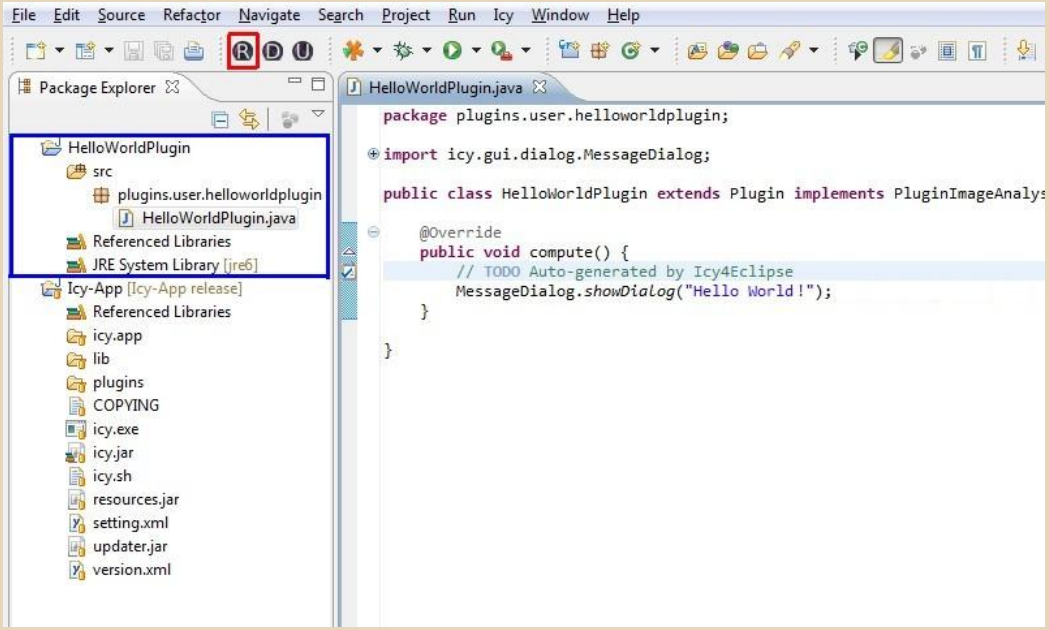
Icy Plugin Development

Training - Level 3

Icy Development Environment

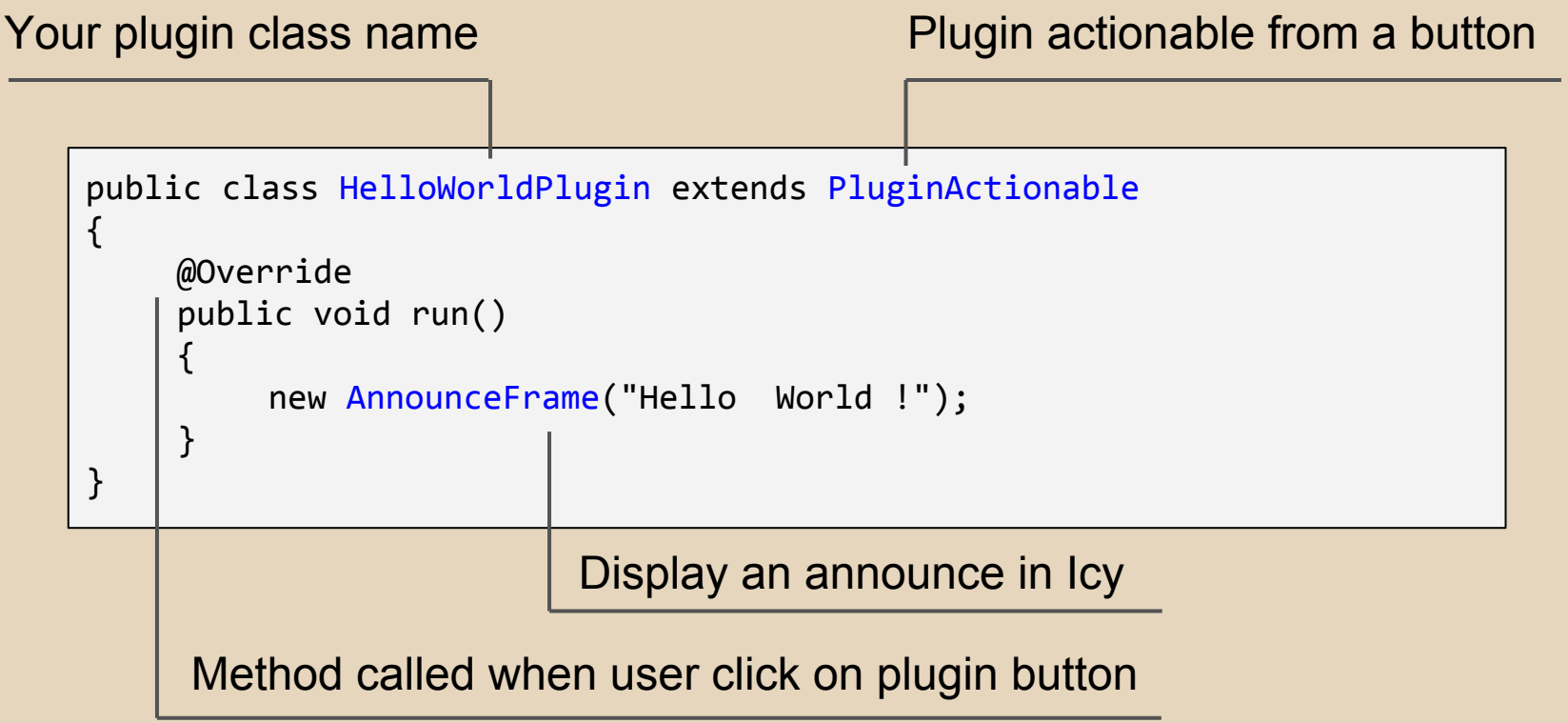
Follow the **Quick Start in 4 steps** tutorial:

http://icy.bioimageanalysis.org/index.php?display=startDevWithIcy



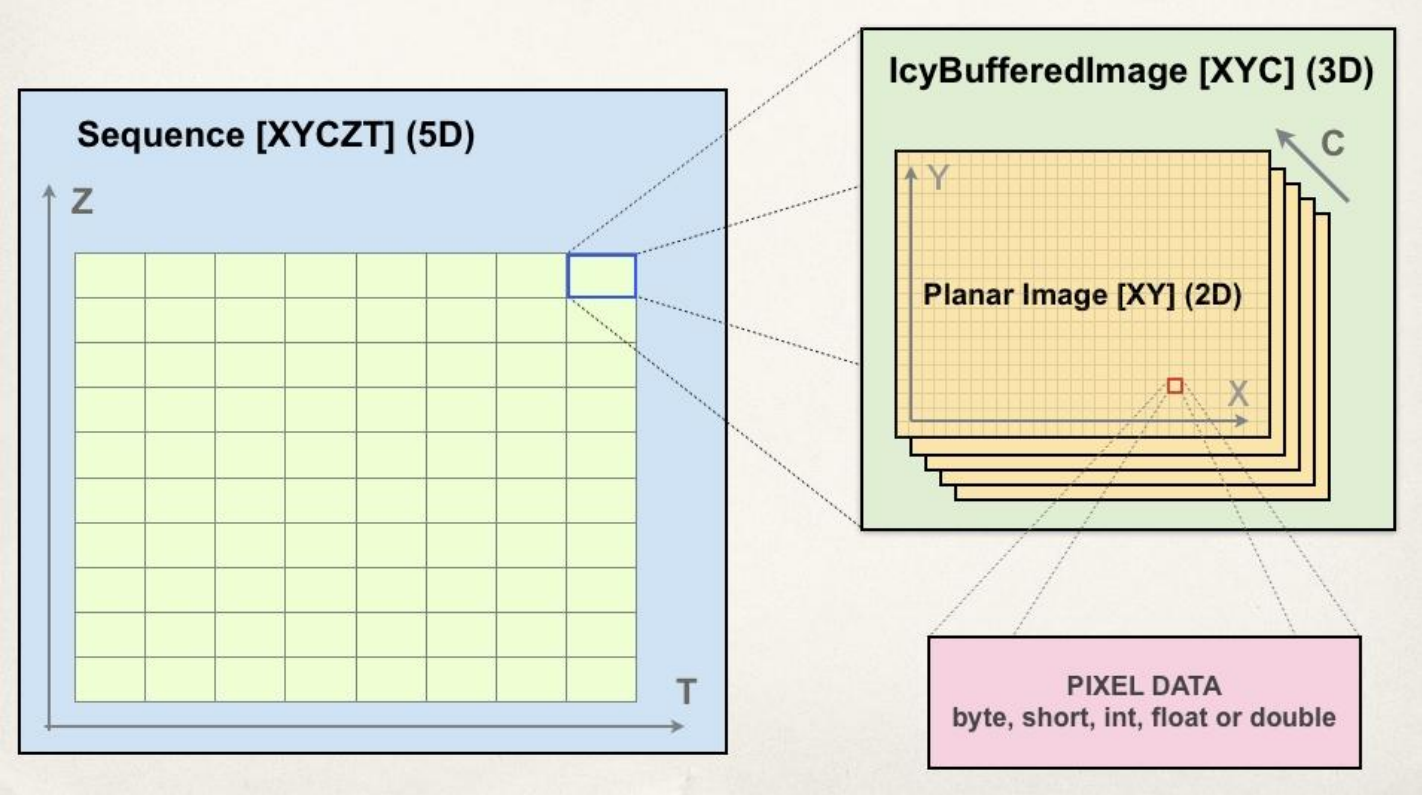
Hello World ! (in Icy)

Something as simple as...



Data structure

5D image data organized by channel



Any data type !

Working with image

The basics…

Get the current active image

```
IcyBufferedImage image = getActiveImage();

// check if an image is opened
if (image == null)
{
     MessageDialog.showDialog("This plugin needs an opened image.");
     return;
}
```

Display a message if no image is opened

Retrieve image information

```
int w = image.getSizeX();
```
Get the image width

```
int h = image.getSizeY();
```
Get the image height

```
int numChannel = image.getSizeC();
```
Get the number of channel

```
DataType type = image.getDataType_();
```
Get the image data type

```
double pixel = image.getData(0, 0, 0);
```
Get the pixel value at specified X, Y, C position

Modify an image

Divide intensity by 2 in a single image

Naive way…

```
for (int x = 0; x < w;  x++)
{
     for (int y = 0; y < h; y++)
     {
          image.setData(x, y, 0, image.getData(x, y, 0) / 2);
     }
}
```

Get pixel value at position X, Y, C

Set pixel value at position X, Y, C

Iterate through all pixels of the image

Modify an image

Divide intensity by 2 in a single image

```
for (int x = 0; x < w;  x++)
{
     for (int y = 0; y < h;  y++)
     {
          image.setData(x,  y, 0, image.getData(x,  y, 0) / 2);
     }
}
```

Slow ! Because image.setData(..) causes image refresh events and other recalculations.

Modify an image

Better way…

Start image modification

```
image.beginUpdate();
try
{
     for (int x = 0; x < w;  x++)
     {
          …
     }
}
finally
{
     image.endUpdate();
}
```

End image modification → image refresh and recalculations

Modify an image

Even better…

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
     image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
     image.isSignedDataType());

image.dataChanged();
```

Let’s explain…

Modify an image

Get a direct reference on the XY planar image data array for the first channel regardless of the data type

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
    image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
    image.isSignedDataType());

image.dataChanged();
```

Modify an image

Convert the data array in double type array for precise mathematical operations by using the `Array1DUtil.arrayToDoubleArray(..)` method

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
     image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
     image.isSignedDataType());

image.dataChanged();
```

Modify an image

Divide contents of the double data array by 2.
We can use the `MathUtil.divide(..)` method to help here.

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
    image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
    image.isSignedDataType());

image.dataChanged();
```

Modify an image

Convert the double data array back to the original image data type by using the `Array1DUtil.doubleArrayToArray(..)` method (or `Array1DUtil.doubleArrayToSafeArray(..)` for “safe” data bounds)

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
    image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
    image.isSignedDataType());

image.dataChanged();
```

Modify an image

Inform the image that its data has been modified !

Why? Because we directly modified the internal data array without using any `setDataxxx(..)` method => the image does not know it has been modified => we have to do it manually with `dataChanged()`

```
Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
    image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
    image.isSignedDataType());

image.dataChanged();
```

Image changed → image refresh and recalculations

Modify an image

Final version…

```
IcyBufferedImage image = getActiveImage();

if (image == null)
{
     MessageDialog.showDialog("This plugin needs an opened image.");
     return;
}

Object dataArray = image.getDataXY(0);

double[] doubleDataArray = Array1DUtil.arrayToDoubleArray(dataArray,
     image.isSignedDataType());

MathUtil.divide(doubleDataArray, 2d);

Array1DUtil.doubleArrayToSafeArray(doubleDataArray, image.getDataXY(0),
     image.isSignedDataType());

image.dataChanged();
```

Image utilities…

IcyBufferedImageUtil class

```
// Add a channel to the image
image = IcyBufferedImageUtil.addChannel(image);

// Change the image data type
image = IcyBufferedImageUtil.convertToType(image, DataType.DOUBLE, true);

// Extract a channel
image = IcyBufferedImageUtil.extractChannel(image, 1);

// Get a region of the image
image = IcyBufferedImageUtil.getSubImage(image,
     new Rectangle(10, 10, 50, 50), 0, 1);

// Resize the image
image = IcyBufferedImageUtil.scale(image, 500, 500);
```

Sequence processing & creation

Compute the maximum intensity projection over the Z dimension

Get the current active sequence

```
Sequence sequence = getActiveSequence();

// check if a sequence is opened
if (sequence  == null)
{
     MessageDialog.showDialog("This plugin needs an opened sequence.");
     return;
}
```

Display a message if no sequence is opened

Sequence processing & creation

the idea...

```
Sequence result = new Sequence(sequence.getName() + " - Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

Create a new empty sequence by using the original name

```
Sequence result = new Sequence(sequence.getName() + " - Z projection");

result.beginUpdate();
try
{
    for (int t = 0; t < sequence.getSizeT(); t++)
    {
        IcyBufferedImage image = getMaxZProjection(sequence, t);
        result.setImage(t, 0, image);
    }
}
finally
{
    result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

Start sequence modification

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

For each frame of the sequence

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

Compute the Z projection for the specified frame of sequence and save the result in `image`

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

Set the `image` result at frame position `t` in the result Sequence

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

End sequence modification → refresh display when visible and internals recalculations

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
    for (int t = 0; t < sequence.getSizeT(); t++)
    {
        IcyBufferedImage image = getMaxZProjection(sequence, t);
        result.setImage(t, 0, image);
    }
}
finally
{
    result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

Make the sequence visible (add it to the GUI)

```
Sequence result = new Sequence(sequence.getName() + "- Z projection");

result.beginUpdate();
try
{
     for (int t = 0; t < sequence.getSizeT(); t++)
     {
          IcyBufferedImage image = getMaxZProjection(sequence, t);
          result.setImage(t, 0, image);
     }
}
finally
{
     result.endUpdate();
}

addSequence(result);
```

Sequence processing & creation

getMaxZProjection(..) method

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

Create a new image with same size and data type as the original Sequence

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMaxZ(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

For each channel of the input Sequence

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

Convert the result image data to double type for calculations

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

For each Z slice of the input Sequence

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

Compute the maximum between the specified slice data and the result array `doubleArray`

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
     IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
          sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

     for (int c = 0; c < sequence.getSizeC(); c++)
     {
          double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
               result.isSignedDataType());

          for (int z = 0; z < sequence.getSizeZ(); z++)
               projectMax(sequence, t, z, c, doubleArray);

          Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
     }

     result.dataChanged();

     return result;
}
```

Sequence processing & creation

Convert back the `doubleArray` to original image data type

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

Inform that the image data has changed

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

And return the result

```
IcyBufferedImage getMaxZProjection(Sequence sequence, int t)
{
      IcyBufferedImage result = new IcyBufferedImage(sequence.getSizeX(),
            sequence.getSizeY(), sequence.getSizeC(), sequence.getDataType_());

      for (int c = 0; c < sequence.getSizeC(); c++)
      {
            double[] doubleArray = Array1DUtil.arrayToDoubleArray(result.getDataXY(c),
                  result.isSignedDataType());

            for (int z = 0; z < sequence.getSizeZ(); z++)
                  projectMax(sequence, t, z, c, doubleArray);

            Array1DUtil.doubleArrayToArray(doubleArray, result.getDataXY(c));
      }

      result.dataChanged();

      return result;
}
```

Sequence processing & creation

projectMax(..) method, easy...

```
void projectMax(Sequence sequence, int t, int z, int c, double[] result)
{
     Object dataArray = sequence.getDataXY(t, z, c);

     double[] imgDoubleArray = Array1DUtil.arrayToDoubleArray(dataArray,
          sequence.isSignedDataType());

     ArrayMath.max(result, imgDoubleArray, result);
}
```

Sequence processing & creation

Get XY planar image data array for the specified T, Z and C position

```
void projectMax(Sequence sequence, int t, int z, int c, double[] result)
{
    Object dataArray = sequence.getDataXY(t, z, c);

    double[] imgDoubleArray = Array1DUtil.arrayToDoubleArray(dataArray,
        sequence.isSignedDataType());

    ArrayMath.max(result, imgDoubleArray, result);
}
```

Sequence processing & creation

Convert to double data array

```
void projectMax(Sequence sequence, int t, int z, int c, double[] result)
{
    Object dataArray = sequence.getDataXY(t, z, c);

    double[] imgDoubleArray = Array1DUtil.arrayToDoubleArray(dataArray,
        sequence.isSignedDataType());

    ArrayMath.max(result, imgDoubleArray, result);
}
```

Sequence processing & creation

Get the maximum from the 2 input arrays and save the result into the `result` array by using the `ArrayMath.max(..)` method

```
void projectMax(Sequence sequence, int t, int z, int c, double[] result)
{
    Object dataArray = sequence.getDataXY(t, z, c);

    double[] imgDoubleArray = Array1DUtil.arrayToDoubleArray(dataArray,
        sequence.isSignedDataType());

    ArrayMath.max(result, imgDoubleArray, result);
}
```

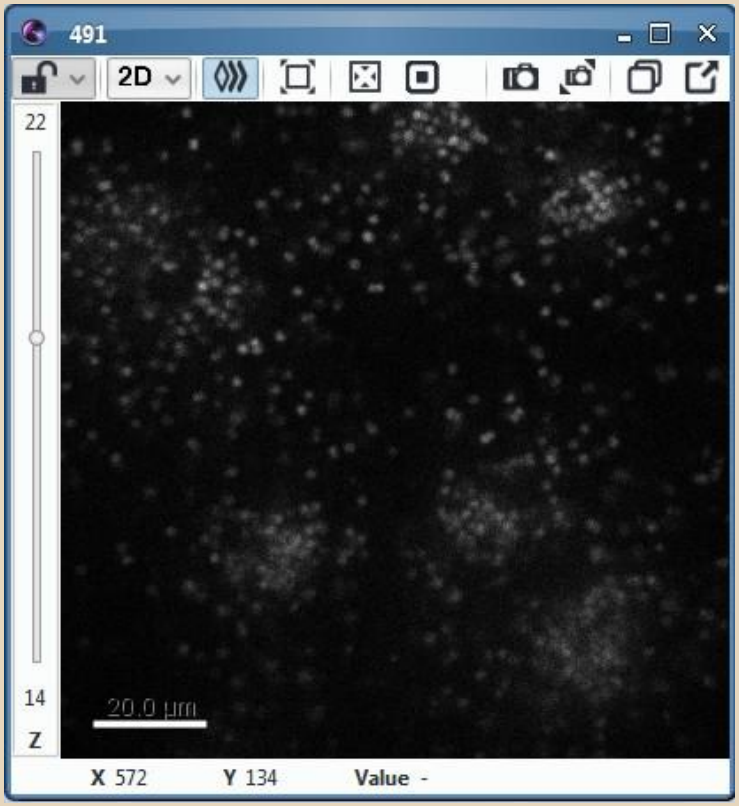
That’s it !

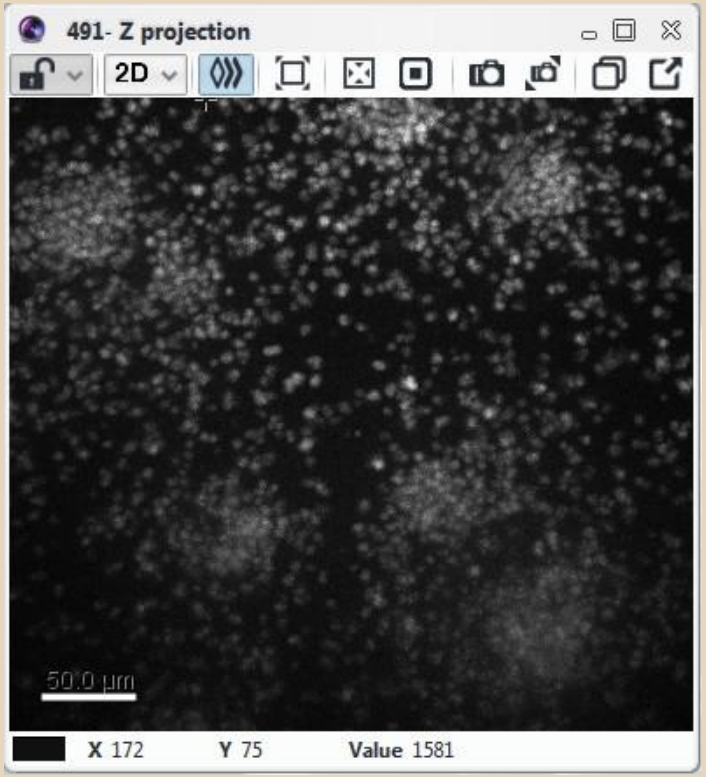
Sequence processing & creation

Before

Z max projection

After





Sequence utilities…

SequenceUtil class

```
// Get a copy of specified sequence
result = SequenceUtil.getCopy(sequence);

// Add one Z slice at position 5
SequenceUtil.addZ(sequence, 5, 1);

// Convert the specified sequence (Z-stack format) to timelapse
SequenceUtil.convertToTime(sequence);

// Remove the frame 12
SequenceUtil.removeTAndShift(sequence, 12);

// Get a sub-region of the sequence
result = SequenceUtil.getSubSequence(sequence,
     new Rectangle5D.Integer(50, 50, 0, 0, 0, 100, 100, sequence.getSizeZ(),
     sequence.getSizeT(), 1));
```

Thread & good practices

The `run` method of the plugin is always called on the *Event Dispatch Thread* (graphic thread) as you need it to create your graphical interface. But if you do a long process in this thread the application won't respond for sometime.

```
public class StupidPlugin extends PluginActionable
{
     @Override
     public void run()
     {
          // We are in the Event Dispatch Thread

          // this is a really bad idea !
          Thread.sleep(10000);
     }
}
```

Thread & good practices

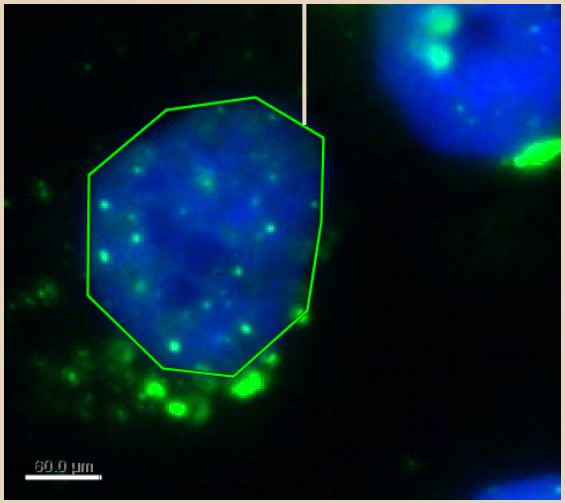
ThreadUtil class

```
public class SmartPlugin extends PluginActionable
{
     @Override
     public void run()
     {
          // WE ARE IN THE EVENT DISPATCH THREAD (GRAPHICS THREAD)

          // request the following Runnable to execute in background thread
          ThreadUtil.bgRun(new Runnable()
          {
               @Override
               public void run()
               {
                    // WE ARE NOT ANYMORE IN THE EVENT DISPATCH THREAD
                    ThreadUtil.Sleep(10000);
               }
          });
     }
}
```

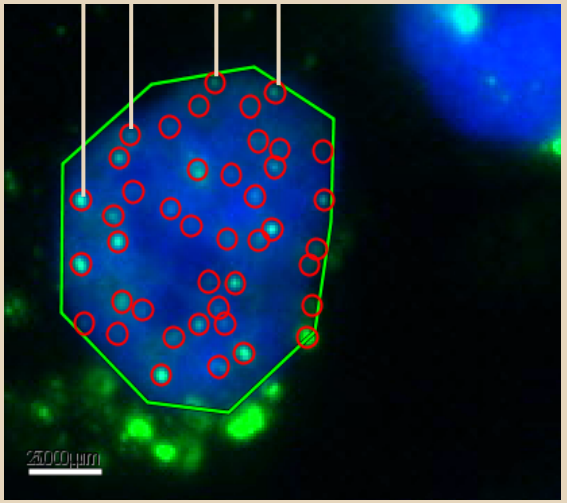
ROI

A good structure to give input to plugins…

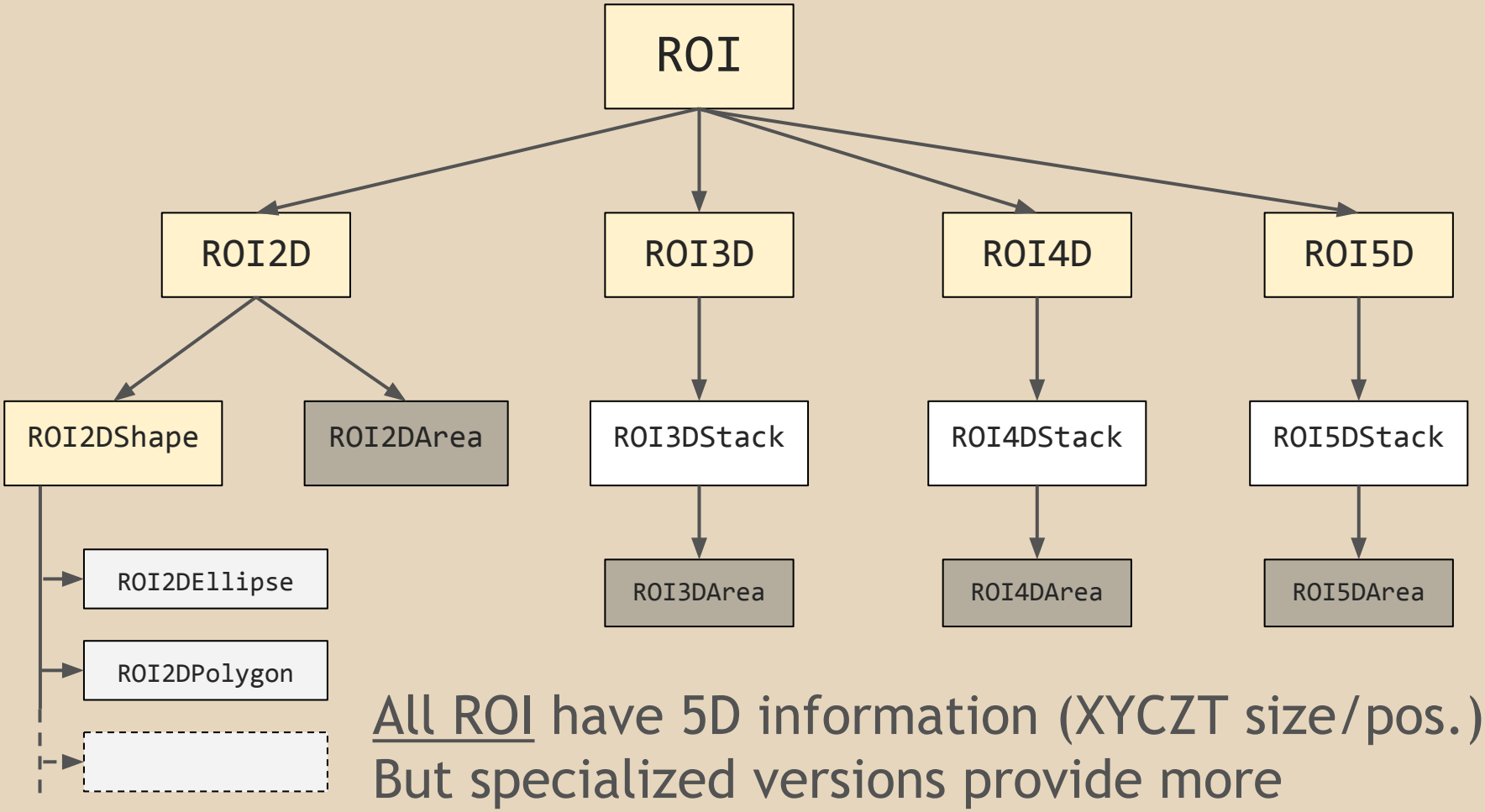
… but also to provide results



Processing



ROI - class hierarchy



All ROI have 5D information (XYCZT size/pos.)
But specialized versions provide more functionalities & operations

ROI - basics

Generic 5D contains method

```
boolean contained = roi.contains(x,  y, z, t, c);

if (roi instanceof ROI2D)
{
     contained = ((ROI2D) roi).contains(x,  y);
}
```

2D specific test

2D specialized contains method

Get all 2D roi from this sequence

Get all selected roi from this sequence

```
List<ROI2D> roi2ds = sequence.getROI2Ds();

List<ROI> rois = sequence.getSelectedROIs();

ROI roi = sequence.getSelectedROI();
```

Get first selected ROI from this sequence

ROI - mean intensity

Simple method using `roi.contains(...)`:

```
// consider first image only here
IcyBufferedImage image = sequence.getFirstImage();
double mean = 0;
double sample = 0;

for(int x = 0; x < sequence.getSizeX(); x++)
{
     for(int y = 0; y < sequence.getSizeY(); y++)
     {
          if (roi.contains(x, y))
          {
               mean += image.getData(x, y, 0);
               sample++;
          }
     }
}

System.out.println("mean intensity over ROI: " + (mean / sample));
```

Roi 'contains' test (can be slow)

ROI - mean intensity

Problem:

The roi `contains()` method can be slow depending the ROI implementation.

→ slow processing on large images

Solution:

Use `BooleanMask2D` object for fast `contains()` method.

ROI - mean intensity

Faster method using `BooleanMask2D`:

```
BooleanMask2D mask = roi.getBooleanMask(true);
// consider first image only here
IcyBufferedImage image = sequence.getFirstImage();
double mean = 0;
double sample = 0;

for(int x = 0; x < sequence.getSizeX();  x++)
{
     for(int y = 0; y < sequence.getSizeY();  y++)
     {
          if (mask.contains(x,  y))
          {
               mean += image.getData(x,  y, 0);
               sample++;
          }
     }
}
```

Mask ‘contains’ test (fast)

ROI - mean intensity

Alternative method using DataIterator:

Create a new Sequence data iterator that iterates through all pixels contained in a ROI

```
SequenceDataIterator iterator = new SequenceDataIterator(sequence, roi);
double mean = 0;
double sample = 0;

while(!iterator.done())
{
     mean += iterator.get();
     iterator.next();
     sample++;
}
```

Get the current pixel value

Move to the next pixel

While we still have pixels to process

ROI - Threshold result

The idea : Create a ROI which contains all pixel value >= threshold value

→ many different ways to do that !

Let’s see some of them…

ROI - Threshold result - 1

Naive way..

Create a new empty ROI (type: mask)

```
ROI2DArea roi = new ROI2DArea();
// consider first image only here
IcyBufferedImage image = sequence.getFirstImage();

for (int x = 0; x < sequence.getSizeX(); x++)
{
     for (int y = 0; y < sequence.getSizeY(); y++)
     {
          if (image.getData(x, y, 0) >= threshold)
          {
               roi.addPoint(x, y);
          }
     }
}

sequence.addROI(roi);
```

Add the accepted point to the ROI

Attach the result ROI to the sequence (make it visible)

ROI - Threshold result - 2

Faster method...

Get sequence data in double array format

```
// consider first image and first channel only here
double[] doubleArray = Array1DUtil.arrayToDoubleArray(
     sequence.getDataXY(0, 0, 0), sequence.isSignedDataType());
boolean[] mask = new boolean[doubleArray.length];
```

Create a boolean array with same size as the data array

Set mask to true where pixels are accepted

```
for (int i = 0; i < doubleArray.length; i++)
     mask[i] = (doubleArray[i] >= threshold);

BooleanMask2D mask2d = new BooleanMask2D(sequence.getBounds2D(), mask);
ROI2DArea roi = new ROI2DArea(mask2d);
```

Create a BooleanMask2D object and then a ROI from this mask

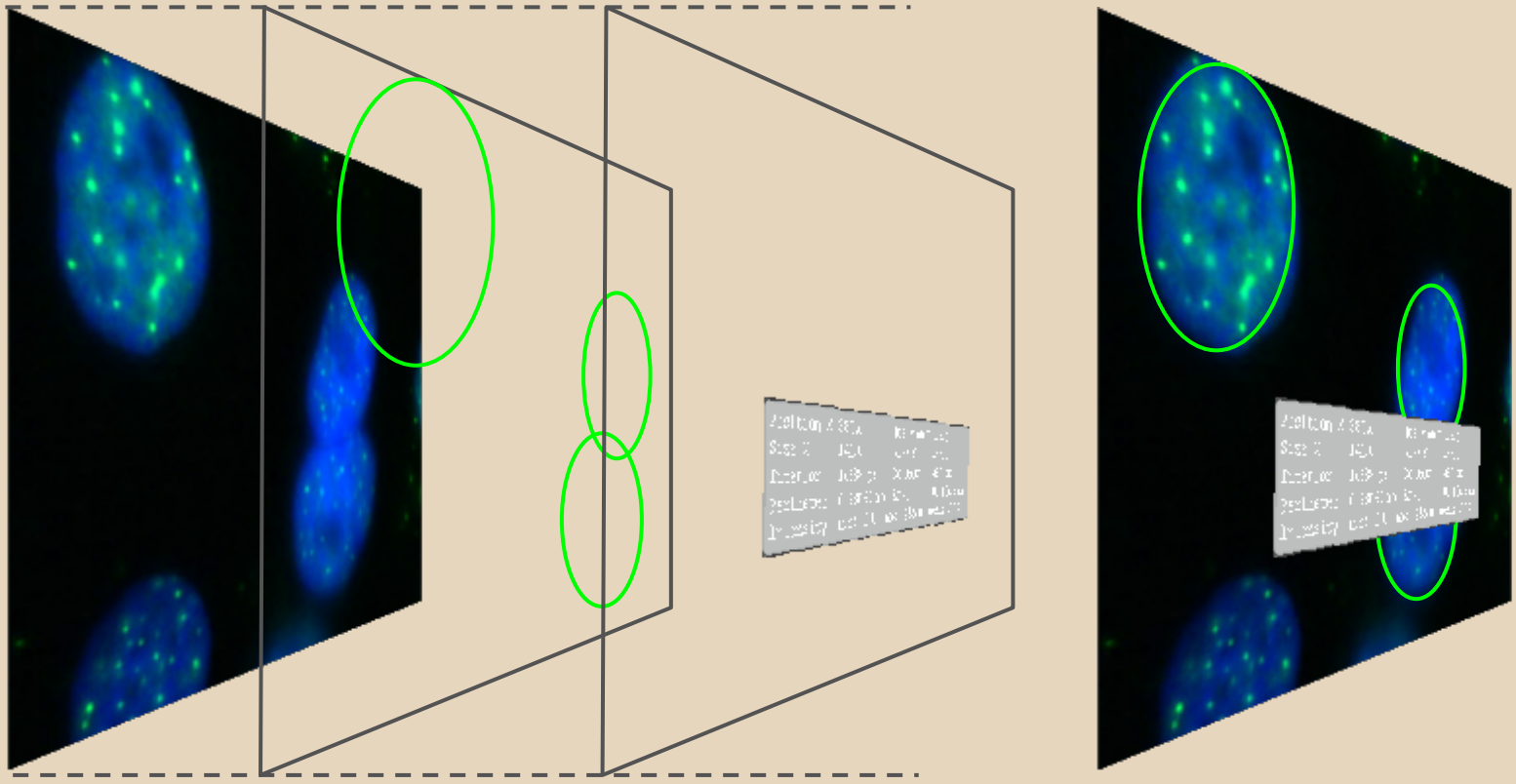
ROI utilities…

ROIUtil class

```
// Get the area of the specified roi and return result with correct units.
area = ROIUtil.getArea(sequence, roi);

// Return the mean intensity of sequence pixels contained in the ROI.
meanIntensity = ROIUtil.getMeanIntensity(sequence, roi);

// Process the union of specified rois
unionRoi = ROIUtil.getUnion(rois);

// Process the subtraction of roi2 to roi1
roi= ROIUtil.subtract(roi1, roi2);
```

Overlay - concept

Image + ROI + Tooltip = **View**



Overlay - description

Plugins can use Overlays to display rich information over images...

...but it also allows user interaction by receiving directly mouse and key events.

Overlay - display

Create a new Overlay class

```
public class SimpleCrossOverlay extends Overlay
{
    public SimpleCrossOverlay()
    {
        super("Simple cross");
    }

    @Override
    public void paint(Graphics2D g, Sequence sequence, IcyCanvas canvas)
    {
        if (g != null)
        {
            // paint a yellow cross all over the image
            g.setColor(Color.YELLOW);
            g.setStroke(new BasicStroke(5));
            g.drawLine(0, 0, sequence.getWidth(), sequence.getHeight());
            g.drawLine(0, sequence.getHeight(), sequence.getWidth(), 0);
        }
    }
```

Name (appears in the ‘Layers’ tab)

Method called by Icy to draw the overlay

Drawing is relative to the image coordinate system

Overlay - user interaction

Mouse click event method

Mouse position in image coordinate space

```
    ...

    @Override
    public void mouseClick(MouseEvent e, Point2D point, IcyCanvas canvas)
    {
        // remove the overlay when the user clicks on the image
        remove();
    }
}
```

Remove the Overlay from sequences where it is attached

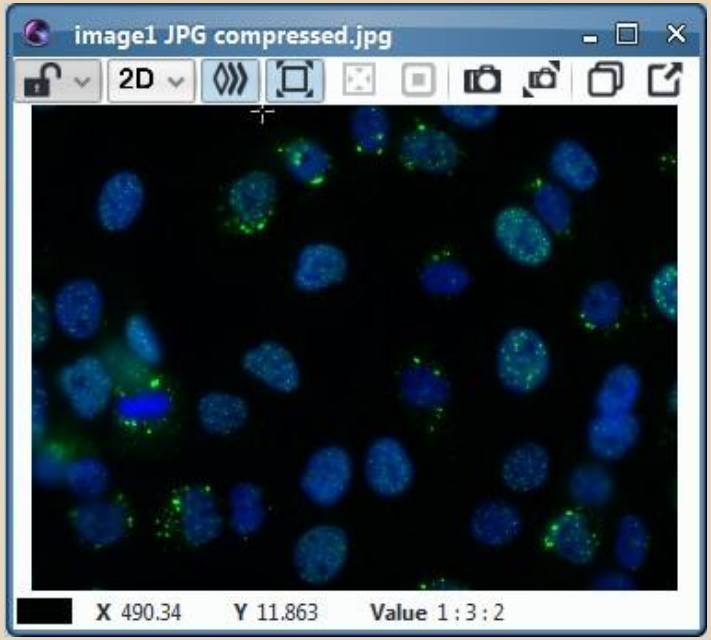
Almost done...

```
sequence.addOverlay(new SimpleCrossOverlay());
```

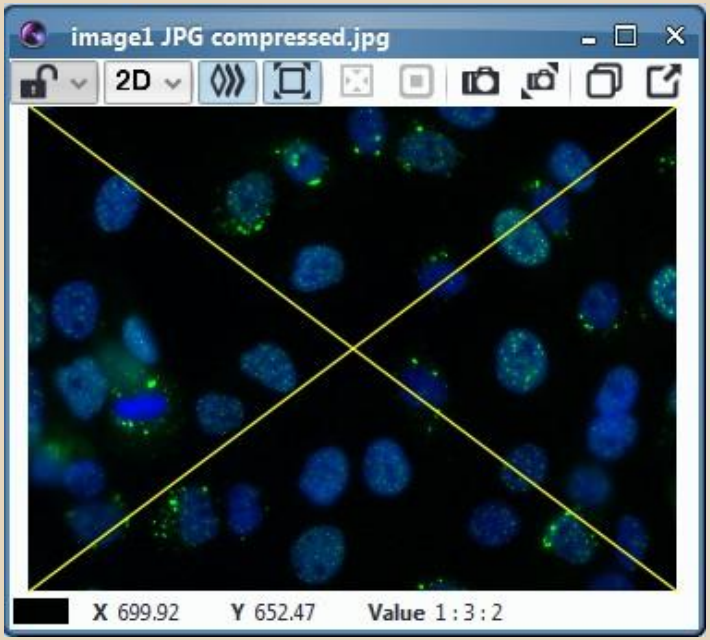
Add a new `SimpleCrossOverlay` to the sequence

Overlay

Before



After



Overlay - Anchor2D & tools

Anchor2D class

Offer simple object mouse manipulation as selection and drag operation

ImageOverlay class

Simple overlay to display an image

VtkPainter interface

Easier VTK overlay implementation

GUI - foreword

Creating a user interface can be a pain (especially in Java)

But in 90% of cases they look the same:

- parameters (name, value)
- buttons here, labels there, etc.

Our goal: provide an API that is

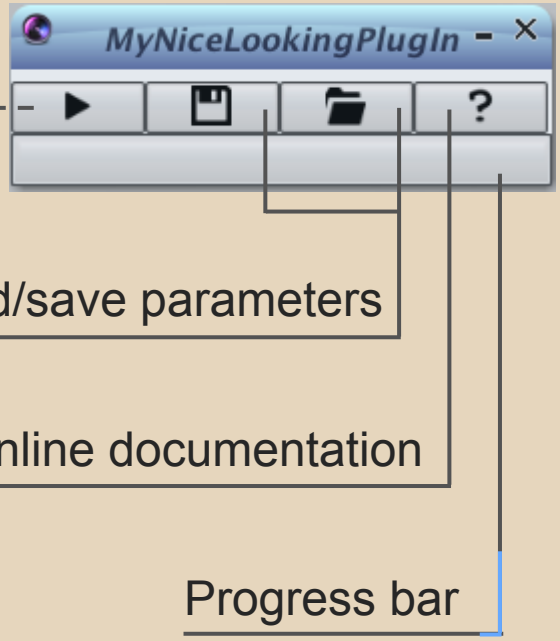
- simple (your plugin in less than 5 min.)
- standardized (users don’t get lost)
- powerful (bells & whistles included!)

EzPlug - basics

```
public class MyNiceLookingPlugIn extends EzPlug
{
     @Override
     public void initialize()
     {
          // create the interface here
     }

     @Override
     public void execute()
     {
          // do some “real” work
     }

     @Override
     public void clean()
     {
          // clean things when closing
     }
}
```

The interface is generated automatically

No more hardcore Swing!

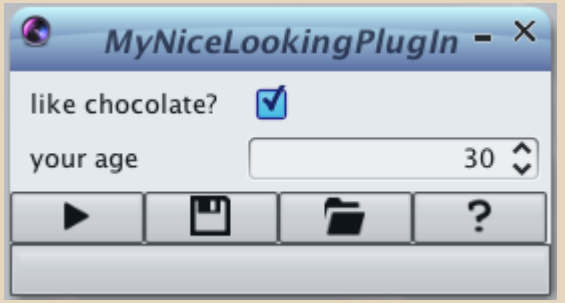


EzPlug - GUI - parameters

```
public class MyNiceLookingPlugIn extends EzPlug
{
     EzVarInteger age =
     new EzVarInteger("your age", 30, 10, 100, 1);
                              name, default, min, max, step

     EzVarBoolean yummy =
     new EzVarBoolean("like chocolate?", true);
                              name,          default

    @Override
    public void initialize()
    {
        // add elements in order of appearance
        addEzComponent(yummy);
        addEzComponent(age);
    }
    …
}
```

Each graphical component adjusts to its contents

The user can't get it wrong!

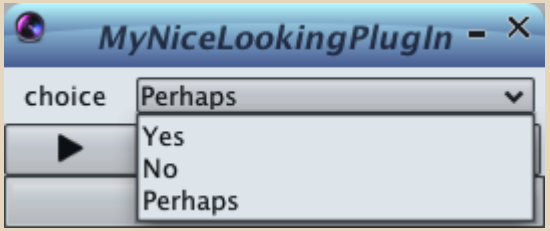


EzPlug - GUI - parameters

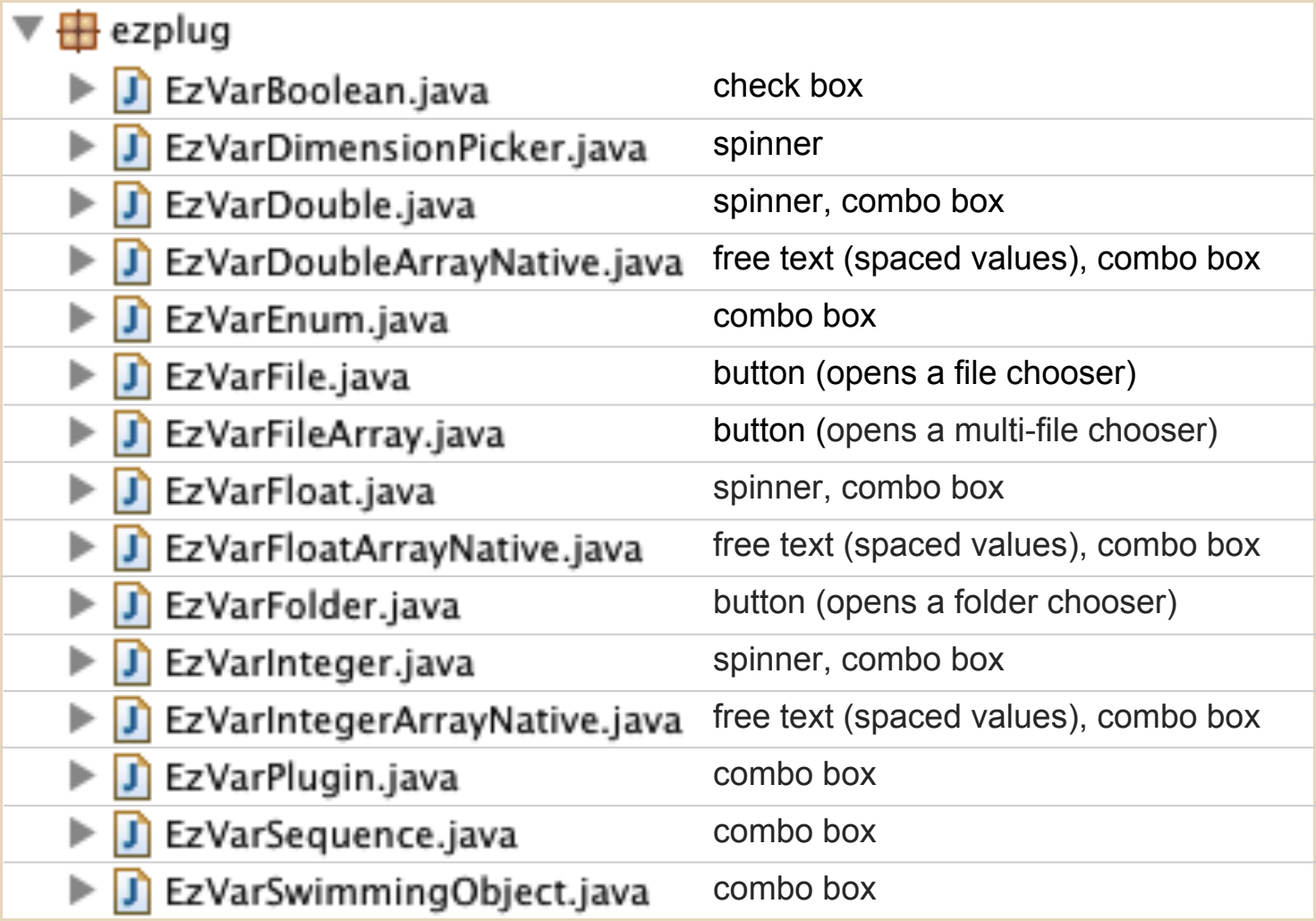
```
public class MyNiceLookingPlugIn extends EzPlug
{
     enum Choice {
          Yes, No, Perhaps
     }

     EzVarEnum<Choice> choice =
     new EzVarEnum<Choice>("choice",          name
                           Choice.values(),   valid values
                           Choice.Perhaps);   default

    @Override
    public void initialize()
    {
        addEzComponent(choice);
    }
    …
}
```

Each graphical component adjusts to its contents

The user can't get it wrong!

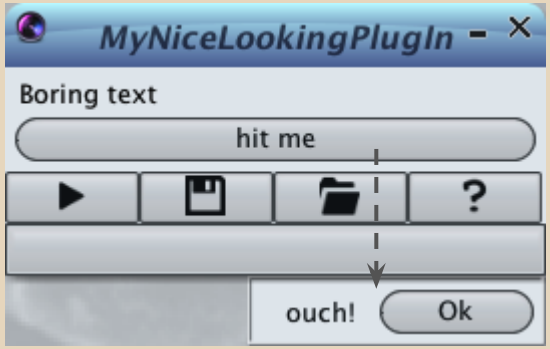


EzPlug - GUI - parameters



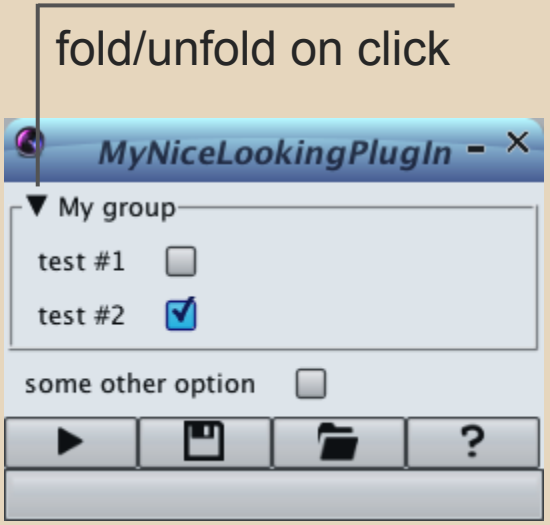
EzPlug - GUI - buttons & labels

```
public class MyNiceLookingPlugIn extends EzPlug
{
     ActionListener action = new ActionListener()
     {
          @Override
          public void actionPerformed(ActionEvent e)
          {
               new AnnounceFrame("ouch!");
          }
     };

     EzButton tap = new EzButton("hit me", action);

     @Override
     public void initialize()
     {
          addEzComponent(new EzLabel("Boring text"));
          addEzComponent(tap);
     }
     …
}
```



EzPlug - GUI - groups

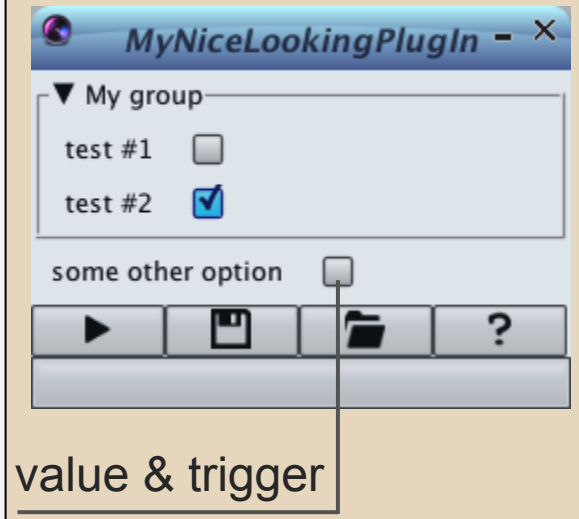
```
public class MyNiceLookingPlugIn extends EzPlug
{
     EzVarBoolean b1 =
     new EzVarBoolean(“test #1”, false);

     EzVarBoolean b2 =
     new EzVarBoolean(“test #2”, true);

     EzVarBoolean b3 =
     new EzVarBoolean(“some other option”, false);

     EzGroup grp = new EzGroup(“My group”, b1, b2);

     @Override
     public void initialize()
     {
          addEzComponent(grp);
          addEzComponent(b3);
     }
     …
}
```



EzPlug - GUI - hiding/showing

```
public class MyNiceLookingPlugIn extends EzPlug
{
     EzVarBoolean b1 =
     new EzVarBoolean("test #1", false);

     EzVarBoolean b2 =
     new EzVarBoolean("test #2", true);

     EzVarBoolean b3 =
     new EzVarBoolean("some other option", false);

     EzGroup grp = new EzGroup("My group", b1, b2);

     @Override
     public void initialize()
     {
          addEzComponent(grp);
          addEzComponent(b3);
          b3.addVisibilityTriggerTo(grp, false);
     }
     ...
}
```

"show {grp} only if {b3} has value {false}"



EzPlug - stop a running process

```
public class MyNiceLookingPlugIn extends EzPlug
     implements EzStoppable
{
     @Override
     public void execute()
     {
          // who am i?!
          Thread t = Thread.currentThread();

          for(int i=1; i<=100; i++)
          {
               // do stuff
               if (t.isInterrupted()) break;
          }
     }
}
```

The easy way...

EzPlug - stop a running process

```
public class MyNiceLookingPlugIn extends EzPlug
     implements EzStoppable
{
     boolean stopFlag;

     @Override
     public void stopExecution()
     {
          stopFlag = true;
     }

     @Override
     public void execute()
     {
          stopFlag = false; // don't, stop me noooow!
          for(int i=1; i<=100; i++)
          {
               // do stuff
               if (stopFlag) break;
          }
     }
}
```

For more control...

EzPlug - going further

Progress bar / main buttons can be hidden

Supports “regular” AWT/Swing components

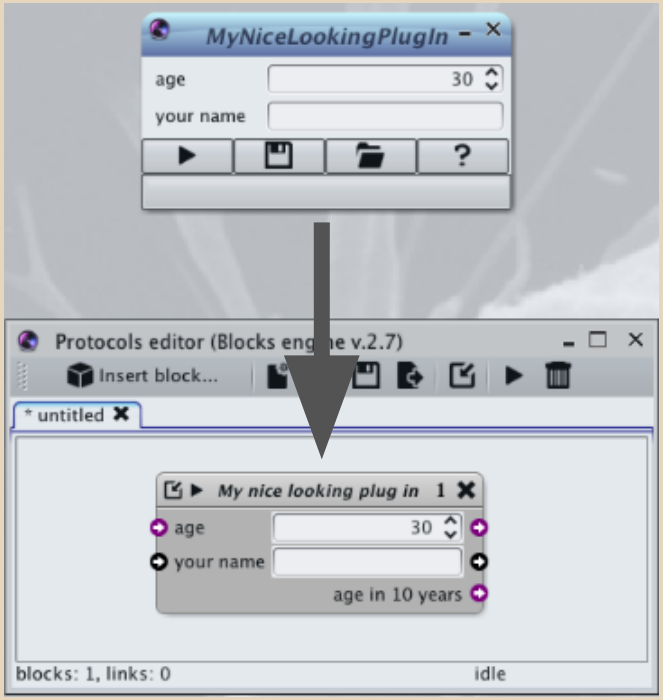
Available as modal dialogs (use EzDialog)

Open to new parameter types (extend EzVar)

EzPlug - going beyond

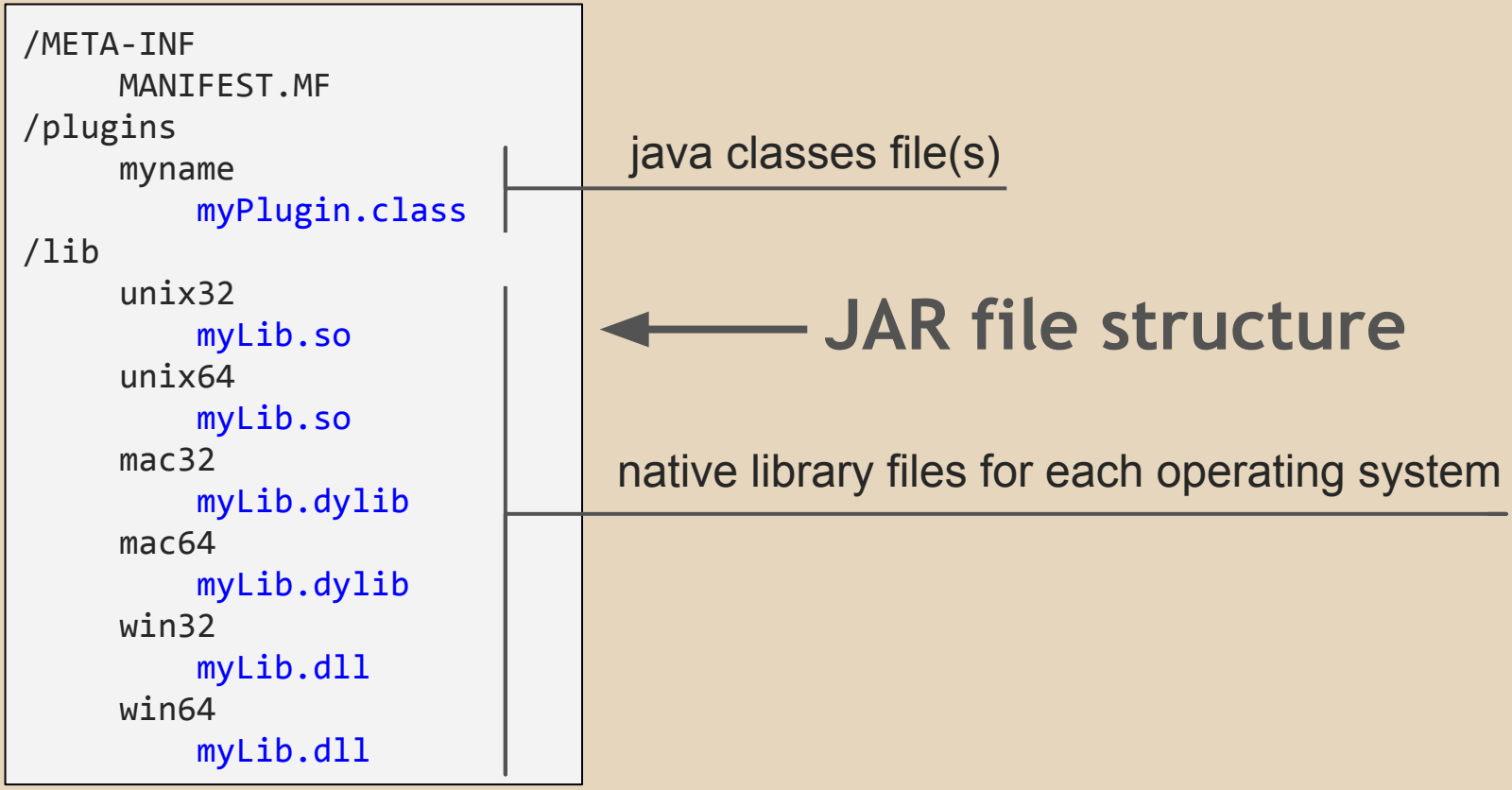
Compatible with block programming

```
public class MyNiceLookingPlugIn extends EzPlug
     implements Block
{
     EzVarInteger age = …
     EzVarText text = …
     // no GUI for this one
     VarInteger out = new VarInteger(“out”, 0);

     … // fill the other methods

     public void declareInput(VarList inputMap)
     {
          inputMap.add(age.getVariable());
          inputMap.add(text.getVariable());
     }

     public void declareOutput(VarList outputMap)
     {
          outputMap.add(out);
     }
}
```



Extra - Native library support

Easy integration of native library directly in the plugin jar file



Extra - Native library support

Load the native library through the plugin

```
public class NativeLibraryTestPlugin extends PluginActionable
{
     public NativeLibraryTestPlugin ()
     {
          super();

          loadLibrary("myLib");
     }

@Override
     public void run()
     {
          myLib.executeNativeMethod(..);
     }
}
```

Load the native library from the plugin JAR file

Executive the library native method