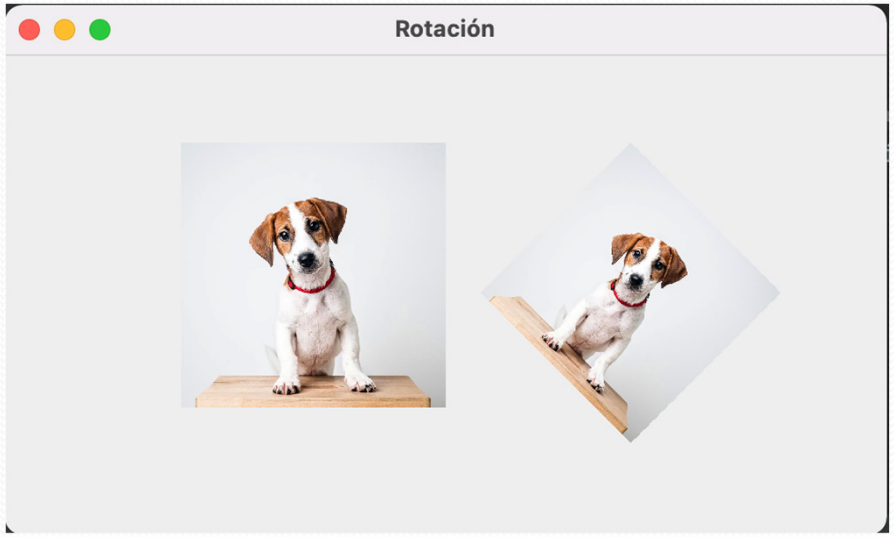
Imágenes

Imágenes

- Se puede desplegar una imagen mediante el método drawImage heredado de Graphics.
- Se puede rotar la imagen.



Imágenes

```
try {
    BufferedImage imagen = ImageIO.read(new File("perro.jpg"));
    BufferedImage imagenRot = rotateImageByDegrees(imagen, 45);
    g.drawImage(imagen, 100, 50, 150, 150, null);
    g.drawImage(imagenRot, 270, 50, 170, 170, null);
}
catch (IOException e) {
    System.out.println("Error: " + e);
}
```

Imágenes

- El método rotateImageByDegrees está en la página del curso: https://www.mat.uson.mx/~havillam/pa/Common/RotateImage.java

Imágenes

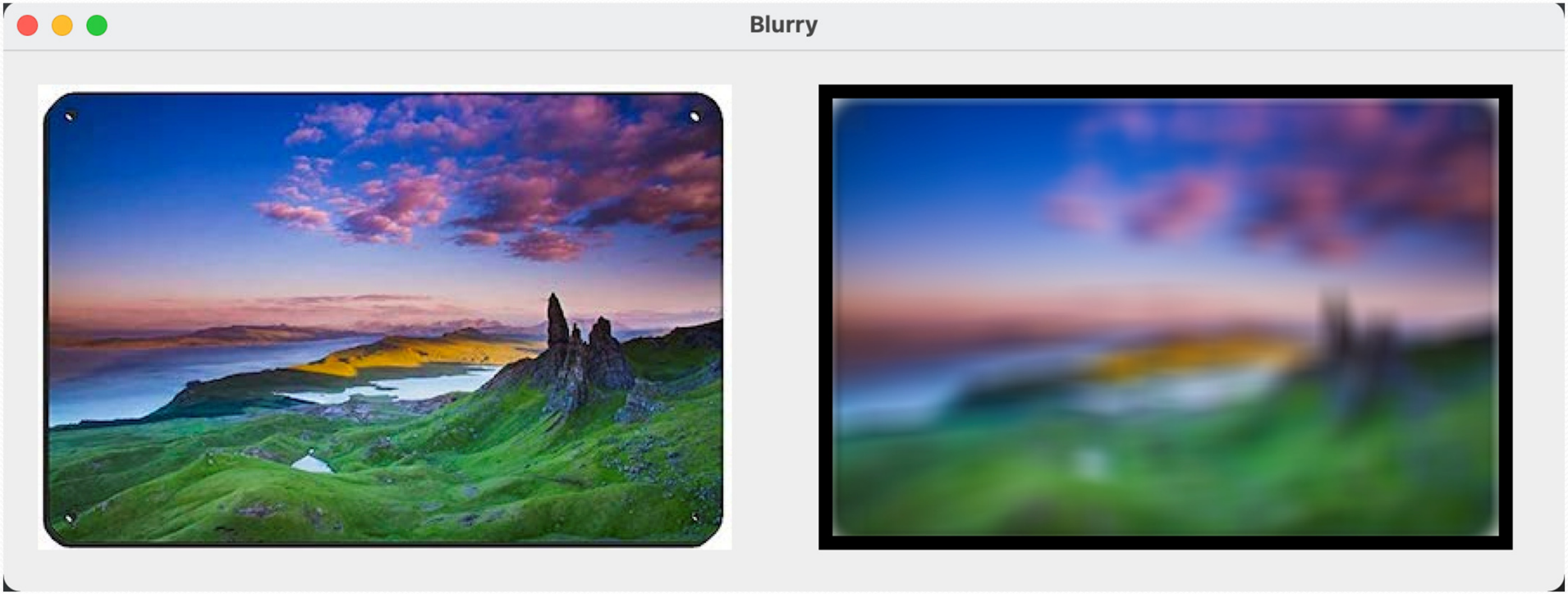
- Graphics2D define una variante de drawImage:

  drawImage(BufferedImage img, BufferedImageOp op, int x, int y)
- Despliega la imagen filtrada por una operación de tipo BufferedImageOp.
- BufferedImageOp es una interface que define operaciones sobre objetos de tipo BufferedImage.
- Clases conocidas que la implementan: AffineTransformOp, ColorConvertOp, ConvolveOp, LookupOp, RescaleOp.

ConvolveOp

- **Convolución**. Una operación que obtiene el valor del pixel de salida a partir de un pixel de entrada multiplicando un kernel con la vecindad del pixel de entrada.
- El kernel es una matriz de pesos.
- Ejemplo: un kernel para hacer una imagen borrosa es:
- $K = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

ConvolveOp



ConvolveOp

```
try {
    BufferedImage image = ImageIO.read(new File("landscape400.jpg"));
    int w = 17; int h = 17;
    float[] weights = new float[w * h];
    Arrays.fill(weights, 1.0f / (w * h));
    Kernel kernel = new Kernel(w, h, weights);
    RenderingHints hints = new RenderingHints(
                RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
```

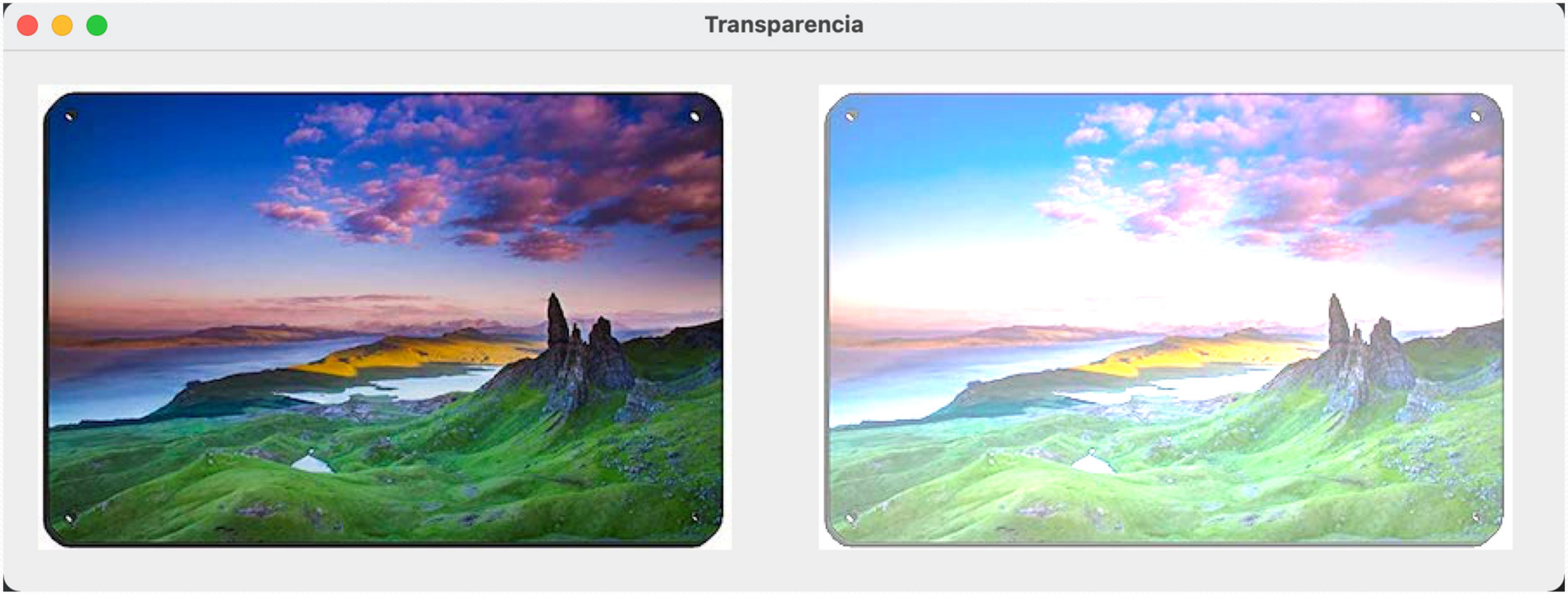
ConvolveOp

```
    ConvolveOp op = new ConvolveOp(kernel,
                                   ConvolveOp.EDGE_ZERO_FILL, hints);
    g2.drawImage(image, 20, 20, null);
    g2.drawImage(image, op, 470, 20);
}
catch (IOException e) {
    System.out.println("Error: " + e);
}
```

RescaleOp

- **Escalamiento**. Multiplica el valor de un pixel por un factor y le suma un offset.
- La operación se hace por canal: rojo, verde, azul y, si existe, alfa.
- Ejemplo: multiplicar el alfa de una imagen png por 0.5 para hacerla traslúcida.

RescaleOp



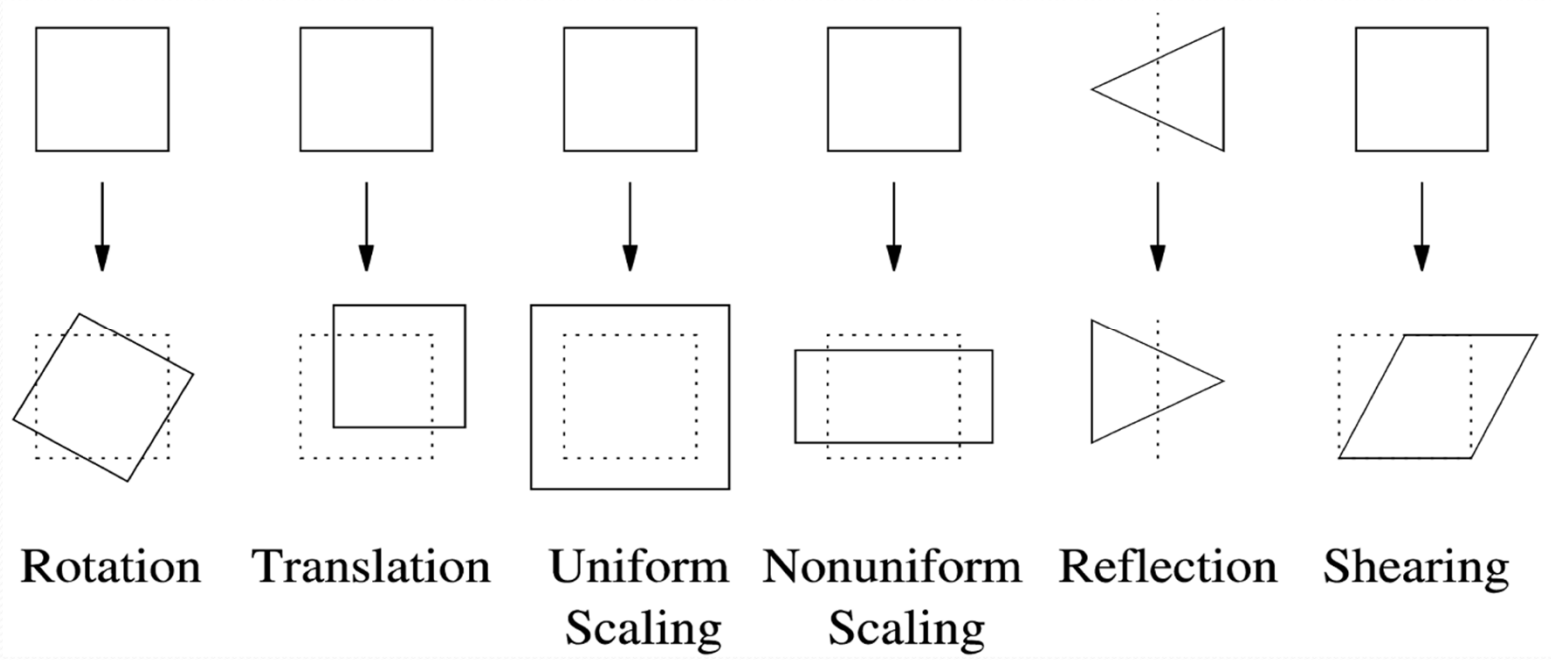
RescaleOp

```
try {
    BufferedImage image = ImageIO.read(new File("landscape400.png"));
    float[] scales = {1f, 1f, 1f, 0.5f};
    float[] offsets = new float[4];
    RescaleOp rescaleOp = new RescaleOp(scales, offsets, hints);
    g2.drawImage(image, 20, 20, null);
    g2.drawImage(image, rescaleOp, 470, 20);
}
catch (IOException e) { … }
```

AffineTransformOp

- **Transformación afín**. Transformación geométrica que conserva las líneas y el paralelismo (pero no necesariamente distancias y ángulos).
- Transformaciones afines: traslación, reflexión (espejo), escala, rotación y shearing.

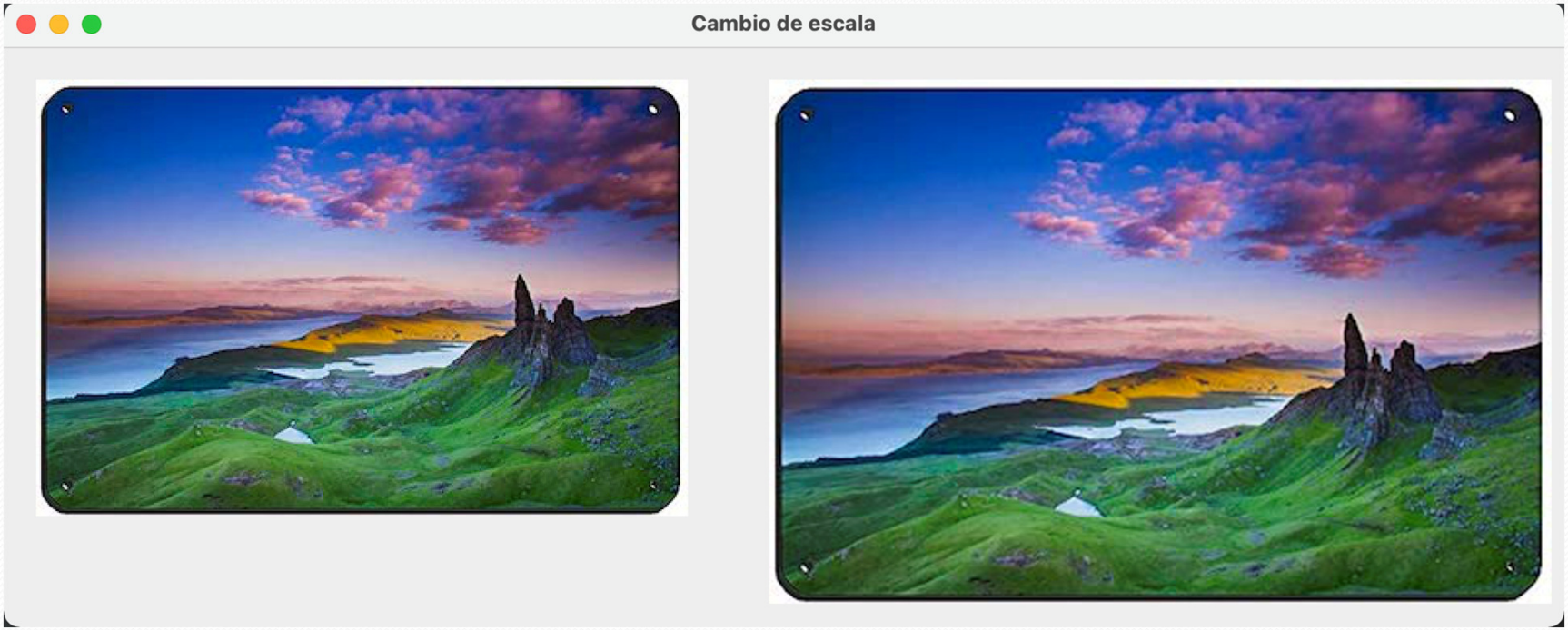
AffineTransformOp



https://www.cs.drexel.edu/~david/Classes/CS536/Lectures/L-18_Transformations.pdf

AffineTransformOp

- Ejemplo: escalar una imagen en un factor de 1.2 (20% más grande).



AffineTransformOp

```
try {
    BufferedImage image = ImageIO.read(new File("landscape400.png"));
    Map<RenderingHints.Key, Object> mapHints = new HashMap<>();
    mapHints.put(RenderingHints.KEY_ANTIALIASING,
                    RenderingHints.VALUE_ANTIALIAS_ON);
    mapHints.put(RenderingHints.KEY_INTERPOLATION,
                    RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    AffineTransformOp transformOp =
        new AffineTransformOp(AffineTransform.getScaleInstance(1.2, 1.2),
        new RenderingHints(mapHints));
```

AffineTransformOp

```
        g2.drawImage(image, 20, 20, null);
        g2.drawImage(image, transformOp, 470, 20);
    }
    catch (IOException e) {
        System.out.println("Error: " + e);
    }
```

Más información

- En el tutorial de Java: https://docs.oracle.com/javase/tutorial/2d/TOC.html