Heat map으로 표현
- 시간별 Heat map을 연속 영상으로 재생
- PMV 수준에 따라 등고선 표현
- 기 구축된 시스템에 요구사항 추가 구현
  - WPF 기반 클라이언트 프로그램
  - .NET Framework 4.6.1 사용

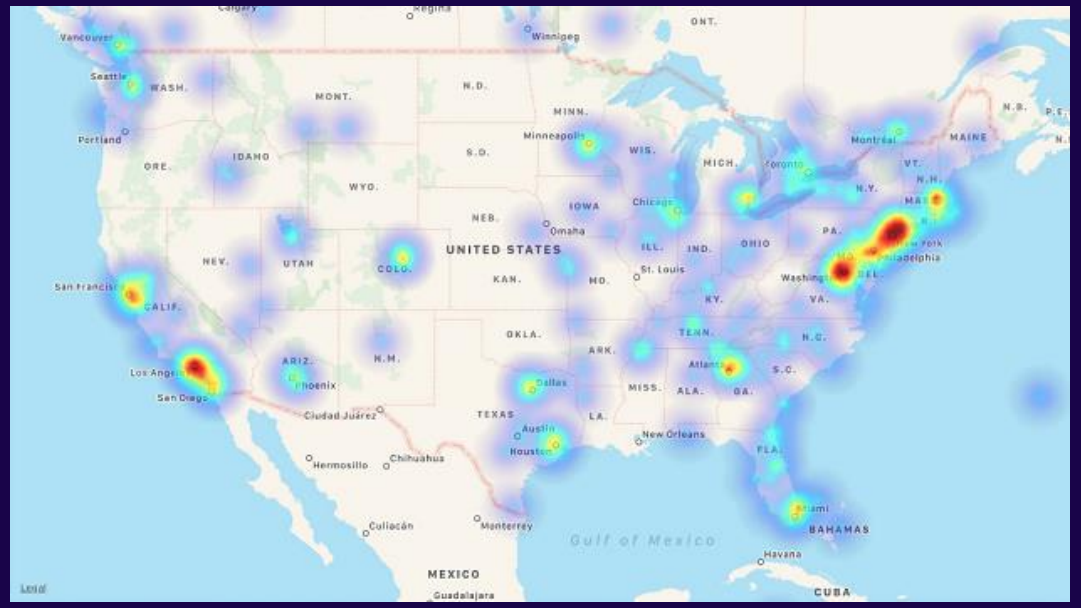


Heat map

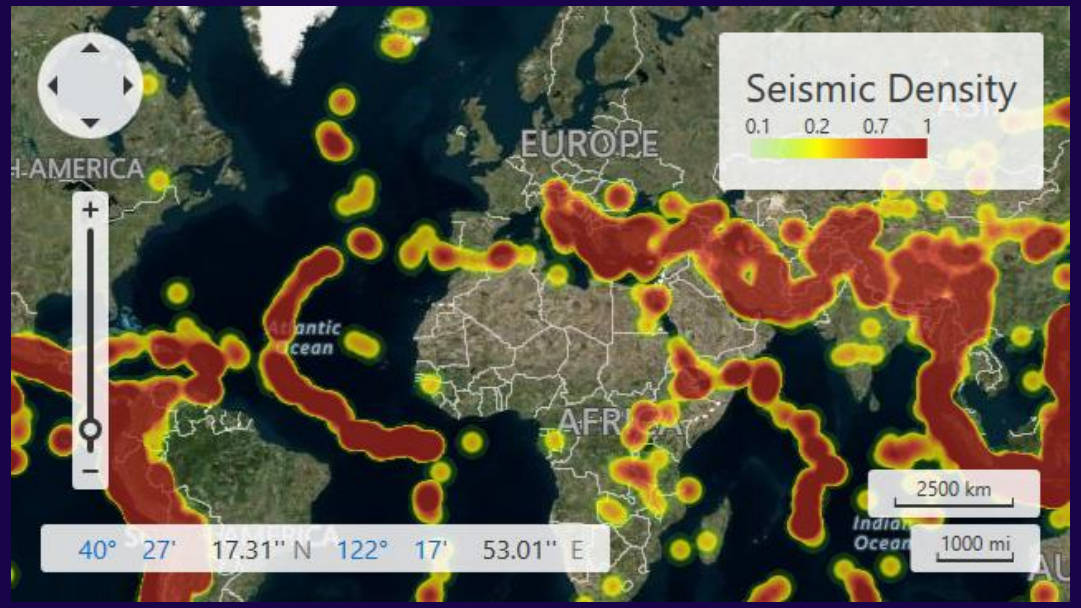
- 열을 뜻하는 heat와 지도를 뜻하는 map을 결합시킨 단어로, 색상으로 표현할 수 있는 다양한 정보를 일정한 이미지 위에 열 분포 형태의 그래픽으로 출력하는 기술

상용 Heat map 컴포넌트 예시

INFRAGISTICS®



DevExpress®



돈

“없으면 만들어 쓴다.”

02

Heat map 구현

Heat map 구현 절차

공간 보간 알고리즘 구현

색상 변환 및 등고선 구현

WPF 기반 렌더링 구현

공간 보간

- 공간 보간(Spatial Interpolation)이란 몇 개의 미리 알려진 위치의 샘플 값들을 이용하여 알려지지 않은 위치들의 값을 추정하는 방법

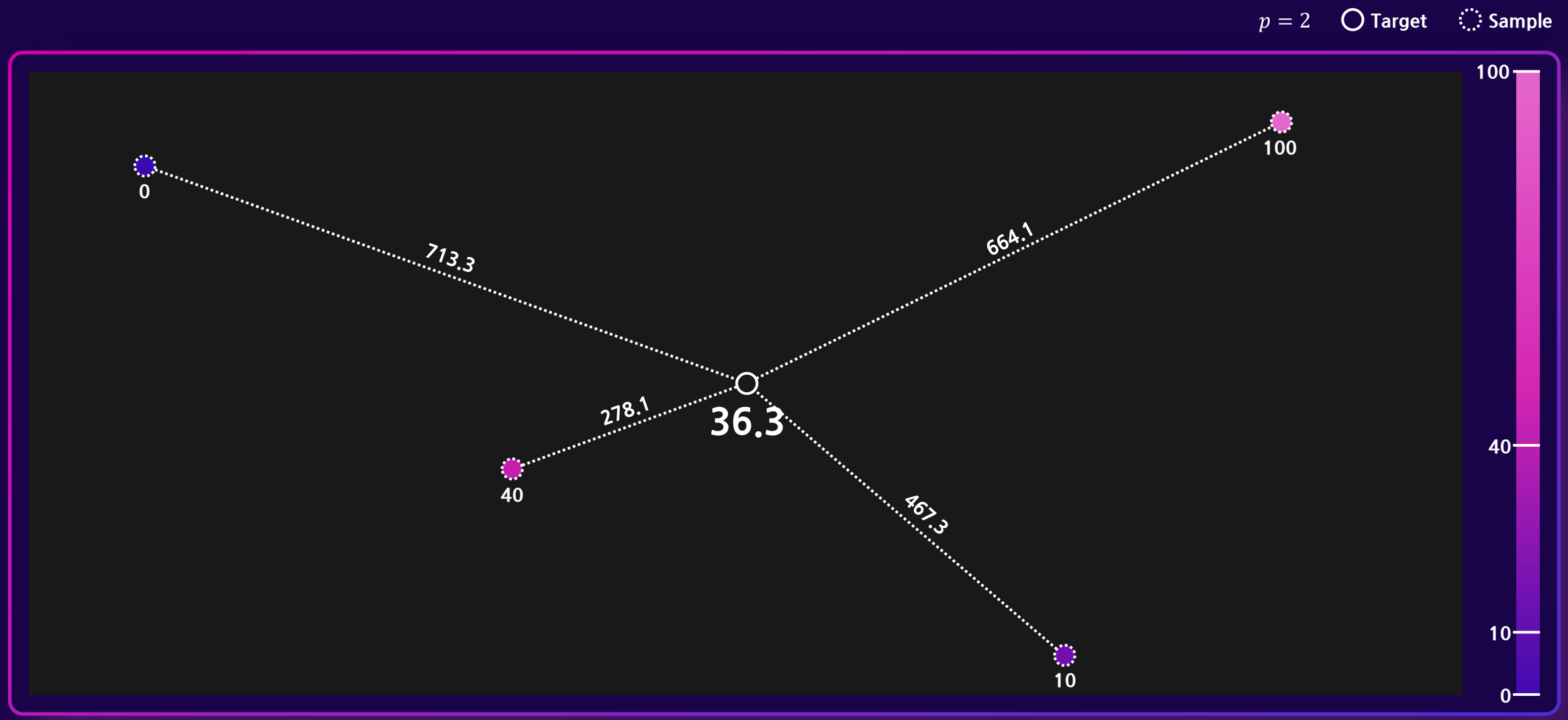
IDW 공간 보간

- 역 거리 가중(Inverse Distance Weighting) 공간 보간
- 특정 샘플의 위치가 계산하고자 하는 위치로부터 멀어질수록 해당 샘플의 영향력이 다른 샘플에 비해 상대적으로 낮아지도록 가중치를 부여함

$$w_i(X) = \frac{1}{d(X, X_i)^p}$$

$$u(X) = \frac{\sum_{i=1}^{N} w_i(X) u_i}{\sum_{i=1}^{N} w_i(X)}$$

IDW 공간 보간 예시



$$\left(0 \times \frac{1}{713.3^2} + 100 \times \frac{1}{664.1^2} + 40 \times \frac{1}{278.1^2} + 10 \times \frac{1}{467.3^2}\right) \div \left(\frac{1}{713.3^2} + \frac{1}{664.1^2} + \frac{1}{278.1^2} + \frac{1}{467.3^2}\right)$$

공간 샘플 구조체 정의

C#

```
public readonly struct SpatialSample
{
    public SpatialSample(float x, float y, float value)
    {
        X = x;
        Y = y;
        Value = value;
    }

    public float X { get; }
    public float Y { get; }
    public float Value { get; }
}
```

IDW 공간 보간 구현

C#

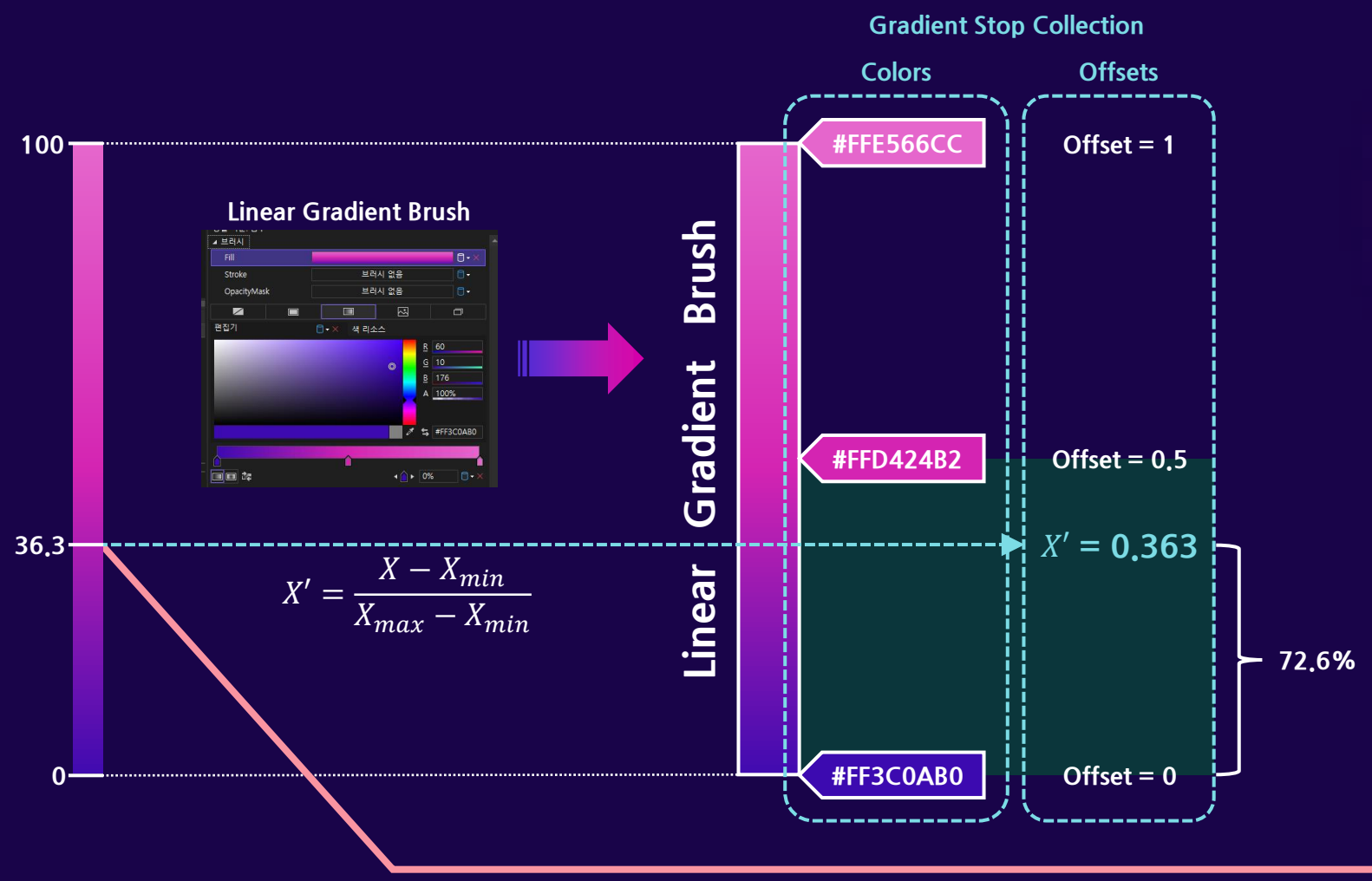
```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    int width = values.GetLength(1);
    int height = values.GetLength(0);

    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            var sum = 0f;
            var weights = 0f;
            foreach (var sample in samples)
            {
                var distance = MathF.Sqrt(MathF.Pow(x - sample.X, 2) + MathF.Pow(y - sample.Y, 2));
                if (distance <= searchRadius)
                {
                    var weight = distance == 0 ? 1 : (1 / MathF.Pow(distance, weightPower));
                    sum += sample.Value * weight;
                    weights += weight;
                }
            }
            values[y, x] = sum / weights;
        }
    }
}
```

$$d(A,B) = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

$$w_i(X) = \frac{1}{d(X, X_i)^p}$$

$$u(X) = \frac{\sum_{i=1}^{N} w_i(X) u_i}{\sum_{i=1}^{N} w_i(X)}$$

보간 값의 색상 변환



Alpha compositing

$$C = A \times \alpha + B \times (1 - \alpha)$$

$A =$ #FFD424B2

$B =$ #FF3C0AB0

$\alpha = 0.726$

$\therefore C =$ #FFBA1FB1 (in scRGB)

색상 변환 메서드 예시

C#

```
float ToOffset(float value, float maximum, float minimum)
    => MathF.Max(0f, MathF.Min(1f, (value - minimum) / (maximum - minimum)));

Color ToColor(float offset, Color[] colors, float[] offsets)
{
    var ranges = colors.Length - 1;

    int index;
    for (index = 0; index < ranges; index++)
        if (offset < offsets[index])
            break;

    if (index == 0 || offsets[index] <= offset) return colors[index];

    var range = offsets[index] - offsets[index - 1];
    var alpha = (offset - offsets[index - 1]) / range;
    var colorA = colors[index];
    var colorB = colors[index - 1];

    return colorA * alpha + colorB * (1 - alpha);
}
```

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

색상 구간 탐색

색상 구간 외 처리

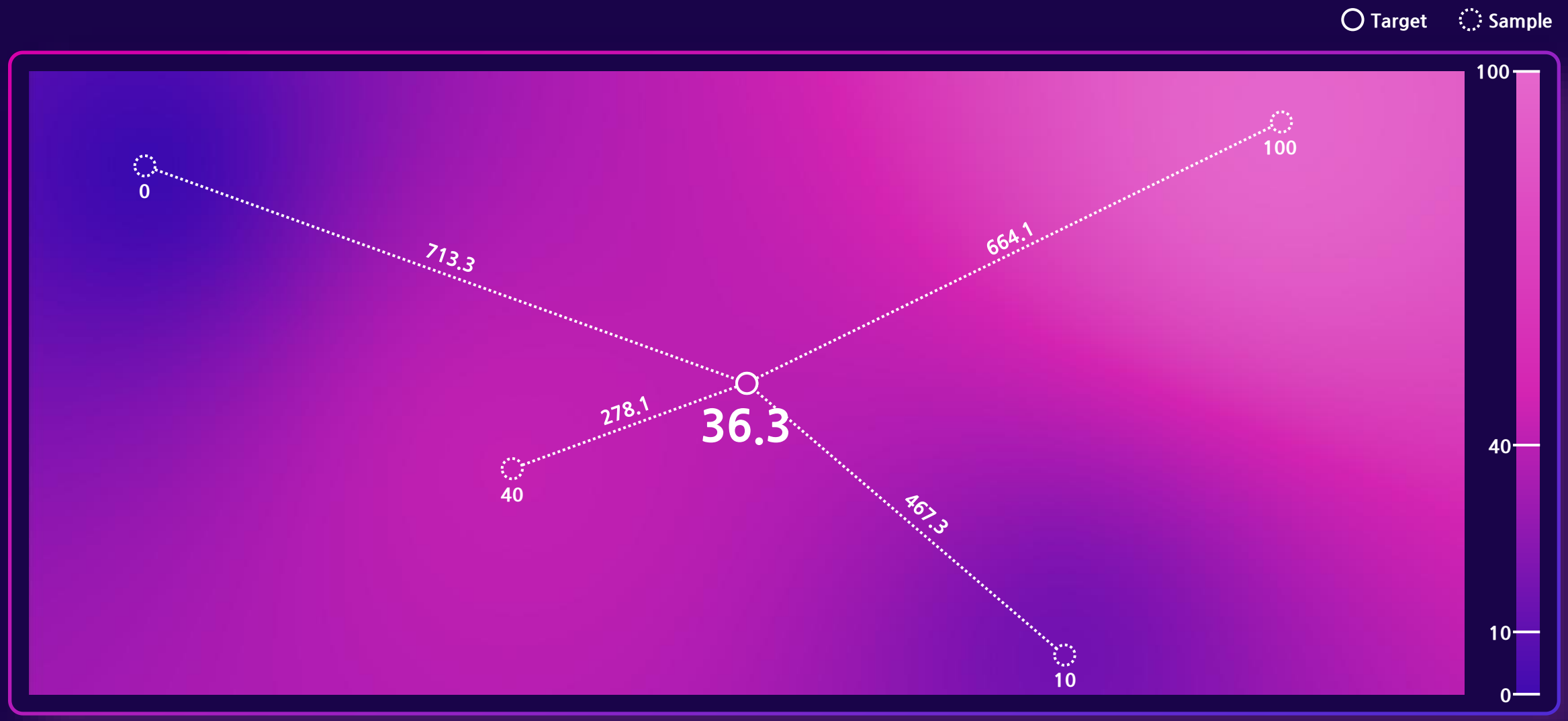
α

A

B

$$C = A \times \alpha + B \times (1 - \alpha)$$

보간 값의 색상 변환 결과



scRGB 색 영역 설정

C#

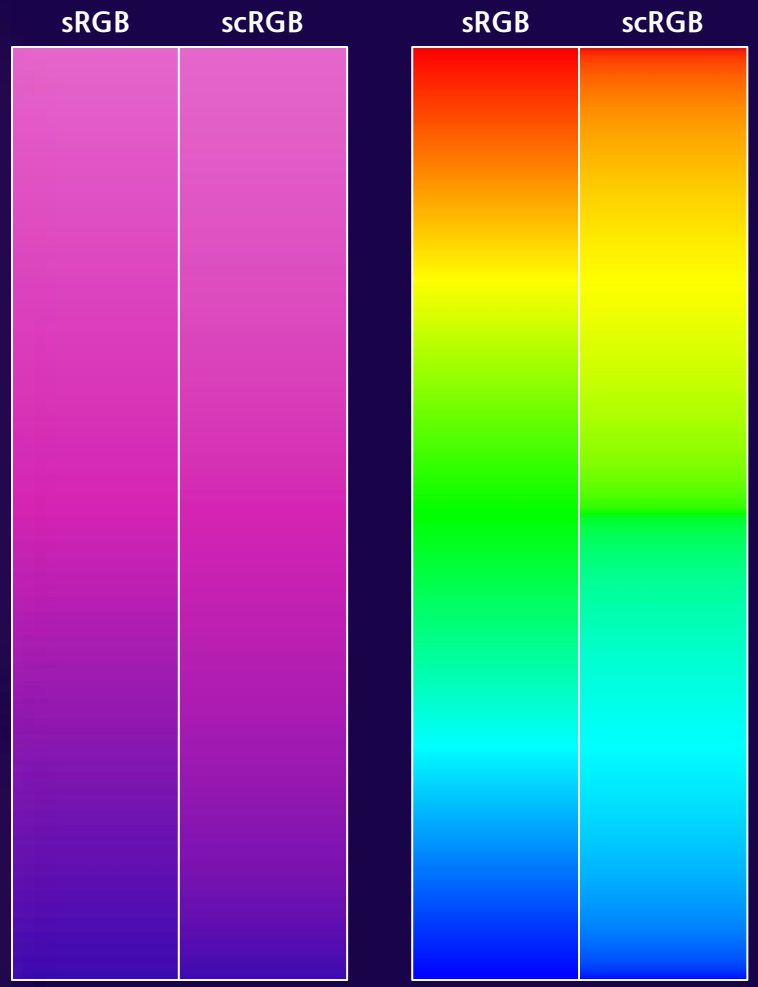
```
public static Color operator *(Color color, float coefficient)
{
    Color c1 = FromScRgb(color.scRgbColor.a * coefficient, color.scRgbColor.r * coefficient,
        color.scRgbColor.g * coefficient, color.scRgbColor.b * coefficient);

    if (color.context == null)
    {
        return c1;
    }
    else
    {
        c1.context = color.context;

        #pragma warning suppress 6506 // c1.context is obviously not null
        c1.ComputeNativeValues(c1.context.NumChannels);
    }
    return c1;
}
```

$$C = A \times \alpha + B \times (1 - \alpha)$$

출처: referencesource.microsoft.com

XAML

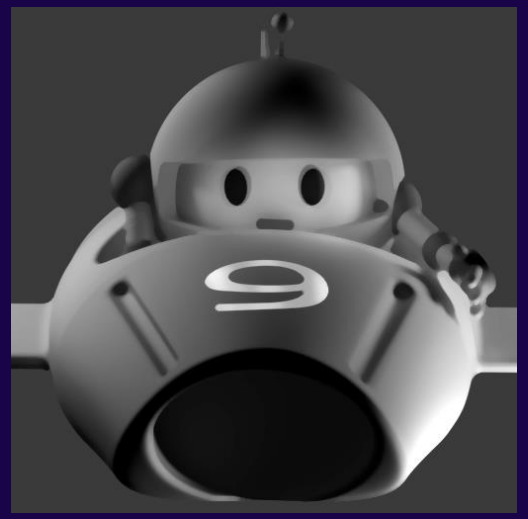
```
<LinearGradientBrush StartPoint="0,1" EndPoint="0,0" ColorInterpolationMode="ScRgbLinearInterpolation">
    <GradientStop Color="#FFE566CC" Offset="1"/>
    <GradientStop Color="#FFD424B2" Offset="0.5"/>
    <GradientStop Color="#FF3C0AB0" Offset="0"/>
</LinearGradientBrush>
```



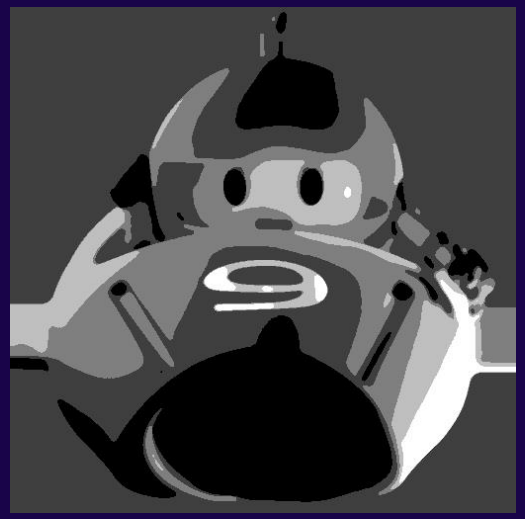
등고선

- Contour line, 같은 값을 가지는 위치들을 연결한 선

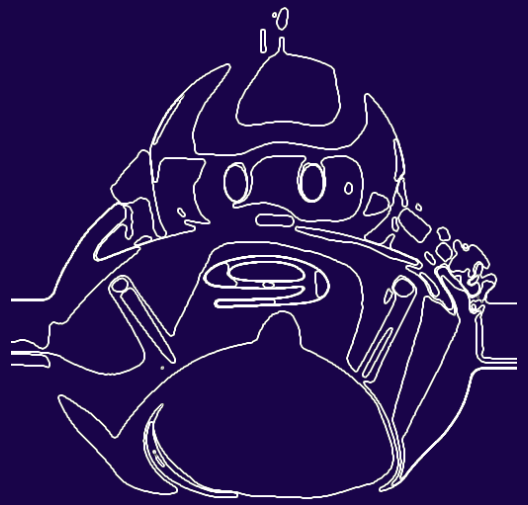
등고선 생성을 위한 영상처리 개요



1. Posterization



2. Edge detection



3. Merge

Posterization

C#

```
float Posterize(float offset, int levels)
    => (MathF.Floor(offset * levels) + 0.5f) / levels;
```

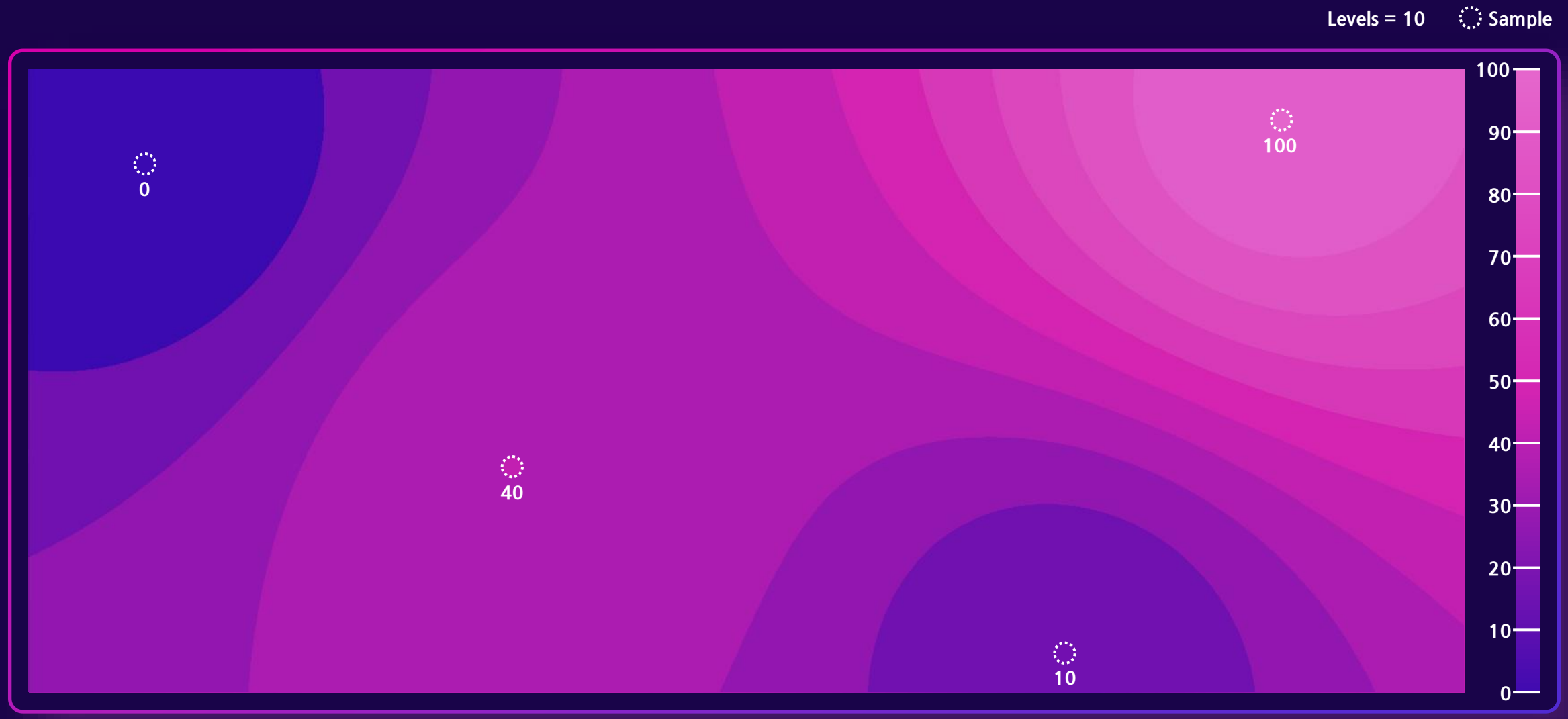
$$X' = \frac{\lfloor X \times L \rfloor + 0.5}{L}$$

C#

```
float Posterize(float offset, int levels)
    => MathF.Floor(offset * levels) / levels;
```

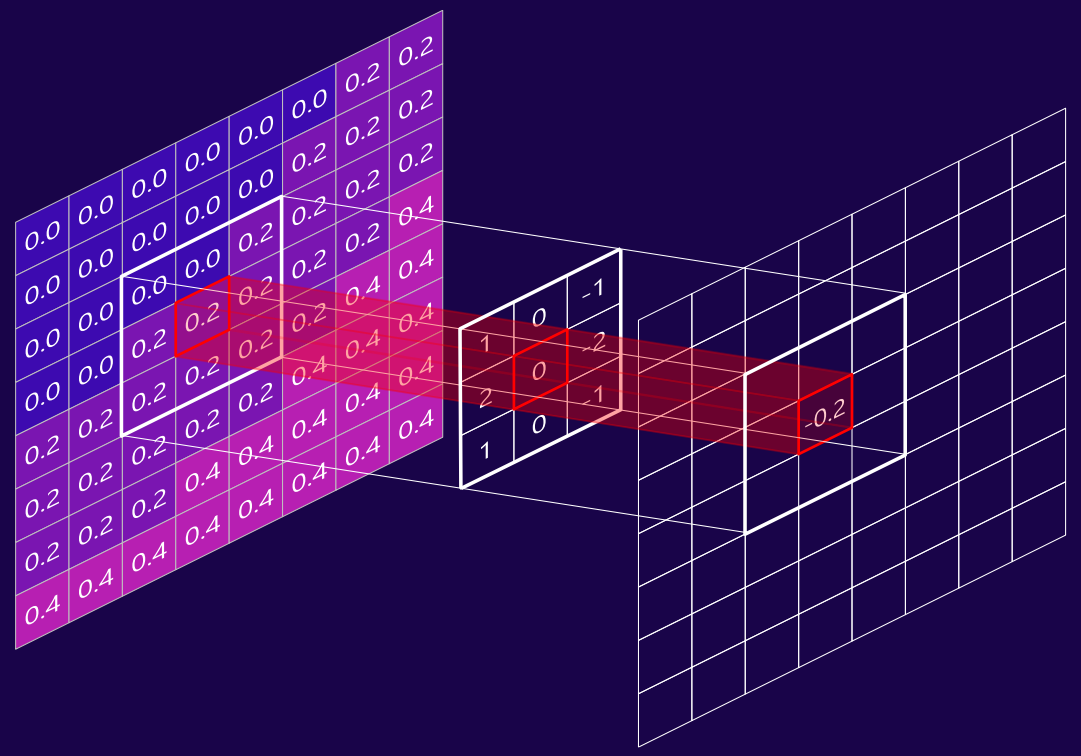
$$X' = \frac{\lfloor X \times L \rfloor}{L}$$

Posterization 예시



Edge detection: Sobel 필터

- 경계선 검출 기법 중 하나로, 가장 단순하고 쉬운 방법
- 2개의 3×3 커널로 Convolution(합성곱, 회선) 후 결합하여 Edge 검출

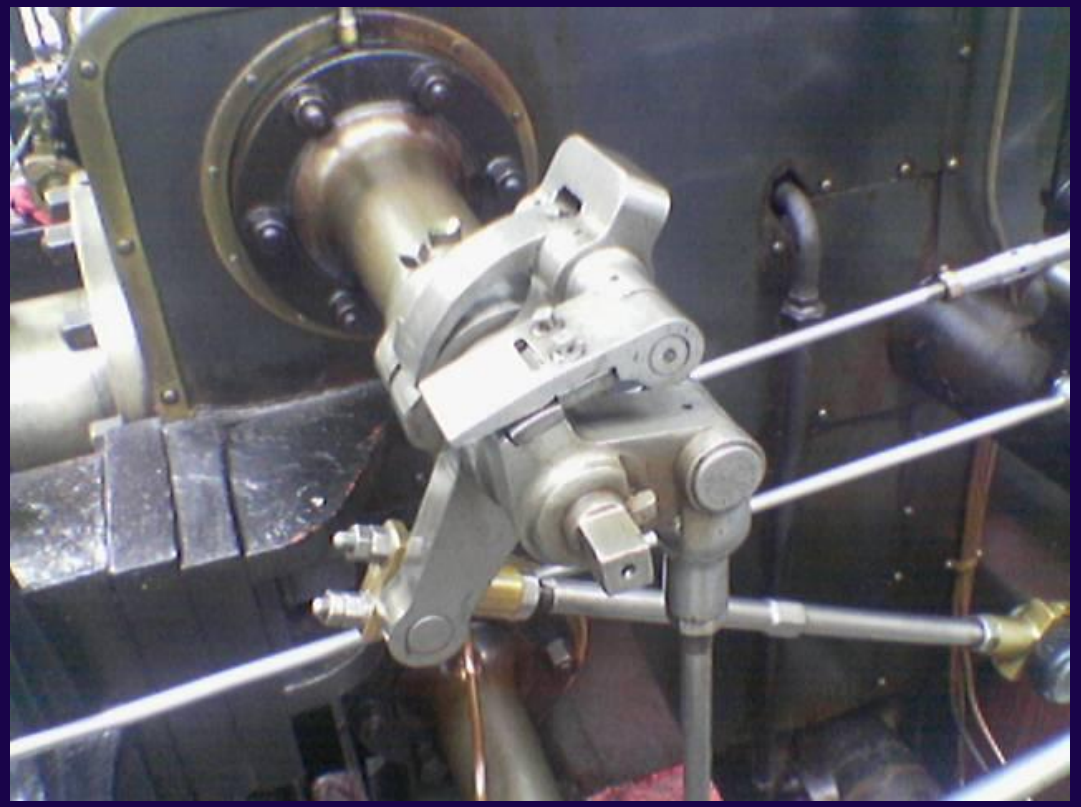


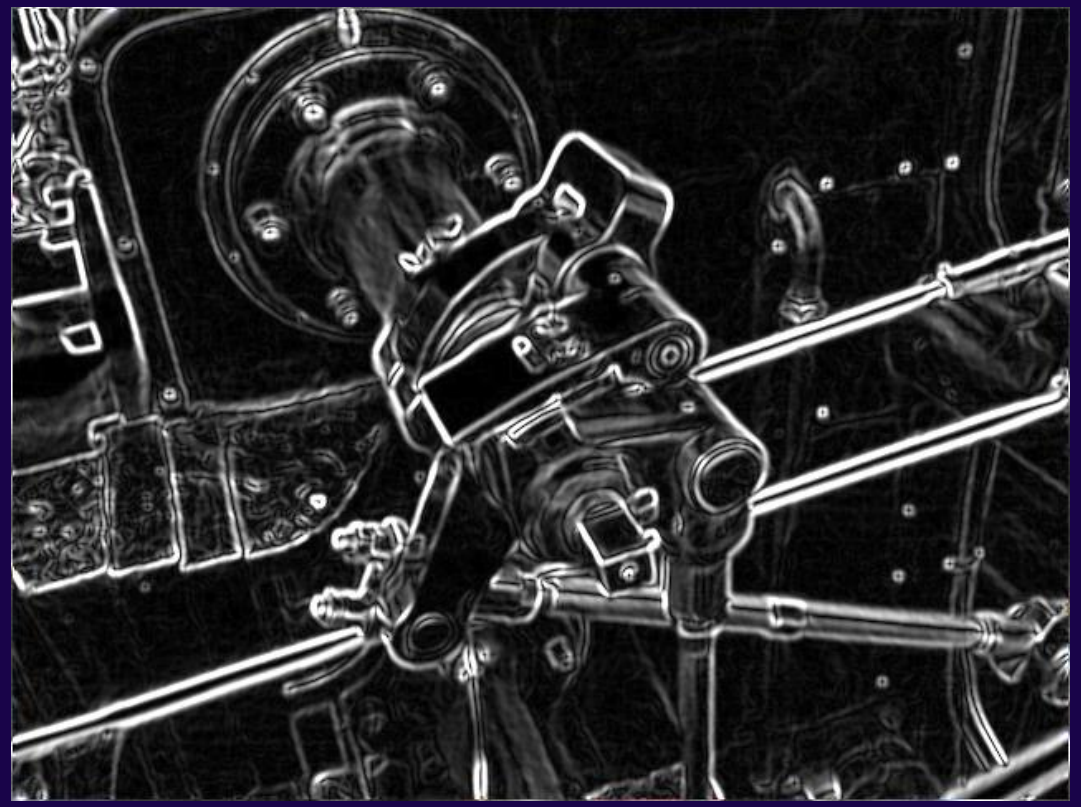
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

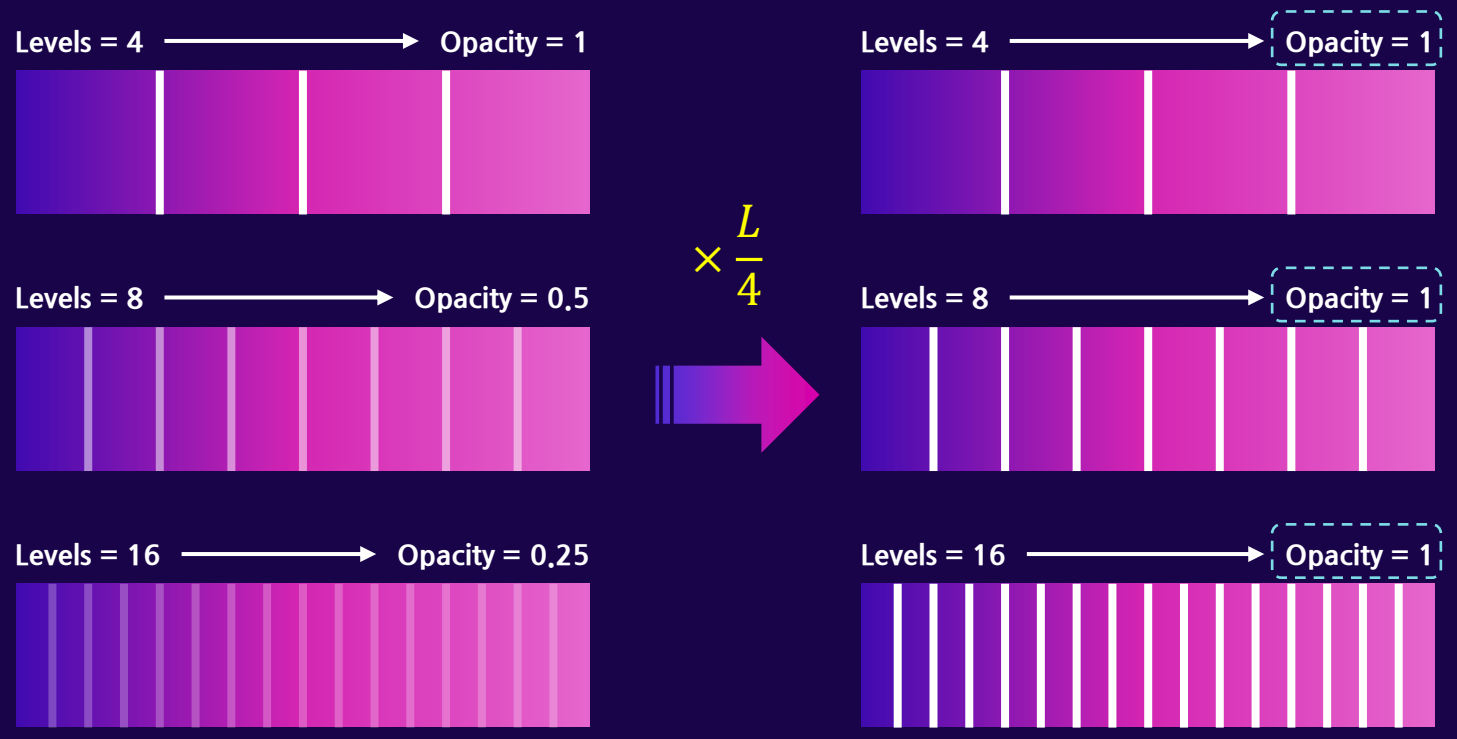
Sobel 필터 적용 예시





등고선을 위한 Sobel 필터 개선

- 경계선의 강도는 Posterization 단계 수에 반비례
- 일정한 강도의 등고선 표시를 위해 보정 필요



$$G_x = \frac{L}{4} \times \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \frac{L}{4} \times \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Sobel 필터를 응용한 등고선 검출

C#

```
float GetLineOpacity(float[,] poster, int currX, int currY, int levels)
{
    var prevX = currX + (currX <= 0 ? 0 : -1);
    var prevY = currY + (currY <= 0 ? 0 : -1);
    var nextX = currX + (currX >= poster.GetLength(1) - 1 ? 0 : 1);
    var nextY = currY + (currY >= poster.GetLength(0) - 1 ? 0 : 1);

    var gX = poster[prevY, prevX] - poster[prevY, nextX]
        + (poster[currY, prevX] - poster[currY, nextX]) * 2
        + poster[nextY, prevX] - poster[nextY, nextX];

    var gY = poster[prevY, prevX] - poster[nextY, prevX]
        + (poster[prevY, currX] - poster[nextY, currX]) * 2
        + poster[prevY, nextX] - poster[nextY, nextX];

    gX *= levels / 4f;
    gY *= levels / 4f;

    return MathF.Sqrt(gX * gX + gY * gY);
}
```

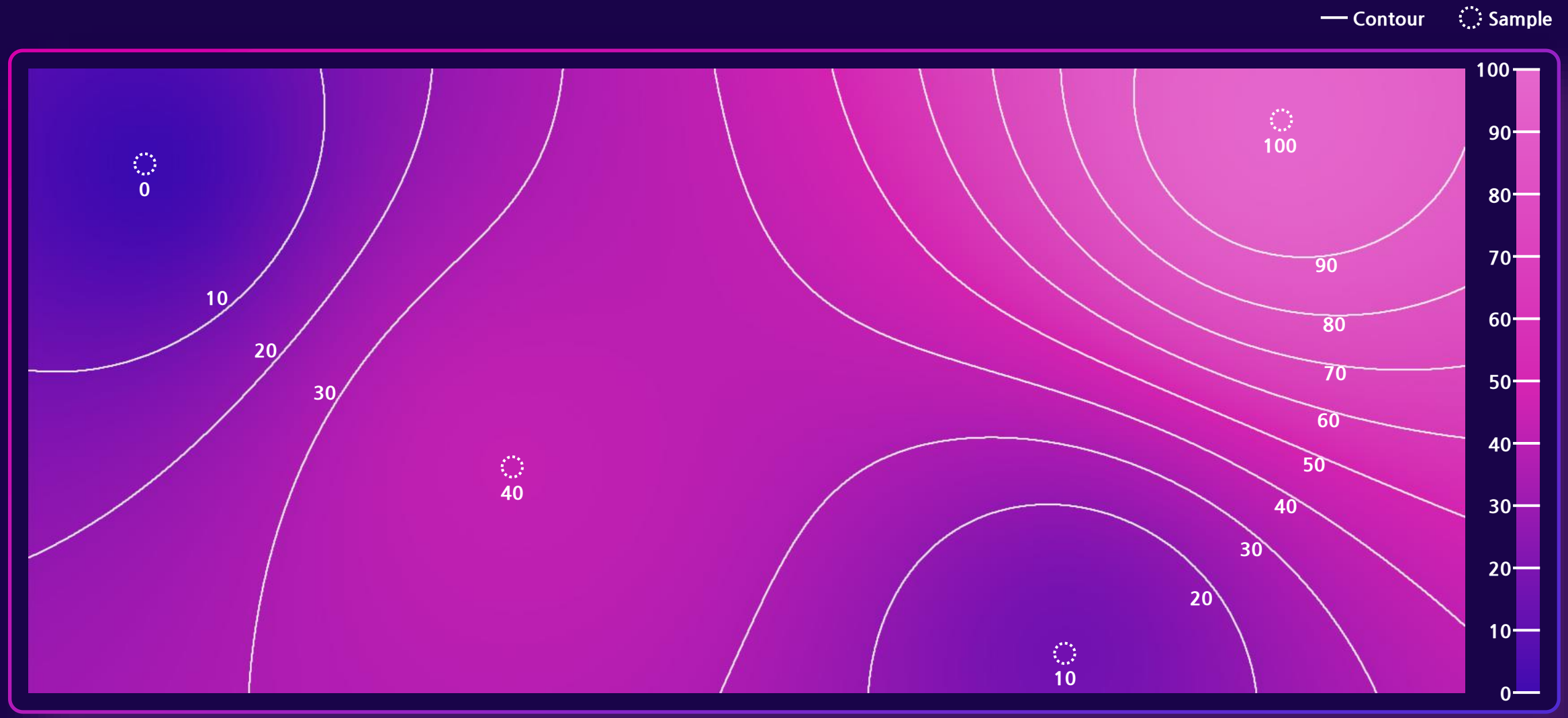
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$\times \frac{L}{4}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

등고선 생성 예시


Contour
Sample
0
10
20
30
40
10
100
90
80
70
60
50
40
30
20
100
90
80
70
60
50
40
30
20
10
0

WriteableBitmap에 그리기

C#

```
ImageSource CreateBitmap(float[,] values, float maximum, float minimum, GradientStopCollection gradientStops, int levels)
{
    int width = values.GetLength(1);
    int height = values.GetLength(0);
    var colors = gradientStops.OrderBy(item => item.Offset).Select(item => item.Color).ToArray();
    var offsets = gradientStops.OrderBy(item => item.Offset).Select(item => (float)item.Offset).ToArray();

    var poster = new float[height, width];
    for (int y = 0; y < height; y++)
        for (int x = 0; x < width; x++)
            poster[y, x] = Posterize(ToOffset(values[y, x], maximum, minimum), levels);

    var bitmap = new WriteableBitmap(width, height, 96, 96, PixelFormats.Bgra32, null);
    bitmap.Lock();
    unsafe
    {
        int* pixel = (int*)bitmap.BackBuffer.ToPointer();
        for (int y = 0; y < height; y++)
            for (int x = 0; x < width; x++)
            {
                var color = ToColor(ToOffset(values[y, x], maximum, minimum), colors, offsets);
                var lineOpacity = GetLineOpacity(poster, x, y, levels);
                var result = Colors.White * lineOpacity + color * (1 - lineOpacity);

                *pixel++ = result.A << 24 | result.R << 16 | result.G << 8 | result.B;
            }
    }
    bitmap.AddDirtyRect(new Int32Rect(0, 0, width, height));
    bitmap.Unlock();
    return bitmap;
}
```

공간 보간 결과

Posterization

보간 값의 색상 변환

등고선 Opacity 계산

$$C = A \times \alpha + B \times (1 - \alpha)$$

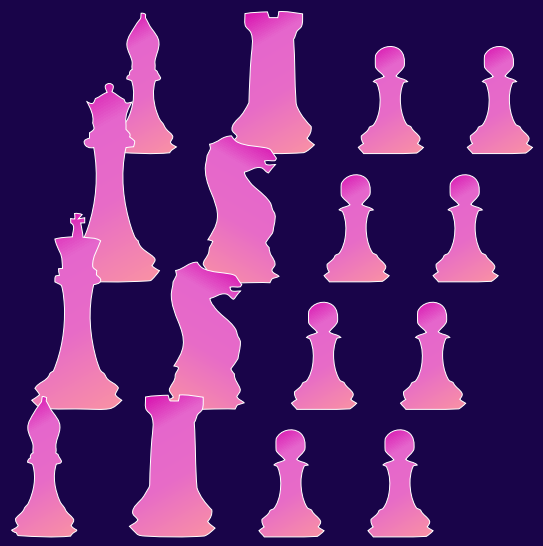
PixelFormats.Bgra32

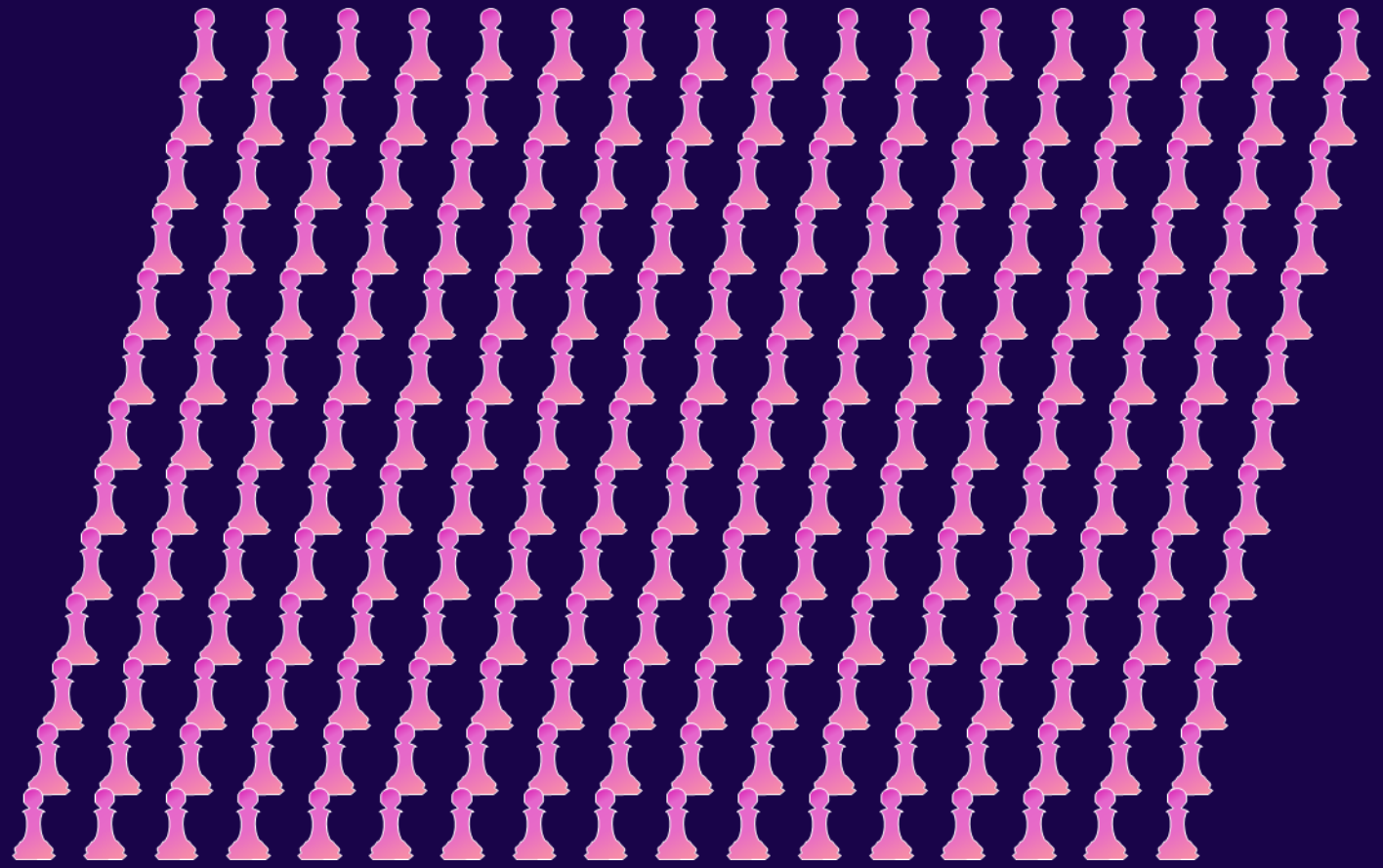
샘플 프로그램 구현 결과 영상

성능 개선 방안

GPU로 돌려보자!

CPU vs. GPU





GPU 기반 연산 개요

준비
- GPU Device 및 연산을 위한 Kernel(Shader) 준비

할당
- GPU Memory에 입력 및 결과 데이터를 위한 영역 할당

Initializing

입력
- 연산에 필요한 입력 데이터를 GPU Memory에 복사

실행
- 병렬 연산 수행 결과를 GPU Memory에 저장

추출
- GPU Memory의 결과 데이터를 주 Memory에 복사

Dispatching

03

GPU를 이용한 성능 개선

ILGPU 검토

ILGPU

- C# 기반 GPU 연산을 위한 일종의 JIT(just-in-time) 컴파일러
  - 멀티 플랫폼 지원 및 CUDA, C++ AMP, OpenCL 등 지원
- 최신 버전(1.5.1) 기준 프레임워크 최소 버전
  - .NET 5.0
  - .NET Standard 2.1
  - .NET Framework 4.7.1
- .NET Framework 4.6.1에서 사용 가능한 버전
  - ILGPU: 0.3.0 (2018년 4월 1일 게시)
  - ILGPU.Algorithms: 없음
- ILGPU.Algorithms 패키지가 없으면?
  - Math 클래스 관련 메서드 중 일부 사용 불가

ComputeSharp 검토

- Source generator를 이용한 DirectX 기반 라이브러리
  - C# 기반 GPU 연산 코드를 HLSL 코드로 변환하여 실행
- 최신 버전(3.1.0) 기준 .NET 8.0 이상 사용 가능
- .NET Framework 4.6.1에서 사용 가능한 버전
  - 2.1.0 (2023년 9월 27일 게시)
  - .NET Standard 2.0 기반으로 적용 가능
  - 단, 과거 버전 기준 튜토리얼 검색 필요
- 안정성 및 호환성에 대한 우려
  - 프로그램을 종료할 때 수 초의 지연시간 발생
  - .NET 8.0에서 사용하면 발생 안함
  - System 및 Microsoft로 시작하는 어셈블리 다수 포함

SharpDX 검토

- .NET에서 DirectX API를 사용할 수 있음
  - C# 코드만으로 GPGPU 불가능
  - HLSL 코드 기반 Shader 작성 필요
- .NET Framework 4.0 및 4.5에서 최신 버전 사용 가능
  - 마지막 업데이트: 2018년 8월 24일
  - Compute Shader 관련 예제 제공
- 고객사 프로젝트에 사용하기로 최종 선택
  - Vs. ComputeSharp: 개발 편의성보다 실행 안정성을 택함
  - HLSL 코드 작성 경험 있음

IDW 공간 보간 Shader

HLSL

```
cbuffer params : register(b0)
{
    float searchRadius;
    float weightPower;
}
StructuredBuffer<float3> samples : register(t0);
RWTexture2D<float> values : register(u0);

[numthreads(32, 32, 1)]
void CS(uint2 id : SV_DispatchThreadID)
{
    uint sampleCount, _;
    samples.GetDimensions(sampleCount, _);

    float sum = 0;
    float weights = 0;
    for (uint i = 0; i < sampleCount; i++)
    {
        float dist = distance(id, samples[i].xy);
        if (dist <= searchRadius)
        {
            float weight = dist == 0 ? 1 : 1 / pow(dist, weightPower);
            sum += weight * samples[i].z;
            weights += weight;
        }
    }
    values[id] = sum / weights;
}
```

CPU와 GPU간 리소스 바인딩을 위한 정의

병렬 처리를 위한 스레드 설정(cs_5_0 기준 최대 1024개)

$$d(X, X_i)$$

$$w_i(X) = \frac{1}{d(X, X_i)^p}$$

$$u(X) = \frac{\sum_{i=1}^{N} w_i(X) u_i}{\sum_{i=1}^{N} w_i(X)}$$

IDW 공간 보간 Shader 실행

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    int width = values.GetLength(1);
    int height = values.GetLength(0);
    var sampleArray = samples.ToArray();
    var parameters = new IdwInterpolationParameters(searchRadius, weightPower);

    using (var bytecode = ShaderBytecode.Compile(hlsl, "CS", "cs_5_0"))
    using (var device = new Device(DriverType.Hardware, DeviceCreationFlags.None))
    using (var shader = new ComputeShader(device, bytecode))

    using (var parametersBuffer = new Buffer(device, ...))                                  //b0, CB, 읽기 전용
    using (var samplesView = new ShaderResourceView(device, new Buffer(device, ...), ...))  //t0, SRV, 읽기 전용
    using (var valuesView = new UnorderedAccessView(device, new Texture2D(device, ...)))    //u0, UAV, 읽기 및 쓰기 가능
    using (var stagingTexture = new Texture2D(device, ...))    // GPU에서 main memory로 복사하기 위한 Staging texture
    {
        var context = device.ImmediateContext;

        context.ComputeShader.Set(shader);
        context.ComputeShader.SetConstantBuffer(0, parametersBuffer);   //b0, CB, 읽기 전용
        context.ComputeShader.SetShaderResource(0, samplesView);        //t0, SRV, 읽기 전용
        context.ComputeShader.SetUnorderedAccessView(0, valuesView);    //u0, UAV, 읽기 및 쓰기 가능
        context.UpdateSubresource(sampleArray, samplesView.Resource);
        context.UpdateSubresource(ref parameters, parametersBuffer);
        ...
}
```

준비

할당

입력

IDW 공간 보간 Shader 실행

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
        ...
        context.Dispatch(Math.Max(1, (width + 31) / 32), Math.Max(1, (height + 31) / 32), 1);

        context.CopyResource(valuesView.Resource, stagingTexture);
        var dataBox = context.MapSubresource(stagingTexture, 0, MapMode.Read, MapFlags.None);
        var sourcePtr = dataBox.DataPointer;

        unsafe
        {
            fixed (float* startPtr = values)
            {
                var valuesPtr = new IntPtr(startPtr);
                var rowPitch = width * sizeof(float);
                for (int i = 0; i < height; i++)
                {
                    Utilities.CopyMemory(valuesPtr, sourcePtr, rowPitch);
                    sourcePtr = IntPtr.Add(sourcePtr, dataBox.RowPitch);
                    valuesPtr = IntPtr.Add(valuesPtr, rowPitch);
                }
            }
        }
        context.UnmapSubresource(stagingTexture, 0);
    }
}
```

실행

추출

Constant buffer 생성 예시

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    ...

    var parameters = new IdwInterpolationParameters(searchRadius, weightPower);

    ...

    using (var parametersBuffer = new Buffer(device,
        new BufferDescription(
            //Constant buffer의 길이는 16의 배수로 설정해야 함.
            (int)Math.Ceiling(Marshal.SizeOf<IdwInterpolationParameters>() / 16d) * 16,
            BindFlags.ConstantBuffer,
            ResourceUsage.Default)))
    {
        var context = device.ImmediateContext;
        ...
        context.ComputeShader.SetConstantBuffer(0, parametersBuffer);
        ...
        context.UpdateSubresource(ref parameters, parametersBuffer);
        ...
    }

    ...
}
```

사용자가 처리하지 않은 예외

SharpDX.SharpDXException: 'HRESULT: [0x80070057], Module: [General], ApiCode: [E_INVALIDARG/Invalid Arguments], Message: 매개 변수가 틀립니다.
'

Copilot에 물어보기 | 호출 스택 표시 | 자세히 보기 | 세부 정보 복사

예외 설정

- [ ] 이 예외 형식이 throw되면 중단
- [x] 이 예외 형식을 사용자가 처리하지 않은 경우 중단

다음에서 발생한 경우 제외:

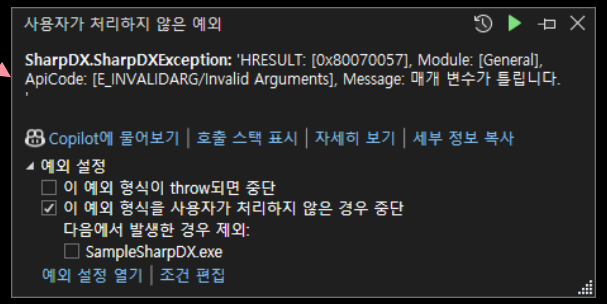
- [ ] SampleSharpDX.exe

예외 설정 열기 | 조건 편집

ShaderResourceView 생성 예시

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    ...

    using (var samplesView = new ShaderResourceView(device,
        new Buffer(device,
            new BufferDescription(sampleArray.Length * Marshal.SizeOf<SpatialSample>(), BindFlags.ShaderResource, ResourceUsage.Default)
            {
                CpuAccessFlags = CpuAccessFlags.None,
                OptionFlags = ResourceOptionFlags.BufferStructured,
                StructureByteStride = sampleSize
            }
        ),
        new ShaderResourceViewDescription
        {
            Format = SharpDX.DXGI.Format.Unknown,
            Dimension = ShaderResourceViewDimension.Buffer,
            Buffer = new ShaderResourceViewDescription.BufferResource() { ElementWidth = sampleArray.Length }
        }))
    {
        ...
    }
    ...
}
```



UnorderedAccessView 생성 예시

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    ...
    using (var valuesView = new UnorderedAccessView(device,
        new Texture2D(device,
            new Texture2DDescription
            {
                Width = width,
                Height = height,
                ArraySize = 1,
                MipLevels = 0,
                Format = SharpDX.DXGI.Format.R32_Float,
                BindFlags = BindFlags.UnorderedAccess,
                Usage = ResourceUsage.Default,
                CpuAccessFlags = CpuAccessFlags.None,
                OptionFlags = ResourceOptionFlags.None,
                SampleDescription = new SharpDX.DXGI.SampleDescription(1, 0)
            })))
    {
        var context = device.ImmediateContext;
        ...
        context.ComputeShader.SetUnorderedAccessView(0, valuesView);    //u0, 읽기 및 쓰기 가능
        ...
    }
    ...
}
```

사용자가 처리하지 않은 예외

SharpDX.SharpDXException: 'HRESULT: [0x80070057], Module: [General], ApiCode: [E_INVALIDARG/Invalid Arguments], Message: 매개 변수가 틀립니다.
'

Copilot에 물어보기 | 호출 스택 표시 | 자세히 보기 | 세부 정보 복사

예외 설정

- [ ] 이 예외 형식이 throw되면 중단
- [x] 이 예외 형식을 사용자가 처리하지 않은 경우 중단
  다음에서 발생한 경우 제외:
  - [ ] SampleSharpDX.exe

예외 설정 열기 | 조건 편집

Staging texture 생성 예시

C#

```
void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    ...
    using (var stagingTexture = new Texture2D(device,
        new Texture2DDescription
        {
            Width = width,
            Height = height,
            ArraySize = 1,
            MipLevels = 0,
            Format = SharpDX.DXGI.Format.R32_Float,
            BindFlags = BindFlags.None,
            Usage = ResourceUsage.Staging,
            CpuAccessFlags = CpuAccessFlags.Read,
            OptionFlags = ResourceOptionFlags.None,
            SampleDescription = new SharpDX.DXGI.SampleDescription(1, 0)
        }))
    {
        var context = device.ImmediateContext;
        ...
        context.Dispatch(Math.Max(1, (width + 31) / 32), Math.Max(1, (height + 31) / 32), 1);
        context.CopyResource(valuesView.Resource, stagingTexture);
        ...
    }
    ...
}
```

사용자가 처리하지 않은 예외

SharpDX.SharpDXException: 'HRESULT: [0x80070057], Module: [General], ApiCode: [E_INVALIDARG/Invalid Arguments], Message: 매개 변수가 틀립니다.
'

Copilot에 물어보기 | 호출 스택 표시 | 자세히 보기 | 세부 정보 복사

예외 설정

- [ ] 이 예외 형식이 throw되면 중단
- [x] 이 예외 형식을 사용자가 처리하지 않은 경우 중단
  다음에서 발생한 경우 제외:
  - [ ] SampleSharpDX.exe

예외 설정 열기 | 조건 편집

C# vs. HLSL: 색상 변환

C#

```
Color ToColor(float offset, Color[] colors, float[] offsets)
{
    var ranges = colors.Length - 1;

    int index;
    for (index = 0; index < ranges; index++)
        if (offset < offsets[index])
            break;

    if (index == 0 || offsets[index] <= offset)
        return colors[index];

    var range = offsets[index] - offsets[index - 1];
    var alpha = (offset - offsets[index - 1]) / range;
    var colorA = colors[index];
    var colorB = colors[index - 1];

    return colorA * alpha + colorB * (1 - alpha);
}
```

색상 구간 탐색

구간 외 처리

α, A, B

$$C = A \times \alpha + B \times (1 - \alpha)$$

HLSL

```
float4 toColor(float offset)
{
    uint ranges;
    colors.GetDimensions(ranges);
    ranges--;

    int index;
    for (index = 0; index < ranges; index++)
        if (offset < offsets[index])
            break;

    if (index == 0 || offsets[index] <= offset)
        return colors[index];

    float range = offsets[index] - offsets[index - 1];
    float alpha = (offset - offsets[index - 1]) / range;
    float4 colorA = colors[index];
    float4 colorB = colors[index - 1];

    return lerp(colorB, colorA, alpha);
}
```

C# vs. HLSL: 등고선 검출

C#

```
float GetLineOpacity(float[,] poster, int currX, int currY, int levels)
{
    var prevX = currX + (currX <= 0 ? 0 : -1);
    var prevY = currY + (currY <= 0 ? 0 : -1);
    var nextX = currX + (currX >= poster.GetLength(1) - 1 ? 0 : 1);
    var nextY = currY + (currY >= poster.GetLength(0) - 1 ? 0 : 1);

    var gX = poster[prevY, prevX] - poster[prevY, nextX]
        + (poster[currY, prevX] - poster[currY, nextX]) * 2
        + poster[nextY, prevX] - poster[nextY, nextX];

    var gY = poster[prevY, prevX] - poster[nextY, prevX]
        + (poster[prevY, currX] - poster[nextY, currX]) * 2
        + poster[prevY, nextX] - poster[nextY, nextX];

    gX *= levels / 4f;
    gY *= levels / 4f;

    return MathF.Sqrt(gX * gX + gY * gY);
}
```

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$\times \frac{L}{4}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

HLSL

```
float getLineOpacity(int currX, int currY)
{
    uint width;
    uint height;
    values.GetDimensions(width, height);

    int prevX = currX + (currX <= 0 ? 0 : -1);
    int prevY = currY + (currY <= 0 ? 0 : -1);
    int nextX = currX + (currX >= width - 1 ? 0 : 1);
    int nextY = currY + (currY >= height - 1 ? 0 : 1);

    float gX = posterize(prevX, prevY) - posterize(nextX, prevY)
        + (posterize(prevX, currY) - posterize(nextX, currY)) * 2
        + posterize(prevX, nextY) - posterize(nextX, nextY);

    float gY = posterize(prevX, prevY) - posterize(prevX, nextY)
        + (posterize(currX, prevY) - posterize(currX, nextY)) * 2
        + posterize(nextX, prevY) - posterize(nextX, nextY);

    gX *= levels / 4.0;
    gY *= levels / 4.0;

    return sqrt(gX * gX + gY * gY);
}
```

GPU를 이용한 성능 개선 결과 영상

04

C# 언어 기반 GPU 연산

IDW 공간 보간 Shader

C#

```
using ComputeSharp;

...

[ThreadGroupSize(DefaultThreadGroupSizes.XY)]
[GeneratedComputeShaderDescriptor]
readonly partial struct GpuIdwInterpolationShader(
    ReadOnlyBuffer<float3> samples, ReadWriteTexture2D<float> values, float searchRadius, float weightPower) : IComputeShader
{
    public void Execute()
    {
        var sum = 0f;
        var weights = 0f;
        for (int i = 0; i < samples.Length; i++)
        {
            var distance = Hlsl.Distance(ThreadIds.XY, samples[i].XY);
            if (distance <= searchRadius)
            {
                var weight = distance == 0 ? 1 : (1 / Hlsl.Pow(distance, weightPower));
                sum += weight * samples[i].Z;
                weights += weight;
            }
        }
        values[ThreadIds.XY] = sum / weights;
    }
}
```

매개변수를 통한 리소스 바인딩

$$d(X, X_i)$$

$$w_i(X) = \frac{1}{d(X, X_i)^p}$$

$$u(X) = \frac{\sum_{i=1}^{N} w_i(X) u_i}{\sum_{i=1}^{N} w_i(X)}$$

IDW 공간 보간 Shader 실행

C#

```
using ComputeSharp;
...

void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    int width = values.GetLength(1);
    int height = values.GetLength(0);

    var device = GraphicsDevice.GetDefault();

    using (var samplesBuffer = device.AllocateReadOnlyBuffer<float3>(sampleArray.Length))
    using (var valuesBuffer = device.AllocateReadWriteTexture2D<float>(width, height))
    {
        samplesBuffer.CopyFrom(samples.Select(sample => new float3(sample.X, sample.Y, sample.Value)).ToArray());

        device.For(width, height, new GpuIdwInterpolationShader(samplesBuffer, valuesBuffer, searchRadius, weightPower));

        valuesBuffer.CopyTo(values);
    }
}
```

준비

할당

입력

입력 및 실행(Shader 준비 포함)

추출

Heat map 생성 Shader

C#

```
[ThreadGroupSize(DefaultThreadGroupSizes.XY)]
[GeneratedComputeShaderDescriptor]
readonly partial struct SpatialHeatMapShader(
    ReadWriteTexture2D<int> heatMap,
    ReadOnlyTexture1D<float4> colors,
    ReadOnlyTexture1D<float> offsets,
    ReadOnlyTexture2D<float> values,
    float4 contourColor,
    uint levels,
    float maximum,
    float minimum) : IComputeShader
{
    private static float ToSRgb(float value)
    {
        if (!(value > 0.0f))
            return 0;
        else if (value <= 0.0031308f)
            return value * 12.92f;
        else if (value < 1.0)
            return 1.055f * Hlsl.Pow(value, 1.0f / 2.4f) - 0.055f;
        else
            return 1.0f;
    }

    private static float4 ToSRgba(float4 color)
        => new(ToSRgb(color.R), ToSRgb(color.G), ToSRgb(color.B), color.A);

    ...
```

scRGB to sRGB 변환

C#

```
...

private float ToOffset(int2 id)
    => Hlsl.IsNaN(values[id])
    ? 0.0f
    : Hlsl.Clamp((values[id] - minimum) / (maximum - minimum), 0, 1);

private float4 ToColor(float offset)
{
    var ranges = colors.Width - 1;

    int index;
    for (index = 0; index < ranges; index++)
        if (offset < offsets[index])
            break;

    if (index == 0 || offsets[index] <= offset)
        return colors[index];

    var range = offsets[index] - offsets[index - 1];
    var alpha = (offset - offsets[index - 1]) / range;
    var colorA = colors[index];
    var colorB = colors[index - 1];

    return Hlsl.Lerp(colorB, colorA, alpha);
}

...
```

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

색상 구간 탐색

구간 외 처리

α, A, B

$$C = A \times \alpha + B \times (1 - \alpha)$$

Heat map 생성 Shader

C#

```
...

private float Posterize(int x, int y)
    => Hlsl.Floor(ToOffset(new int2(x, y)) * levels) / levels;

private float GetLineOpacity(int currX, int currY)
{
    if (levels < 2) return 0;

    var prevX = currX + (currX <= 0 ? 0 : -1);
    var prevY = currY + (currY <= 0 ? 0 : -1);
    var nextX = currX + (currX >= values.Width - 1 ? 0 : 1);
    var nextY = currY + (currY >= values.Height - 1 ? 0 : 1);

    var gX = Posterize(prevX, prevY) - Posterize(nextX, prevY)
        + (Posterize(prevX, currY) - Posterize(nextX, currY)) * 2
        + Posterize(prevX, nextY) - Posterize(nextX, nextY);

    var gY = Posterize(prevX, prevY) - Posterize(prevX, nextY)
        + (Posterize(currX, prevY) - Posterize(currX, nextY)) * 2
        + Posterize(nextX, prevY) - Posterize(nextX, nextY);

    gX *= levels / 4f;
    gY *= levels / 4f;

    return Hlsl.Sqrt(gX * gX + gY * gY);
}

...
```

$$X' = \frac{\lfloor X \times L \rfloor}{L}$$

Sobel 필터

$$\times \frac{L}{4}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

C#

```
...

public void Execute()
{
    var color = ToColor(ToOffset(ThreadIds.XY));
    var lineOpacity =
        GetLineOpacity(ThreadIds.X, ThreadIds.Y) * contourColor.A;
    var result =
        Hlsl.Lerp(color, new float4(contourColor.RGB, 1), lineOpacity);

    result = ToSRgba(result);
    heatMap[ThreadIds.XY] = ((int)(result.W * 255f) << 24)
        | ((int)(result.X * 255f) << 16)
        | ((int)(result.Y * 255f) << 8)
        | ((int)(result.Z * 255f));
}
}
```

보간 값의 색상 변환, 등고선 Opacity에 따른 최종 픽셀 값 계산

Heat map 생성 Shader 실행

C#

```
ImageSource CreateBitmap(float[,] values, float maximum, float minimum, GradientStopCollection gradientStops, int levels)
{
    var width = values.GetLength(1);
    var height = values.GetLength(0);
    var stops = gradientStops.OrderBy(item => item.Offset);
    var bitmap = new WriteableBitmap(width, height, 96, 96, PixelFormats.Bgra32, null);
    var device = GraphicsDevice.GetDefault();
    using (var heatMapBuffer = device.AllocateReadWriteTexture2D<int>(width, height))
    using (var valuesBuffer = device.AllocateReadOnlyTexture2D<float>(width, height))
    using (var colorsBuffer = device.AllocateReadOnlyTexture1D<float4>(gradientStops.Count))
    using (var offsetsBuffer = device.AllocateReadOnlyTexture1D<float>(gradientStops.Count))
    {
        valuesBuffer.CopyFrom(values);
        colorsBuffer.CopyFrom(stops.Select(item => new float4(item.Color.ScR, item.Color.ScG, item.Color.ScB, item.Color.ScA)).ToArray());
        offsetsBuffer.CopyFrom(stops.Select(item => (float)item.Offset).ToArray());
        device.For(width, height,
            new SpatialHeatMapShader(heatMapBuffer, colorsBuffer, offsetsBuffer, valuesBuffer, new float4(1, 1, 1, 1), (uint)levels, maximum, minimum));

        bitmap.Lock();
        unsafe
        {
            heatMapBuffer.CopyTo(new Span<int>((void*)bitmap.BackBuffer, width * height));
        }
        bitmap.AddDirtyRect(new Int32Rect(0, 0, width, height));
        bitmap.Unlock();
    }
    return bitmap;
}
```

IDW 공간 보간 Kernel

ILGPU

C#

```
using ILGPU;
using ILGPU.Runtime;
using System.Numerics;

...

struct GpuIdwInterpolationKernel(
    ArrayView1D<SpatialSample, Stride1D.Dense> samples, ArrayView2D<float, Stride2D.DenseX> values, float searchRadius, float weightPower)
{
    public void Execute(Index2D id)
    {
        var sum = 0f;
        var weights = 0f;
        for (int i = 0; i < samples.Length; i++)
        {
            var distance = Vector2.Distance(new Vector2(id.X, id.Y), new Vector2(samples[i].Y, samples[i].X));
            if (distance <= searchRadius)
            {
                var weight = distance == 0 ? 1 : (1 / float.Pow(distance, weightPower));
                sum += weight * samples[i].Value;
                weights += weight;
            }
        }
        values[id] = sum / weights;
    }
}
```

매개변수를 통한 리소스 바인딩

$$d(X, X_i)$$

$$w_i(X) = \frac{1}{d(X, X_i)^p}$$

$$u(X) = \frac{\sum_{i=1}^{N} w_i(X) u_i}{\sum_{i=1}^{N} w_i(X)}$$

IDW 공간 보간 Kernel 실행

ILGPU

C#

```
using ILGPU;
using ILGPU.Runtime;
...

void Interpolate(IEnumerable<SpatialSample> samples, float[,] values, float searchRadius = float.PositiveInfinity, float weightPower = 2)
{
    int width = values.GetLength(1);
    int height = values.GetLength(0);

    using (var context = Context.Create(b => b.AllAccelerators().EnableAlgorithms()))
    {
        var device = context.Devices.FirstOrDefault(device => device.AcceleratorType != AcceleratorType.CPU) ?? context.Devices.FirstOrDefault();
        using (var accelerator = device.CreateAccelerator(context))
        {
            var kernel = accelerator.LoadAutoGroupedStreamKernel<Index2D, GpuIdwInterpolationKernel>((id, kernel) => kernel.Execute(id));

            using (var samplesBuffer = accelerator.Allocate1D<SpatialSample>(sampleArray.Length))
            using (var valuesBuffer = accelerator.Allocate2DDenseX<float>(new Index2D(height, width)))
            {
                samplesBuffer.CopyFromCPU(samples.ToArray());

                kernel(new Index2D(height, width), new GpuIdwInterpolationKernel(samplesBuffer.View, valuesBuffer.View, searchRadius, weightPower));

                valuesBuffer.CopyToCPU(values);
            }
        }
    }
}
```

준비

할당

입력

실행

추출

Heat map 생성 Kernel

ILGPU

C#

```
struct SpatialHeatMapKernel(
    ArrayView2D<int, Stride2D.DenseX> heatMap,
    ArrayView1D<Vector4, Stride1D.Dense> colors,
    ArrayView1D<float, Stride1D.Dense> offsets,
    ArrayView2D<float, Stride2D.DenseX> values,
    Vector4 contourColor,
    uint levels,
    float maximum,
    float minimum)
{
    private static float ToSRgb(float value)
    {
        if (!(value > 0.0f))
            return 0;
        else if (value <= 0.0031308f)
            return value * 12.92f;
        else if (value < 1.0)
            return 1.055f * float.Pow(value, 1.0f / 2.4f) - 0.055f;
        else
            return 1.0f;
    }

    private static Vector4 ToSRgba(Vector4 color)
        => new(ToSRgb(color.X), ToSRgb(color.Y), ToSRgb(color.Z), color.W);

    ...
```

scRGB to sRGB 변환

C#

```
    ...

    private readonly float ToOffset(Index2D id)
        => float.IsNaN(values[id])
        ? 0.0f
        : float.Max(0f,
            float.Min(1f, (values[id] - minimum) / (maximum - minimum)));

    private readonly Vector4 ToColor(float offset)
    {
        var ranges = colors.Length - 1;

        int index;
        for (index = 0; index < ranges; index++)
            if (offset < offsets[index])
                break;

        if (index == 0 || offsets[index] <= offset)
            return colors[index];

        var range = offsets[index] - offsets[index - 1];
        var alpha = (offset - offsets[index - 1]) / range;
        var colorA = colors[index];
        var colorB = colors[index - 1];

        return Vector4.Lerp(colorB, colorA, alpha);
    }

    ...
```

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

색상 구간 탐색

구간 외 처리

α, A, B

$$C = A \times \alpha + B \times (1 - \alpha)$$

Heat map 생성 Kernel

ILGPU

C#

```
...

private readonly float Posterize(int x, int y)
    => float.Floor(ToOffset(new Index2D(x, y)) * levels) / levels;

private readonly float GetLineOpacity(int currX, int currY)
{
    if (levels < 2) return 0;

    var prevX = currX + (currX <= 0 ? 0 : -1);
    var prevY = currY + (currY <= 0 ? 0 : -1);
    var nextX = currX + (currX >= values.IntExtent.X - 1 ? 0 : 1);
    var nextY = currY + (currY >= values.IntExtent.Y - 1 ? 0 : 1);

    var gX = Posterize(prevX, prevY) - Posterize(nextX, prevY)
        + (Posterize(prevX, currY) - Posterize(nextX, currY)) * 2
        + Posterize(prevX, nextY) - Posterize(nextX, nextY);

    var gY = Posterize(prevX, prevY) - Posterize(prevX, nextY)
        + (Posterize(currX, prevY) - Posterize(currX, nextY)) * 2
        + Posterize(nextX, prevY) - Posterize(nextX, nextY);

    gX *= levels / 4f;
    gY *= levels / 4f;

    return float.Sqrt(gX * gX + gY * gY);
}

...
```

$$X' = \frac{\lfloor X \times L \rfloor}{L}$$

Sobel 필터

$$\times \frac{L}{4}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

C#

```
...

public void Execute(Index2D id)
{
    var color = ToColor(ToOffset(id));
    float lineOpacity = GetLineOpacity(id.X, id.Y) * contourColor.W;
    var result = Vector4.Lerp(color,
        new Vector4(
            contourColor.X,
            contourColor.Y,
            contourColor.Z, 1), lineOpacity);

    result = ToSRgba(result);
    heatMap[id] = ((int)(result.W * 255f) << 24)
        | ((int)(result.X * 255f) << 16)
        | ((int)(result.Y * 255f) << 8)
        | ((int)(result.Z * 255f));
}
}
```

보간 값의 색상 변환, 등고선 Opacity에 따른 최종 픽셀 값 계산

Heat map 생성 Kernel 실행

ILGPU

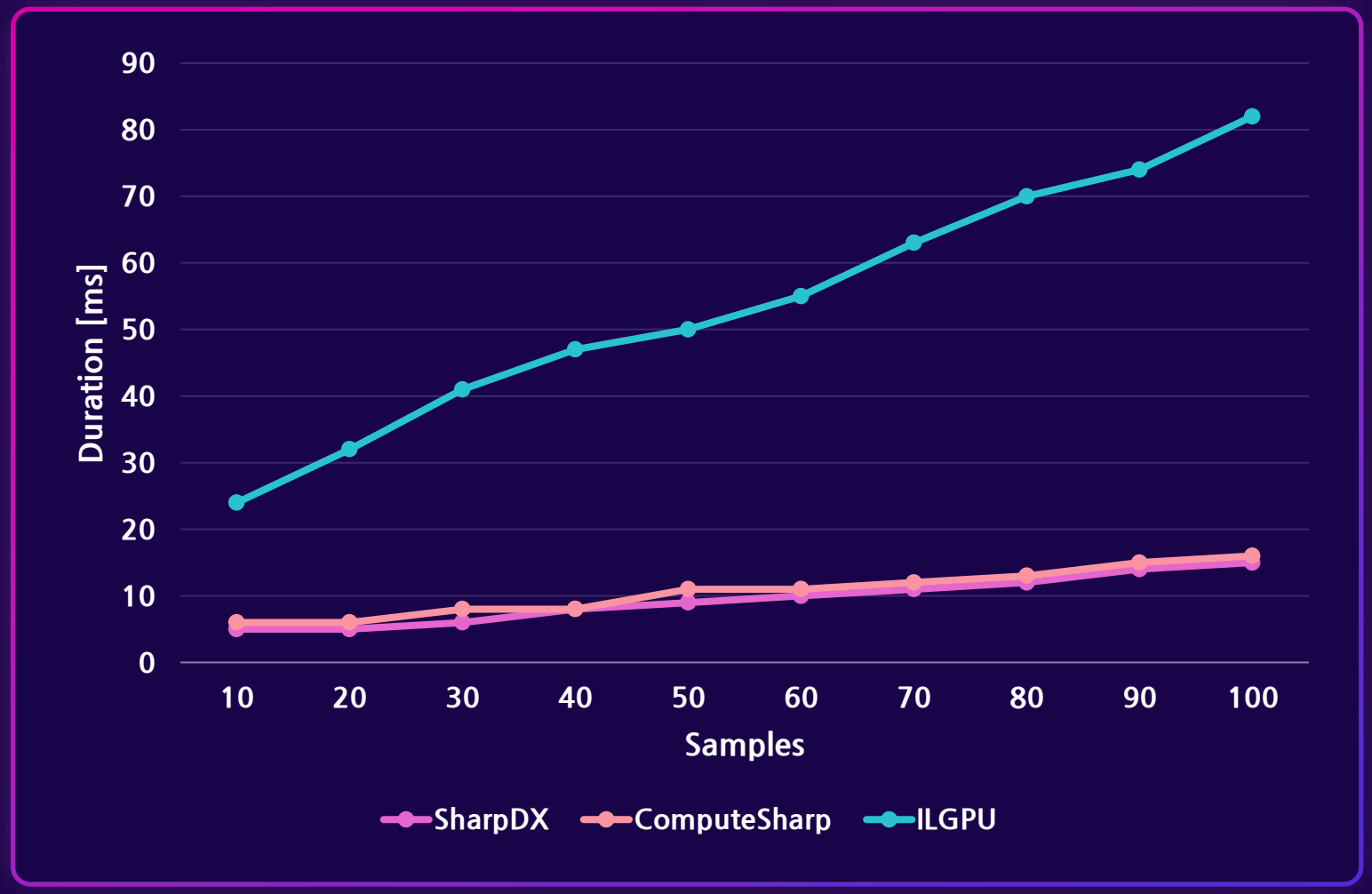
C#

```
ImageSource CreateBitmap(float[,] values, float maximum, float minimum, GradientStopCollection gradientStops, int levels)
{
    var width = values.GetLength(1);
    var height = values.GetLength(0);
    var stops = gradientStops.OrderBy(item => item.Offset);
    var bitmap = new WriteableBitmap(width, height, 96, 96, PixelFormats.Bgra32, null);
    using (var context = Context.Create(b => b.AllAccelerators().EnableAlgorithms()))
    {
        var device = context.Devices.FirstOrDefault(device => device.AcceleratorType != AcceleratorType.CPU) ?? context.Devices.FirstOrDefault();
        using (var accelerator = device.CreateAccelerator(context))
        using (var heatMapBuffer = accelerator.Allocate2DDenseX<int>(new Index2D(height, width)))
        using (var valuesBuffer = accelerator.Allocate2DDenseX<float>(new Index2D(height, width)))
        using (var colorsBuffer = accelerator.Allocate1D<Vector4>(gradientStops.Count))
        using (var offsetsBuffer = accelerator.Allocate1D<float>(gradientStops.Count))
        {
            var kernel = accelerator.LoadAutoGroupedStreamKernel<Index2D, SpatialHeatMapKernel>((id, kernel) => kernel.Execute(id));

            valuesBuffer.CopyFromCPU(values);
            colorsBuffer.CopyFromCPU(stops.Select(item => new Vector4(item.Color.ScR, item.Color.ScG, item.Color.ScB, item.Color.ScA)).ToArray());
            offsetsBuffer.CopyFromCPU(stops.Select(item => (float)item.Offset).ToArray());

            kernel(new Index2D(height, width), new SpatialHeatMapKernel(
                heatMapBuffer.View, colorsBuffer.View, offsetsBuffer.View, valuesBuffer.View, new Vector4(1, 1, 1, 1), (uint)levels, maximum, minimum));

            bitmap.WritePixels(new Int32Rect(0, 0, width, height), heatMapBuffer.GetAsArray2D(), width * bitmap.Format.BitsPerPixel / 8, 0);
        }
    }
    return bitmap;
}
```

05

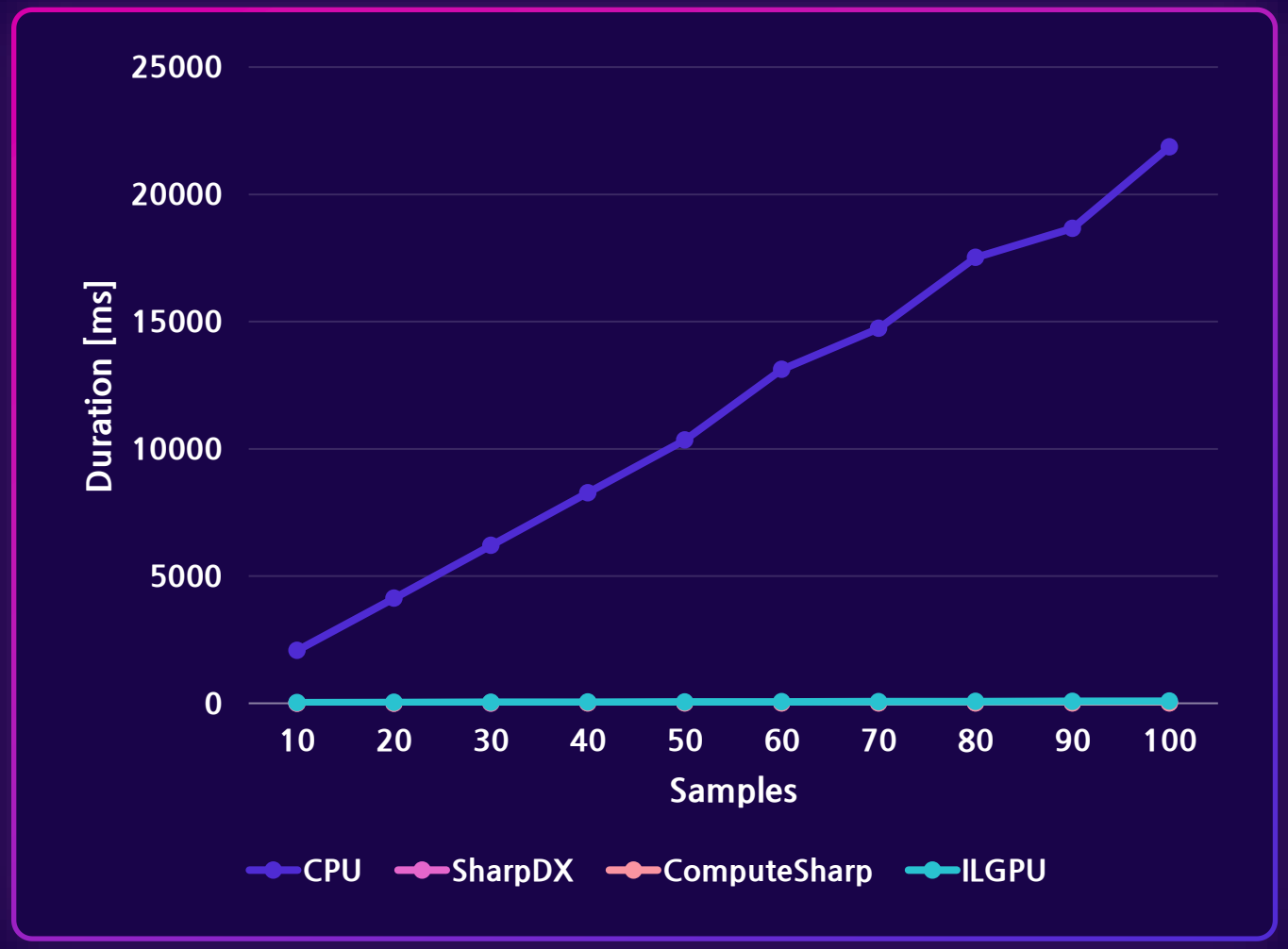
결과 분석 및 마무리

GPU 라이브러리별 연산 소요시간



Samples	SharpDX	Compute Sharp	ILGPU
10	5	6	24
20	5	6	32
30	6	8	41
40	8	8	47
50	9	11	50
60	10	11	55
70	11	12	63
80	12	13	70
90	14	15	74
100	15	16	82

CPU vs. GPU 성능 비교



Samples	CPU	SharpDX	Compute Sharp	ILGPU
10	2076	5	6	24
20	4126	5	6	32
30	6200	6	8	41
40	8274	8	8	47
50	10342	9	11	50
60	13115	10	11	55
70	14730	11	12	63
80	17518	12	13	70
90	18666	14	15	74
100	21867	15	16	82

GPU 라이브러리 비교

항목	SharpDX	ComputeSharp	ILGPU
플랫폼	Windows Only	Windows Only	크로스 플랫폼
.NET 버전	.NET Framework 4.5	.NET 8.0	.NET 5.0
기반 기술	DirectX	DirectX	CUDA, C++ AMP, OpenCL
시각적 렌더링	○	○	×
GPGPU 언어	HLSL	C#	C#
작업 비용	최상	하	하
성능(???)	1	1	1/5
주관적 평가	3rd	2nd	★ 1st

결론

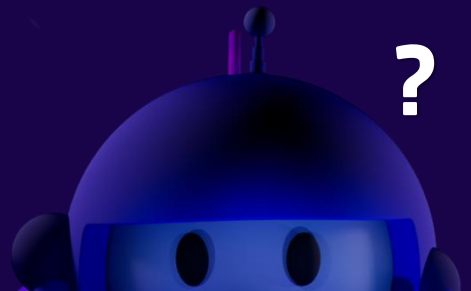
기, 승, 전, 업그레이드

참고자료

- Source code
  - https://github.com/Vagabond-K/SpatialInterpolationSample
- Inverse distance weighting
  - https://en.wikipedia.org/wiki/Inverse_distance_weighting
- Alpha compositing
  - https://en.wikipedia.org/wiki/Alpha_compositing
- Posterization
  - https://en.wikipedia.org/wiki/Posterization
- Sobel operator
  - https://en.wikipedia.org/wiki/Sobel_operator

Thank you

But...



?

발표자 소개

- 경남 창원의 어느 중소기업에서 19년 가까이 개발자로 근무 중
- 입사 초반에는 주로 ERP와 MES, 공장 자동화 관련 개발 수행
- 중반부터 에너지 관리 시스템 개발 및 에너지 기술 연구 과제 수행
- 연구기관 SW 개발 용역 수행 (한국전기연구원, 한국전자기술연구원 등)
- 개인 활동: Blog, GitHub, ~~YouTube~~
- 닉네임: Vagabond K

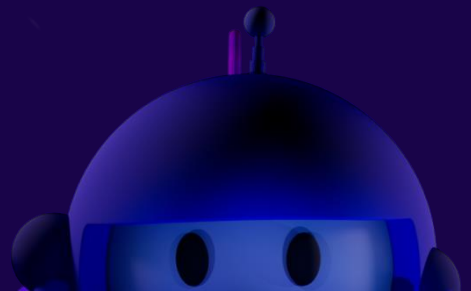
VAGABOND

www.vagabond-k.com

발표자 소속 회사 소개

- 1999년에 창립한 경남 창원의 어느 중소기업
- 사업분야: 스마트 팩토리, 에너지 관리 플랫폼
- 인원: 26명
- 매출액: 43억 (2023년 기준, 출처: 잡코리아)
- 전화: 055-274-0300
- 홈페이지: http://tisys.co.kr
- 슬로건:

"작지만, 강한 기업!"

(주)TIS
TIS Co., Ltd.

Thank you