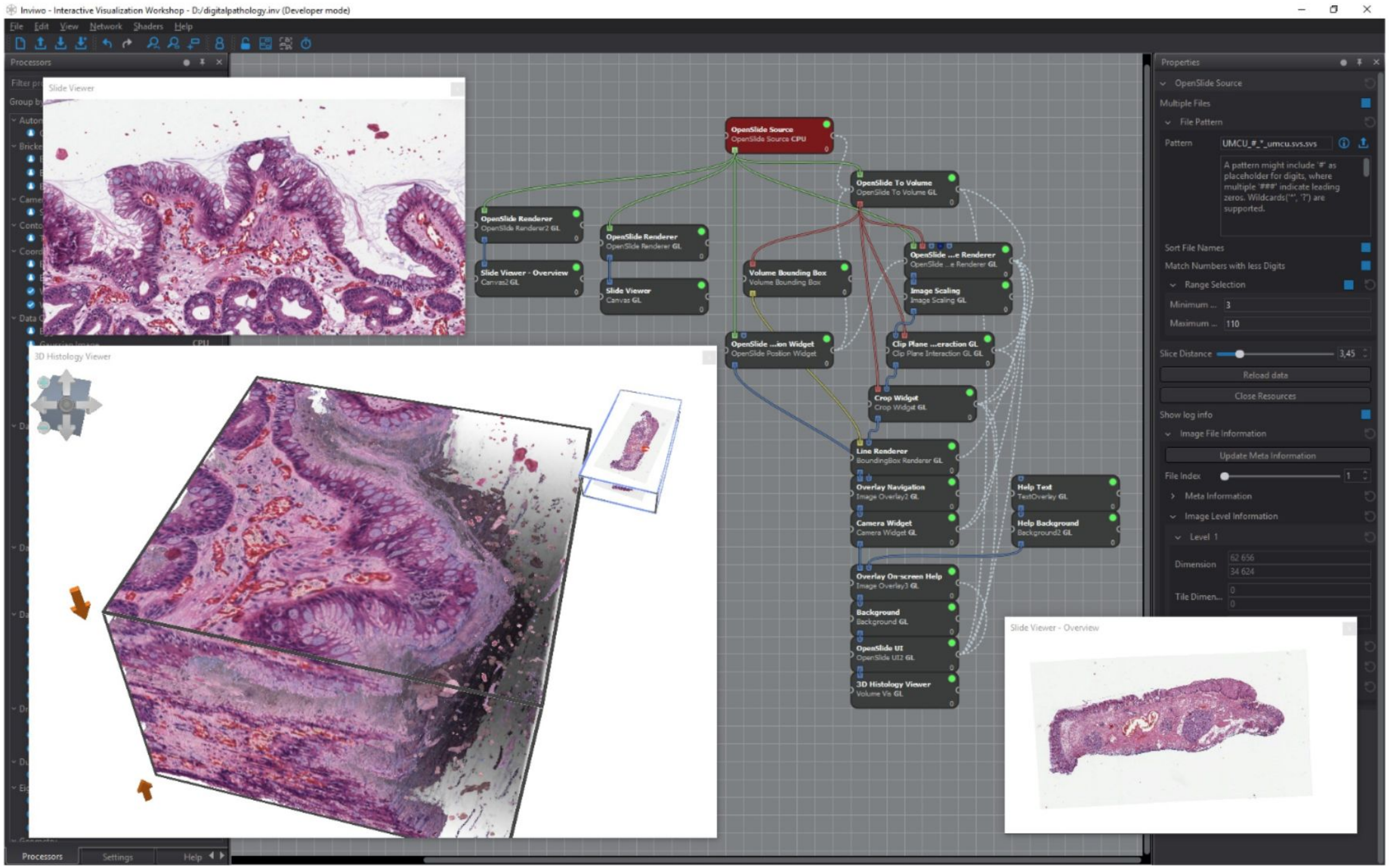
Inviwo — A Visualization System with Usage Abstraction Levels

Lucas Zamprogno

More info, video, and more at https://inviwo.org/

Motivations

- Accessibility: Should be accessible to those without programming experience
- Performance: Take advantage of low-level optimizations
- Adaptability: Work with many algorithms and technologies at once



Properties

OpenSlide Source

Multiple Files

File Pattern

Pattern UMCU_#_*_umcu.svs.svs

A pattern might include '#' as placeholder for digits, where multiple '###' indicate leading zeros. Wildcards('*', '?') are supported.

Sort File Names

Match Numbers with less Digits

Range Selection

Minimum ... 3

Maximum ... 110

Slice Distance 3,45

Show log info

Image File Information

File Index 1

Meta Information

Image Level Information

Level 1

Dimension 62 656
34 624

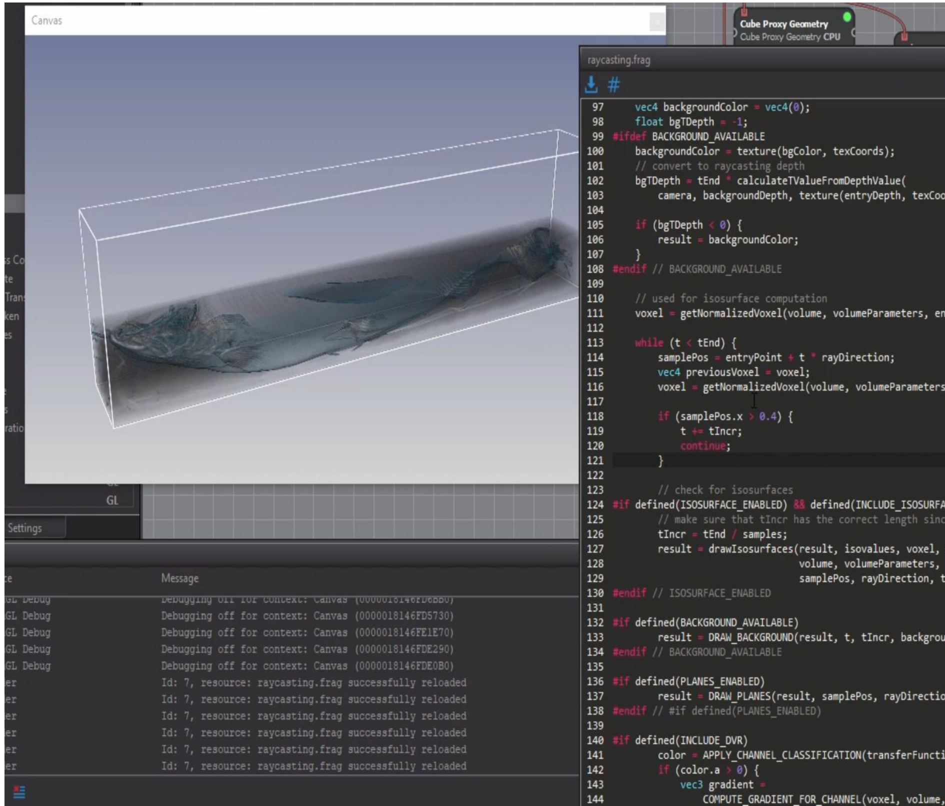
Tile Dimen... 0
0

Design Principles: Interactive Development

- Want to be able to view low-level change impacts
- Requires recompiling and/or observing source code changes
- Observe changes throughout pipeline

Inviwo Implementation: Interactive Development

- Integrated editor allows for low-level code changes that are immediately reflected in the output



```
    vec4 backgroundColor = vec4(0);
    float bgTDepth = -1;
#ifdef BACKGROUND_AVAILABLE
    backgroundColor = texture(bgColor, texCoords);
    // convert to raycasting depth
    bgTDepth = tEnd * calculateTValueFromDepthValue(
        camera, backgroundDepth, texture(entryDepth, texCoo

    if (bgTDepth < 0) {
        result = backgroundColor;
    }
#endif // BACKGROUND_AVAILABLE

    // used for isosurface computation
    voxel = getNormalizedVoxel(volume, volumeParameters, en

    while (t < tEnd) {
        samplePos = entryPoint + t * rayDirection;
        vec4 previousVoxel = voxel;
        voxel = getNormalizedVoxel(volume, volumeParameters

        if (samplePos.x > 0.4) {
            t += tIncr;
            continue;
        }

        // check for isosurfaces
#if defined(ISOSURFACE_ENABLED) && defined(INCLUDE_ISOSURFA
        // make sure that tIncr has the correct length sinc
        tIncr = tEnd / samples;
        result = drawIsosurfaces(result, isovalues, voxel,
                                 volume, volumeParameters,
                                 samplePos, rayDirection, t
#endif // ISOSURFACE_ENABLED

#if defined(BACKGROUND_AVAILABLE)
        result = DRAW_BACKGROUND(result, t, tIncr, backgrou
#endif // BACKGROUND_AVAILABLE

#if defined(PLANES_ENABLED)
        result = DRAW_PLANES(result, samplePos, rayDirectio
#endif // #if defined(PLANES_ENABLED)

#if defined(INCLUDE_DVR)
        color = APPLY_CHANNEL_CLASSIFICATION(transferFuncti
        if (color.a > 0) {
            vec3 gradient =
                COMPUTE_GRADIENT_FOR_CHANNEL(voxel, volume,
```

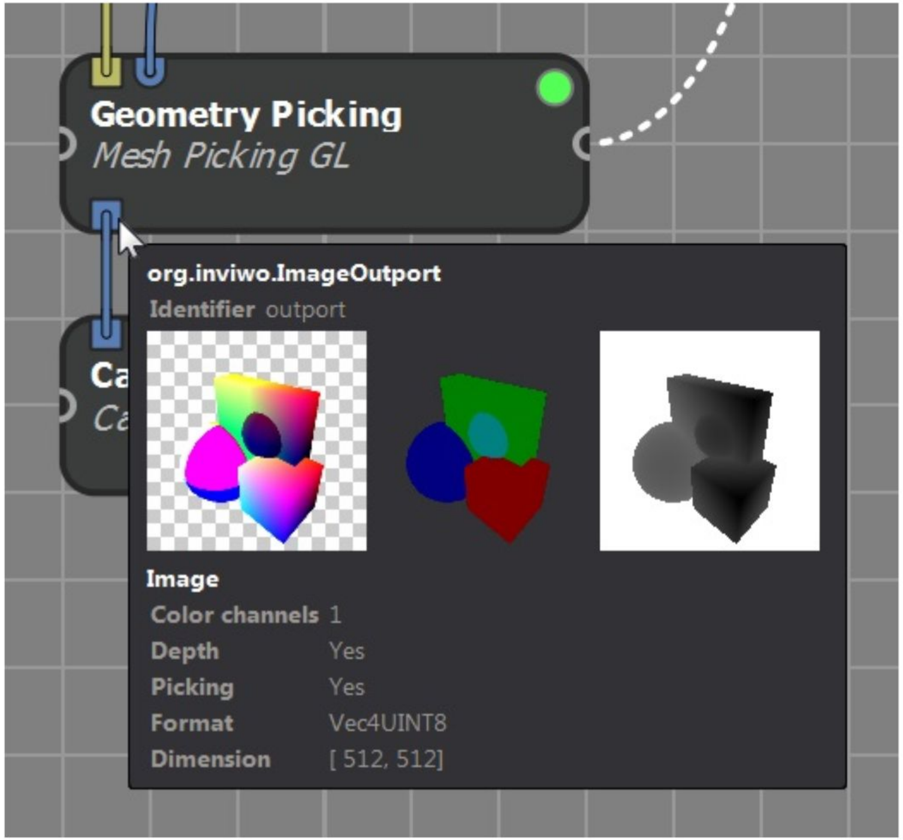
```
e                  Message
GL Debug           Debugging off for context: Canvas (0000018146FD6BB0)
GL Debug           Debugging off for context: Canvas (0000018146FD5730)
GL Debug           Debugging off for context: Canvas (0000018146FE1E70)
GL Debug           Debugging off for context: Canvas (0000018146FDE290)
GL Debug           Debugging off for context: Canvas (0000018146FDE0B0)
er                 Id: 7, resource: raycasting.frag successfully reloaded
er                 Id: 7, resource: raycasting.frag successfully reloaded
er                 Id: 7, resource: raycasting.frag successfully reloaded
er                 Id: 7, resource: raycasting.frag successfully reloaded
er                 Id: 7, resource: raycasting.frag successfully reloaded
er                 Id: 7, resource: raycasting.frag successfully reloaded
```

Design Principles: Debugging

- Typically challenging to debug a multi-stage pipeline
- Idea: Have “ports” in and out of each stage to view intermediate data
- Different data types can support different views

Inviwo Implementation: Debugging

- Inspect output from each processor in the pipeline



org.inviwo.ImageOutport
Identifier outport

Image
Color channels 1
Depth Yes
Picking Yes
Format Vec4UINT8
Dimension [512, 512]

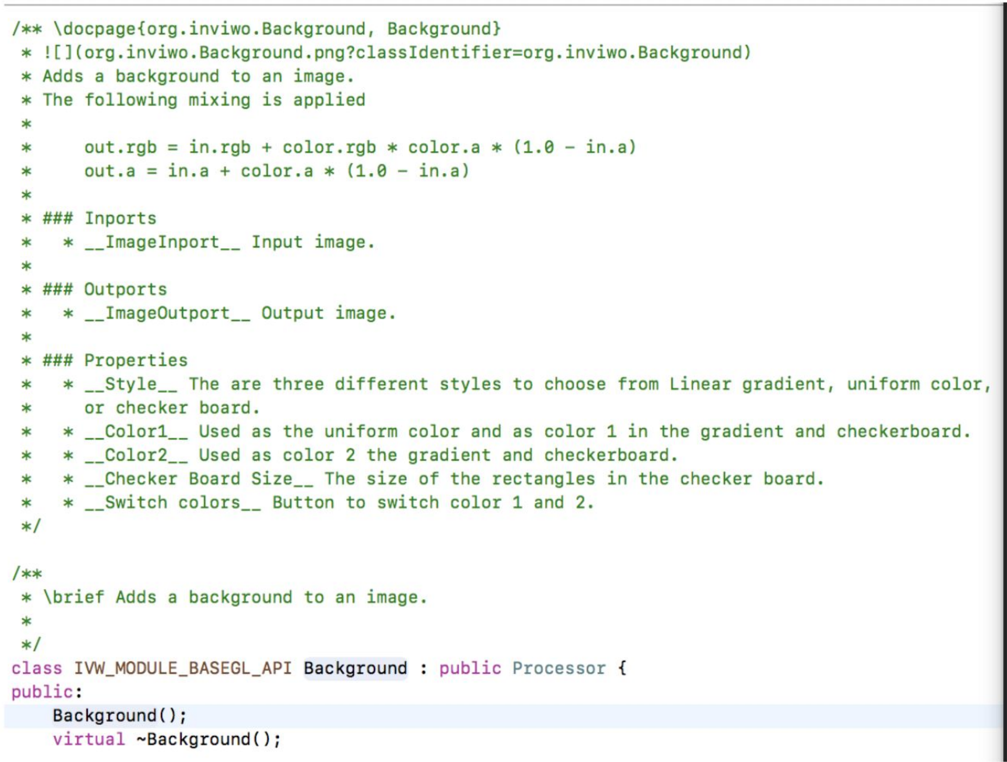
Design Principles: Documentation

- Tends to be targeted at developers, but vis designers need access

- Suggest incorporating it into the API designers use

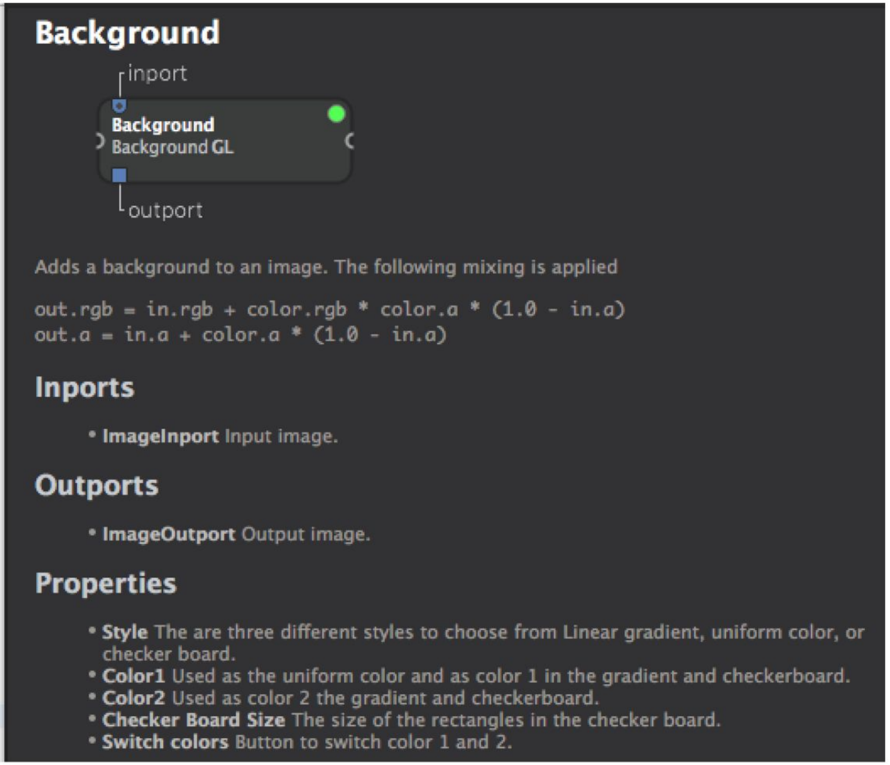
- Tailor documentation appropriately

Inviwo Implementation: Documentation



```
/** \docpage{org.inviwo.Background, Background}
 * ![](org.inviwo.Background.png?classIdentifier=org.inviwo.Background)
 * Adds a background to an image.
 * The following mixing is applied
 *
 *     out.rgb = in.rgb + color.rgb * color.a * (1.0 - in.a)
 *     out.a = in.a + color.a * (1.0 - in.a)
 *
 * ### Inports
 *   * __ImageInport__ Input image.
 *
 * ### Outports
 *   * __ImageOutport__ Output image.
 *
 * ### Properties
 *   * __Style__ The are three different styles to choose from Linear gradient, uniform color,
 *     or checker board.
 *   * __Color1__ Used as the uniform color and as color 1 in the gradient and checkerboard.
 *   * __Color2__ Used as color 2 the gradient and checkerboard.
 *   * __Checker Board Size__ The size of the rectangles in the checker board.
 *   * __Switch colors__ Button to switch color 1 and 2.
 */

/**
 * \brief Adds a background to an image.
 *
 */
class IVW_MODULE_BASEGL_API Background : public Processor {
public:
    Background();
    virtual ~Background();
```



Background

inport
Background
Background GL
outport

Adds a background to an image. The following mixing is applied

```
out.rgb = in.rgb + color.rgb * color.a * (1.0 - in.a)
out.a = in.a + color.a * (1.0 - in.a)
```

Inports

- **ImageInport** Input image.

Outports

- **ImageOutport** Output image.

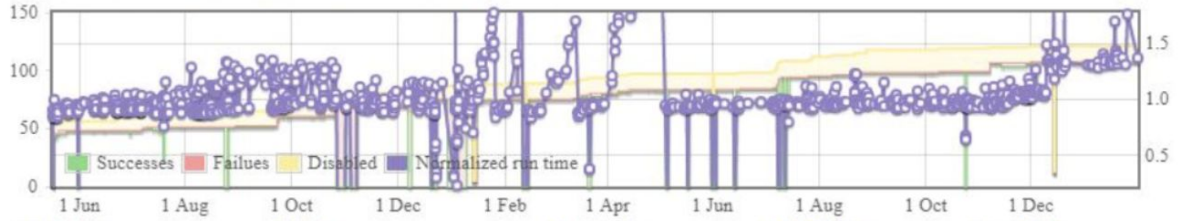
Properties

- **Style** The are three different styles to choose from Linear gradient, uniform color, or checker board.
- **Color1** Used as the uniform color and as color 1 in the gradient and checkerboard.
- **Color2** Used as color 2 the gradient and checkerboard.
- **Checker Board Size** The size of the rectangles in the checker board.
- **Switch colors** Button to switch color 1 and 2.

Design Principles: Testing

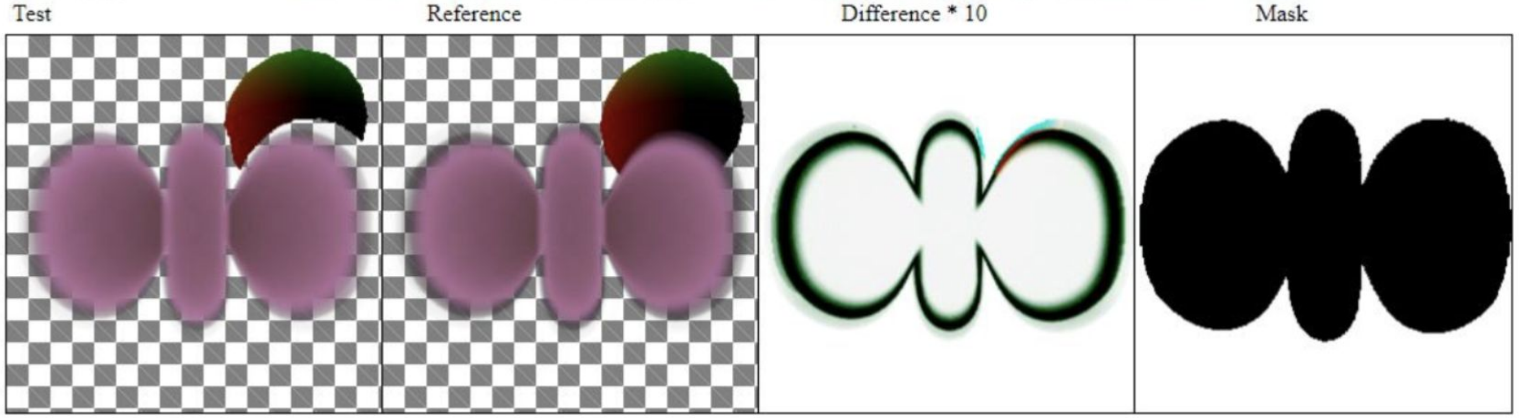
- Unit testing often used for low-level code
- Open that option up to high-level designers
- Need to compensate for hardware differences

Inviwo Implementation: Testing



Module	Name	Failures ▲	Run Time	Last Run
basegl	raycasterwithmeshes	1	1.77s	2016-07-14 15:18:21
Images	1/1 Images failed			

1. CanvasGL.png Difference: 1.54393364% Pixels: 30945 (47.2183%) Largest delta: 0.37843137



Design Principles: Interoperability

- For performance reasons, want access to computing platforms like OpenGL, OpenCl, CUDA

- Algorithms in one system can’t easily interact with others

- This can be a challenge with new technology not being compatible with existing algorithms

Inviwo Implementation: Interoperability

- Different computing systems can be chained together

- Get the low level optimizations of each

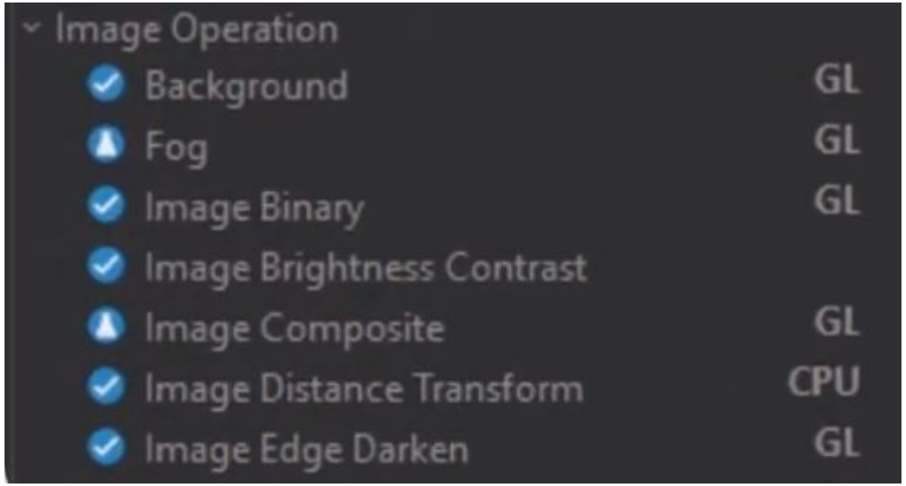


Image Operation

Background	GL
Fog	GL
Image Binary	GL
Image Brightness Contrast	
Image Composite	GL
Image Distance Transform	CPU
Image Edge Darken	GL

Inviwo Demo Video

https://www.youtube.com/watch?v=9yZWjxlV6OQ

Highlights

- Already shows good adoption
- Open source
- Extensible and user friendly

Highlights

- Supports software engineering good practices:
    - Debugger support
    - Documentation integration
    - Unit and integration/regression testing support

Paper Critiques

- Almost no discussion of their own limitations

- Seemed to oversell some points
    - Default implementation of port debugging
    - Running Inviwo tests on the same machine

Questions/Discussion