Processing

Interactive video playback and capture in the Processing Language via gstreamer

Andrés Colubri

GStreamer Conference 2015
Dublin, Ireland

Overview of the talk

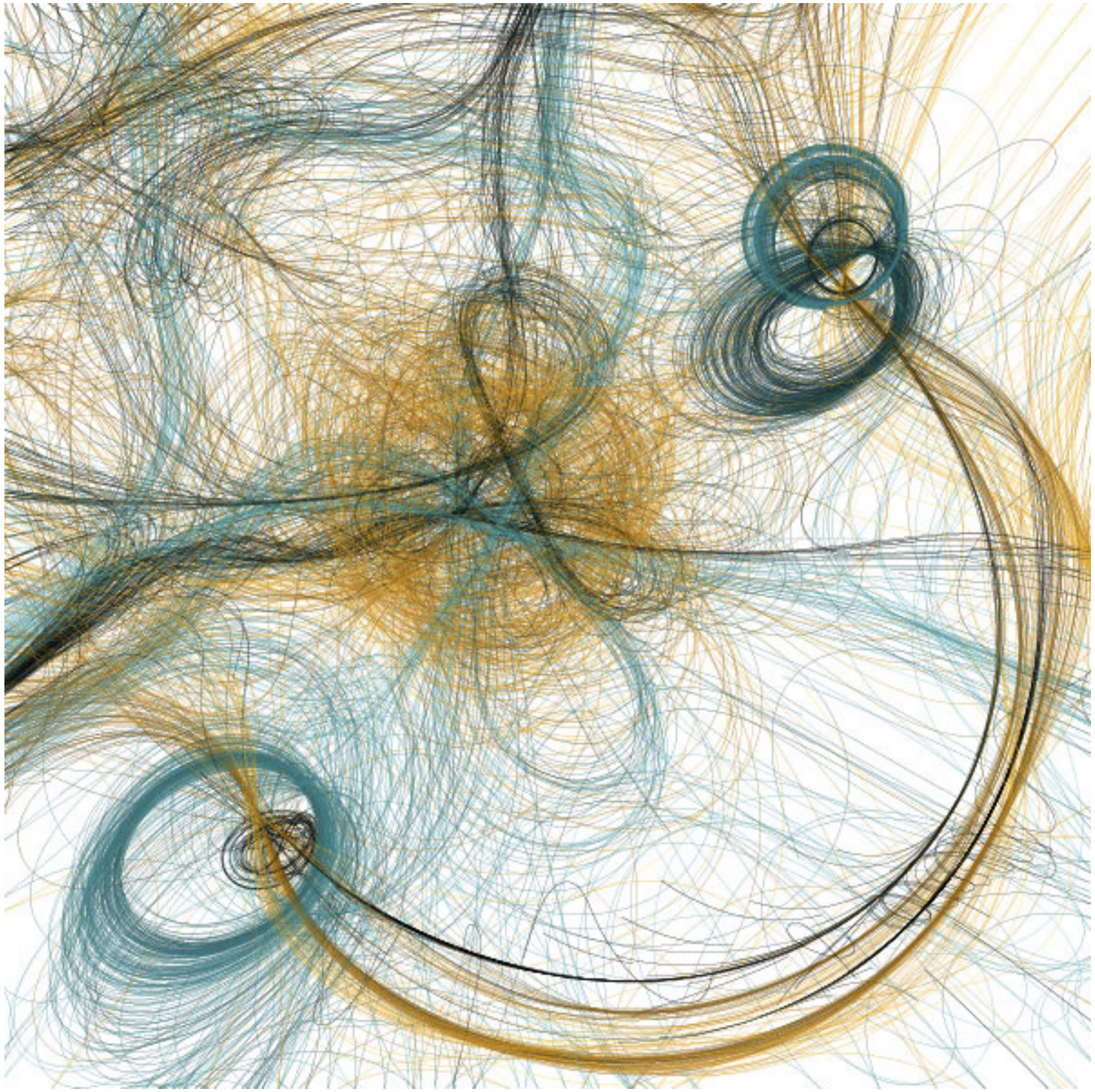
1. Quick intro to Processing
2. The video library (and gstreamer-java)
3. The future (is now): moving to GStreamer 1.0

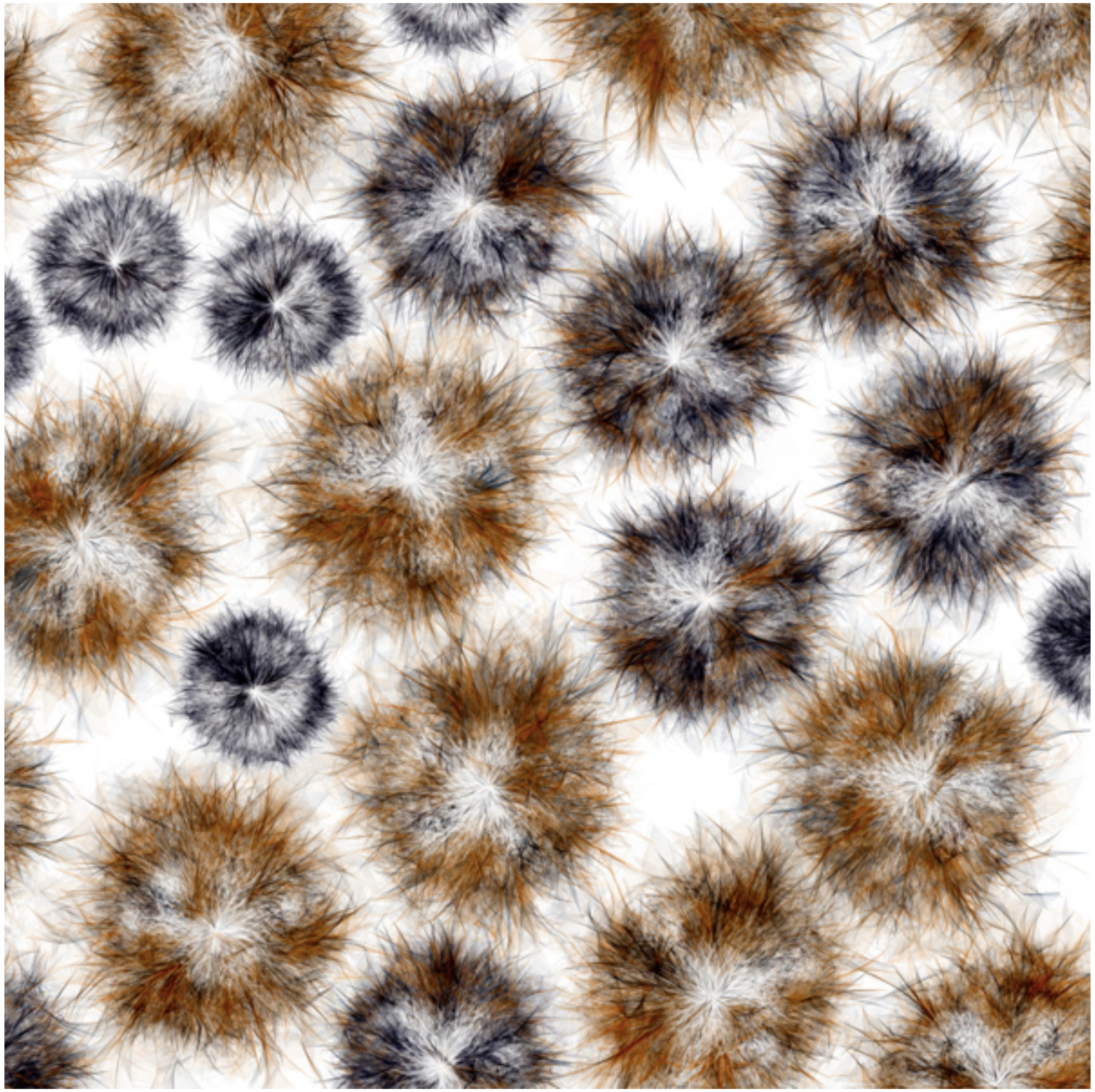
What is Processing?

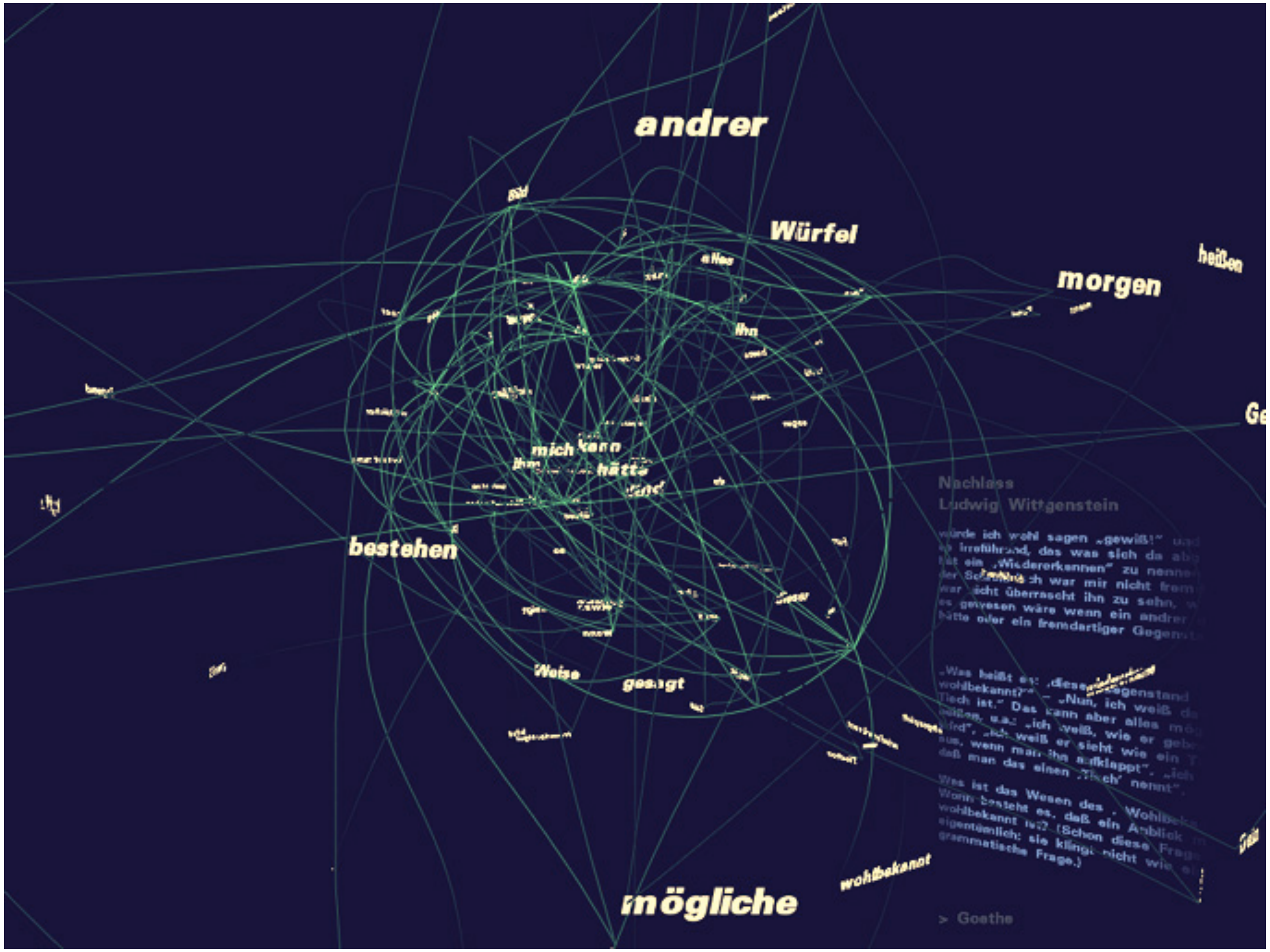
Processing is a programming language and software sketchbook for learning how to code within the context of the visual arts

- It was initiated by Ben Fry and Casey Reas in 2001 at the MIT Media Lab
- Their goal was to create a programming tool to make code more accessible to artists and designers
- It's built around the idea of "code sketching" by implementing visual ideas and iterating quickly
- It is based on Java to reach a compromise between language simplicity, performance, and portability
- http://www.processing.org/ for more details!

Hello, world! in Processing


Path 00. Casey Reas (2001)


Process 6. Casey Reas (2005)



Nachlass
Ludwig Wittgenstein

würde ich wohl sagen „gewiß!" und
[illegible] irreführend, das was sich da [illegible]
[illegible] ein „Wiedererkennen" zu nennen
der [illegible] ch war mir nicht frem
war nicht überrascht ihn zu sehn, [illegible]
es gewesen wäre wenn ein andrer [illegible]
hätte oder ein fremdartiger Gegen[illegible]

„Was heißt es: ‚dieser Gegenstand
wohlbekannt?'" — „Nun, ich weiß, [illegible]
Tisch ist." Das kann aber alles mö[illegible]
heißen, u.a.: „ich weiß, wie er gebr[illegible]
wird", „ich weiß er sieht wie ein T[illegible]
aus, wenn man ihn aufklappt", „ich
daß man das einen ‚Tisch' nennt".

Was ist das Wesen des „Wohlbek[illegible]
Worin besteht es, daß ein Anblick [illegible]
wohlbekannt ist? (Schon diese Frage
eigentümlich: sie klingt nicht wie ei[illegible]
grammatische Frage.)

> Goethe

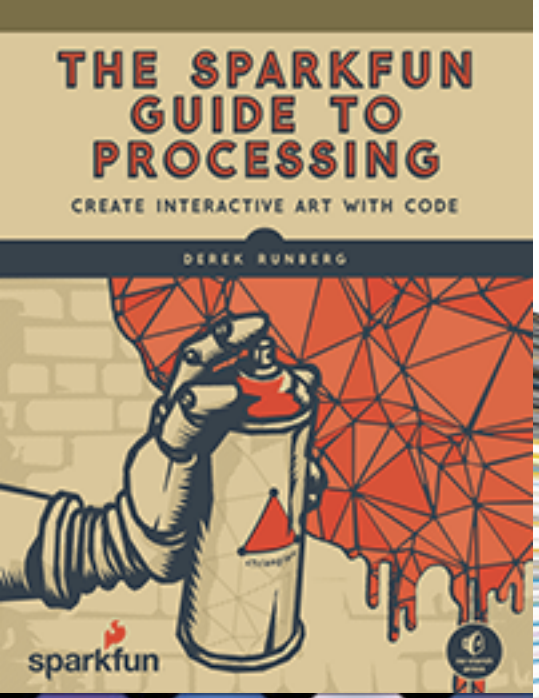
Valence. Ben Fry (1999-2002)

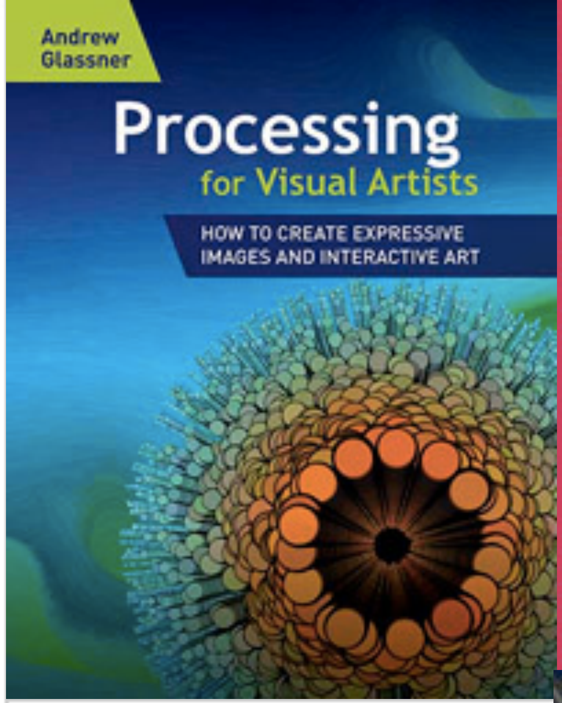
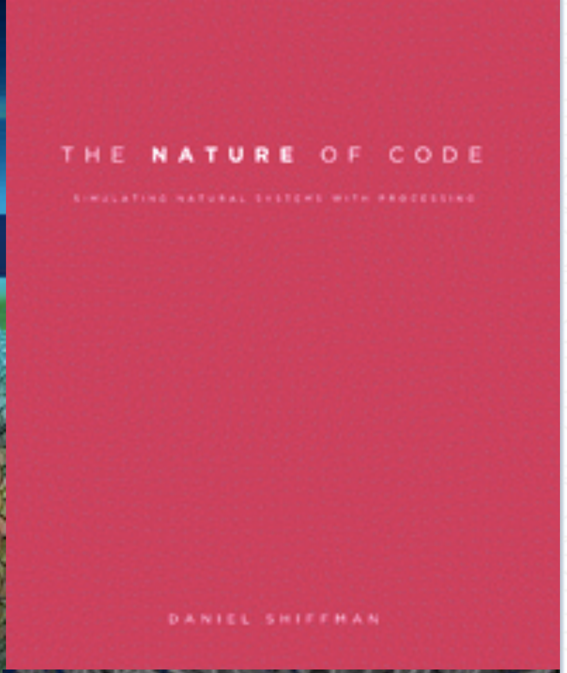
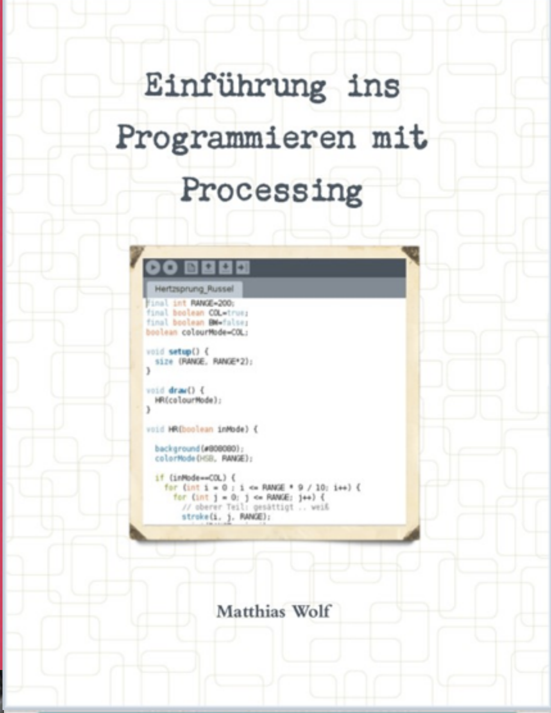
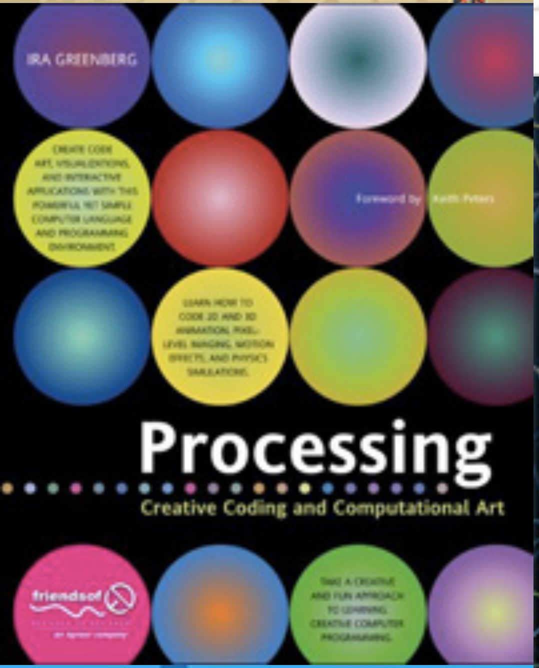
Isometric Blocks. Ben Fry (2002-2004)

unnamed soundsculpture (2013)
https://vimeo.com/38840688

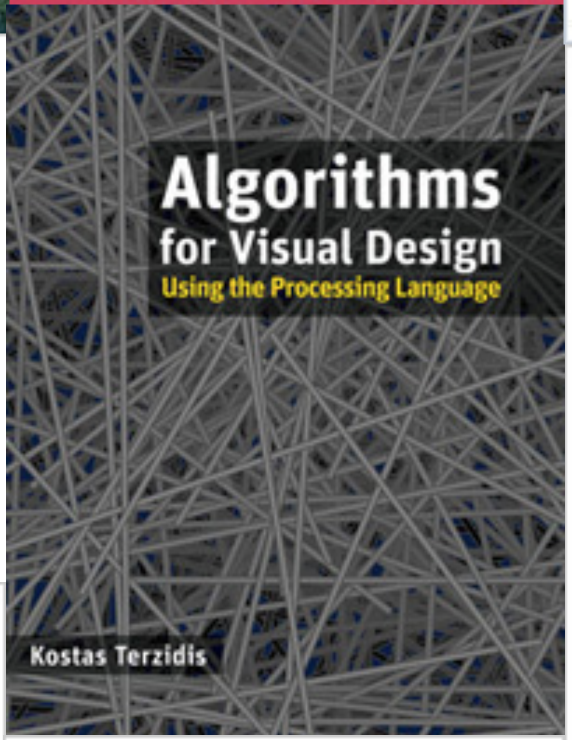
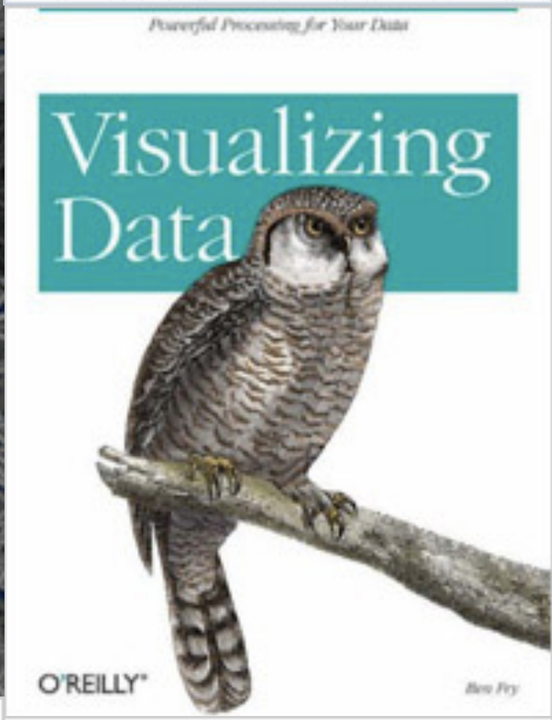
Community

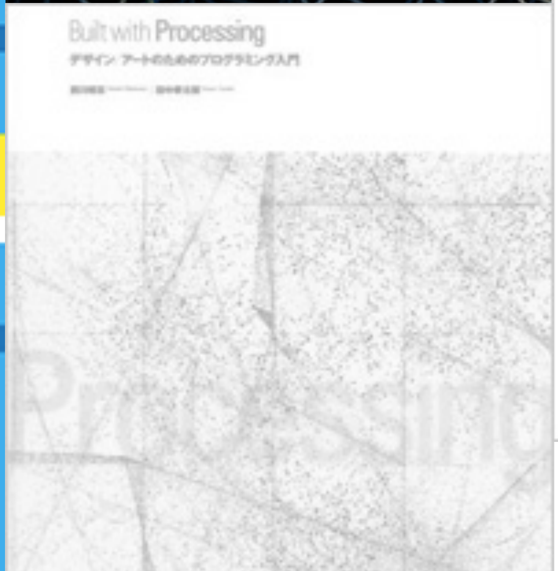
- Thousands of people use the software every day for teaching and production
- http://forum.processing.org/
- https://github.com/processing
- http://p5js.org/
- http://openprocessing.org/
- http://fyprocessing.tumblr.com/
- Related projects: Arduino, openFrameworks, …

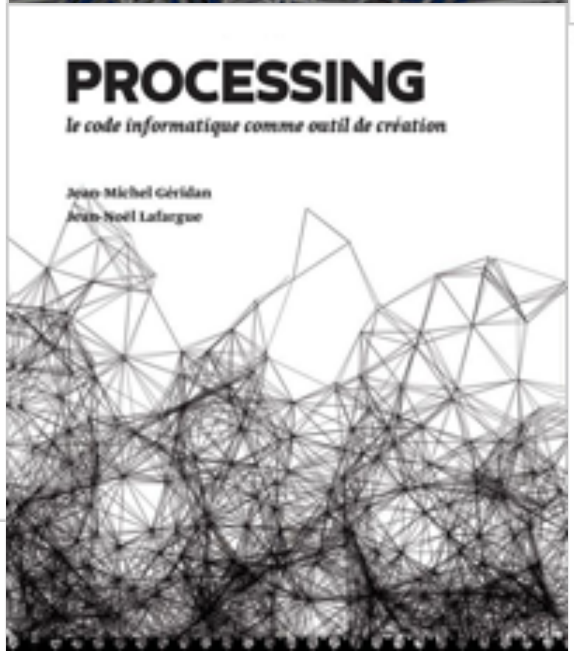

THE SPARKFUN GUIDE TO PROCESSING
CREATE INTERACTIVE ART WITH CODE
DEREK RUNBERG
sparkfun

GENERATIVE DESIGN
Visualize, Program, and Create with Processing

Andrew Glassner
Processing for Visual Artists
HOW TO CREATE EXPRESSIVE IMAGES AND INTERACTIVE ART

THE NATURE OF CODE
SIMULATING NATURAL SYSTEMS WITH PROCESSING
DANIEL SHIFFMAN

Einführung ins Programmieren mit Processing
Matthias Wolf

IRA GREENBERG
Foreword by Keith Peters
Processing
Creative Coding and Computational Art
friendsof

Second Edition
Processing
A Programming Handbook for Visual Designers and Artists
Casey Reas
Ben Fry

Generative Art
A practical guide using Processing
Matt Pearson
MANNING

Algorithms for Visual Design
Using the Processing Language
Kostas Terzidis

Powerful Processing for Your Data
Visualizing Data
O'REILLY
Ben Fry

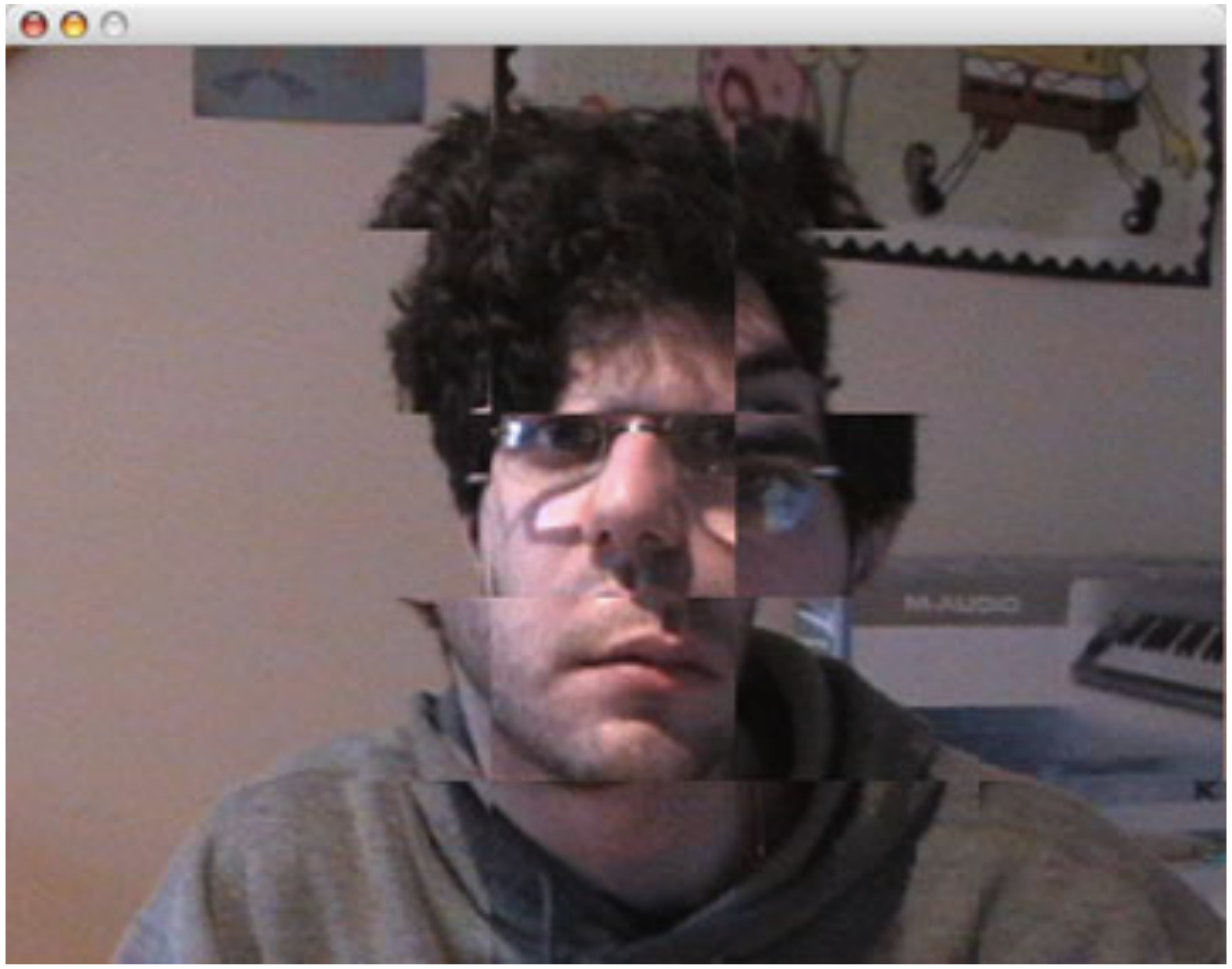
Second Edition
Make:
Getting Started with Processing
A Hands-On Introduction to Making Interactive Graphics

Built with Processing
Processing

Make: Books
Making Things See
Learn by Discovery
Greg Borenstein
O'REILLY
Make:

PROCESSING
le code informatique comme outil de création
Jean-Michel Géridan
Jean-Noël Lafargue

Bartmann
Processing

Practical methods for connecting physical objects.
2nd Edition
Making Things Talk
Learn by Discovery
O'REILLY
Make:

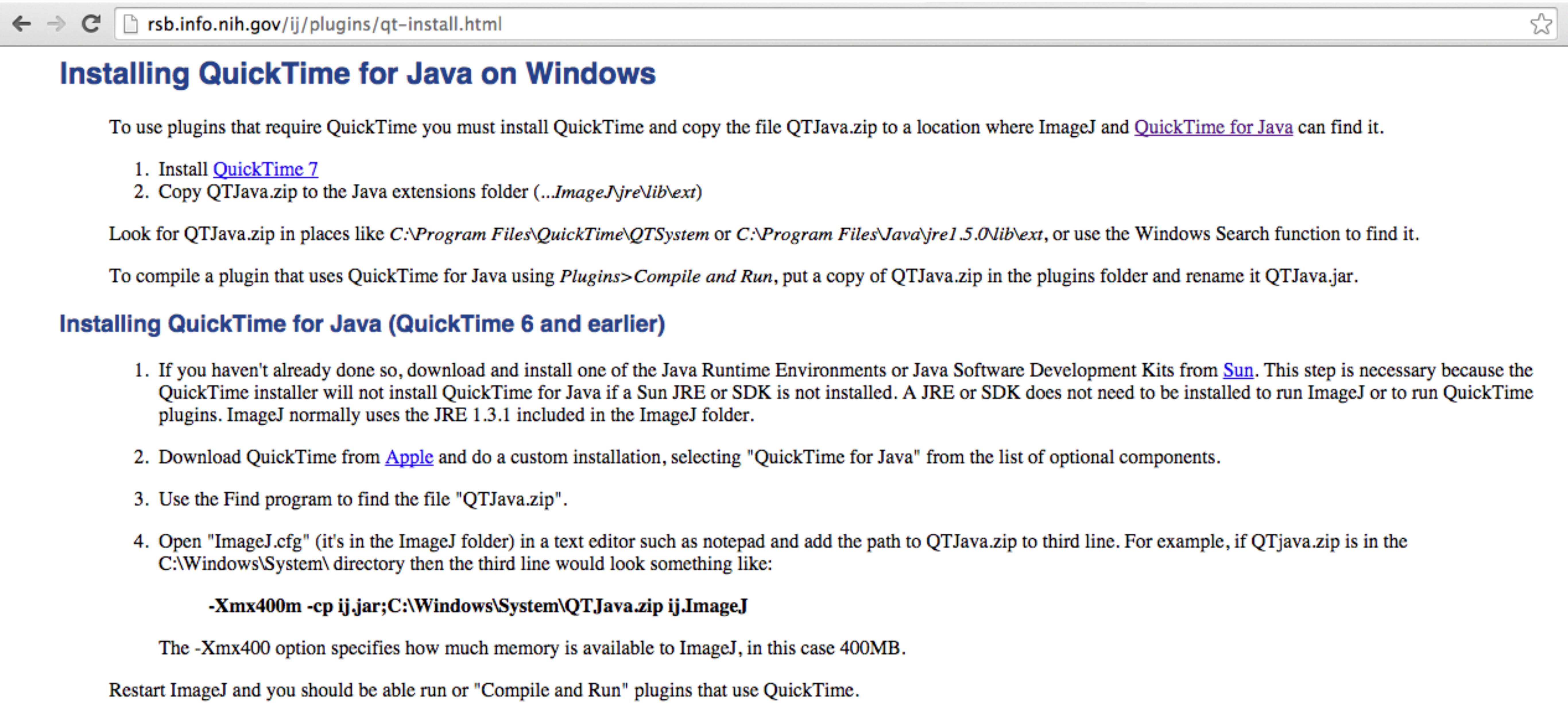
Video in Processing



(Image by Elie Zananiri)

- Processing has a modular architecture that makes it easy to extend its core functionality through libraries
- Since early versions (pre1.0), Processing included a video library
- This library follows the principles of the core API: immediate visual feedback, simple naming conventions, circumscribed functionality
- It was originally based on QuickTime for Java :-(

demo of the video library



rsb.info.nih.gov/ij/plugins/qt-install.html

Installing QuickTime for Java on Windows

To use plugins that require QuickTime you must install QuickTime and copy the file QTJava.zip to a location where ImageJ and QuickTime for Java can find it.

1. Install QuickTime 7
2. Copy QTJava.zip to the Java extensions folder (*...ImageJ\jre\lib\ext*)

Look for QTJava.zip in places like *C:\Program Files\QuickTime\QTSystem* or *C:\Program Files\Java\jre1.5.0\lib\ext*, or use the Windows Search function to find it.

To compile a plugin that uses QuickTime for Java using *Plugins>Compile and Run*, put a copy of QTJava.zip in the plugins folder and rename it QTJava.jar.

Installing QuickTime for Java (QuickTime 6 and earlier)

1. If you haven't already done so, download and install one of the Java Runtime Environments or Java Software Development Kits from Sun. This step is necessary because the QuickTime installer will not install QuickTime for Java if a Sun JRE or SDK is not installed. A JRE or SDK does not need to be installed to run ImageJ or to run QuickTime plugins. ImageJ normally uses the JRE 1.3.1 included in the ImageJ folder.

2. Download QuickTime from Apple and do a custom installation, selecting "QuickTime for Java" from the list of optional components.

3. Use the Find program to find the file "QTJava.zip".

4. Open "ImageJ.cfg" (it's in the ImageJ folder) in a text editor such as notepad and add the path to QTJava.zip to third line. For example, if QTjava.zip is in the C:\Windows\System\ directory then the third line would look something like:

   ```
   -Xmx400m -cp ij.jar;C:\Windows\System\QTJava.zip ij.ImageJ
   ```

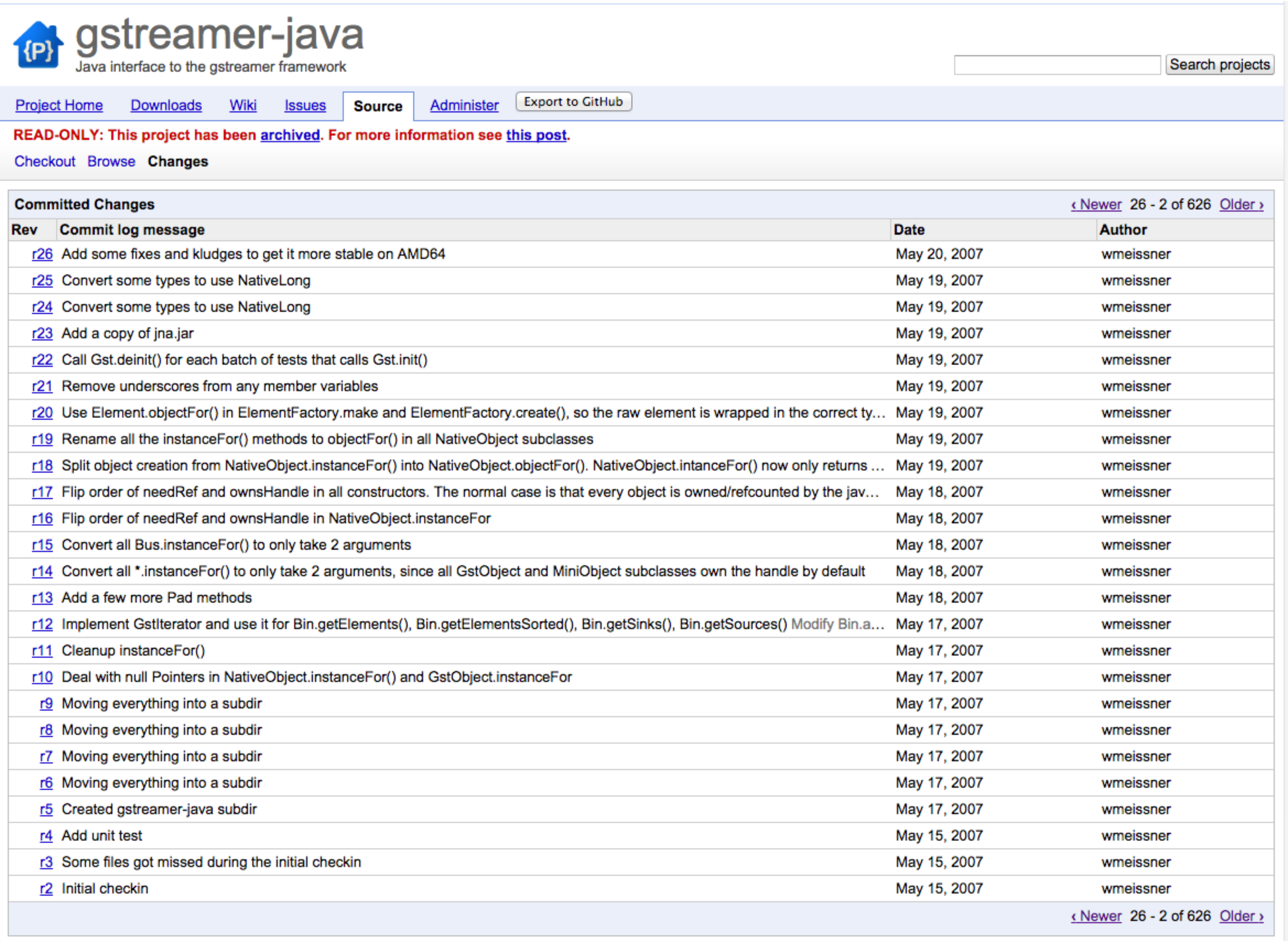
   The -Xmx400 option specifies how much memory is available to ImageJ, in this case 400MB.

Restart ImageJ and you should be able run or "Compile and Run" plugins that use QuickTime.

Already in 2007, QuickTime for Java had many disadvantages:

- It was not cross-platform, as a consequence the video library did not work on Linux
- It was very slow: pixel buffers were copied from QuickTime over to Java
- You needed to install QuickTime on Windows

Enter gstreamer-java (2007 - present)!



gstreamer-java

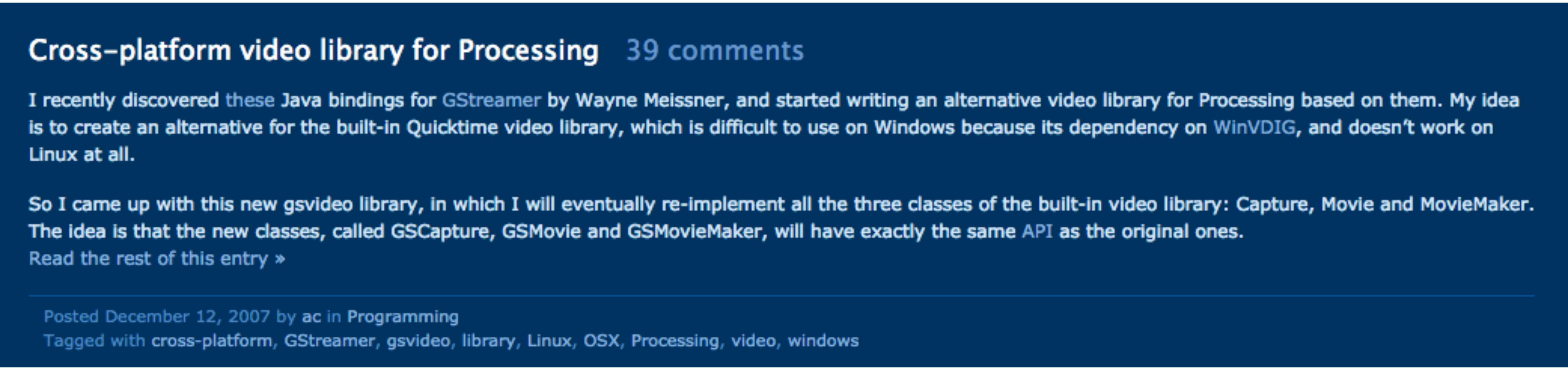
Java interface to the gstreamer framework

READ-ONLY: This project has been archived. For more information see this post.

Committed Changes ‹ Newer 26 - 2 of 626 Older ›

Rev	Commit log message	Date	Author
r26	Add some fixes and kludges to get it more stable on AMD64	May 20, 2007	wmeissner
r25	Convert some types to use NativeLong	May 19, 2007	wmeissner
r24	Convert some types to use NativeLong	May 19, 2007	wmeissner
r23	Add a copy of jna.jar	May 19, 2007	wmeissner
r22	Call Gst.deinit() for each batch of tests that calls Gst.init()	May 19, 2007	wmeissner
r21	Remove underscores from any member variables	May 19, 2007	wmeissner
r20	Use Element.objectFor() in ElementFactory.make and ElementFactory.create(), so the raw element is wrapped in the correct ty…	May 19, 2007	wmeissner
r19	Rename all the instanceFor() methods to objectFor() in all NativeObject subclasses	May 19, 2007	wmeissner
r18	Split object creation from NativeObject.instanceFor() into NativeObject.objectFor(). NativeObject.intanceFor() now only returns …	May 19, 2007	wmeissner
r17	Flip order of needRef and ownsHandle in all constructors. The normal case is that every object is owned/refcounted by the jav…	May 18, 2007	wmeissner
r16	Flip order of needRef and ownsHandle in NativeObject.instanceFor	May 18, 2007	wmeissner
r15	Convert all Bus.instanceFor() to only take 2 arguments	May 18, 2007	wmeissner
r14	Convert all *.instanceFor() to only take 2 arguments, since all GstObject and MiniObject subclasses own the handle by default	May 18, 2007	wmeissner
r13	Add a few more Pad methods	May 18, 2007	wmeissner
r12	Implement GstIterator and use it for Bin.getElements(), Bin.getElementsSorted(), Bin.getSinks(), Bin.getSources() Modify Bin.a…	May 17, 2007	wmeissner
r11	Cleanup instanceFor()	May 17, 2007	wmeissner
r10	Deal with null Pointers in NativeObject.instanceFor() and GstObject.instanceFor	May 17, 2007	wmeissner
r9	Moving everything into a subdir	May 17, 2007	wmeissner
r8	Moving everything into a subdir	May 17, 2007	wmeissner
r7	Moving everything into a subdir	May 17, 2007	wmeissner
r6	Moving everything into a subdir	May 17, 2007	wmeissner
r5	Created gstreamer-java subdir	May 17, 2007	wmeissner
r4	Add unit test	May 15, 2007	wmeissner
r3	Some files got missed during the initial checkin	May 15, 2007	wmeissner
r2	Initial checkin	May 15, 2007	wmeissner

‹ Newer 26 - 2 of 626 Older ›



Cross-platform video library for Processing 39 comments

I recently discovered these Java bindings for GStreamer by Wayne Meissner, and started writing an alternative video library for Processing based on them. My idea is to create an alternative for the built-in Quicktime video library, which is difficult to use on Windows because its dependency on WinVDIG, and doesn't work on Linux at all.

So I came up with this new gsvideo library, in which I will eventually re-implement all the three classes of the built-in video library: Capture, Movie and MovieMaker. The idea is that the new classes, called GSCapture, GSMovie and GSMovieMaker, will have exactly the same API as the original ones.
Read the rest of this entry »

Posted December 12, 2007 by ac in Programming
Tagged with cross-platform, GStreamer, gsvideo, library, Linux, OSX, Processing, video, windows

- gstreamer-java is a set of Java bindings for GStreamer 0.10
- It is based on Java Native Access (JNA)
- It was initiated by Wayne Meissner in early 2007
- Levente Farkas, Tal Shalif, and David Hoyt made important contributions early on and maintained the project after Wayne's departure
- I added a few components that allowed to extract the video frames into the Java application, and re-wrote the video library in Processing to use gstreamer-java instead of QuickTime for Java

a few projects using video:

Videorative Portraits: https://vimeo.com/32760578

The Life in a Day Gallery: https://www.youtube.com/watch?v=g4y6cppFxgo

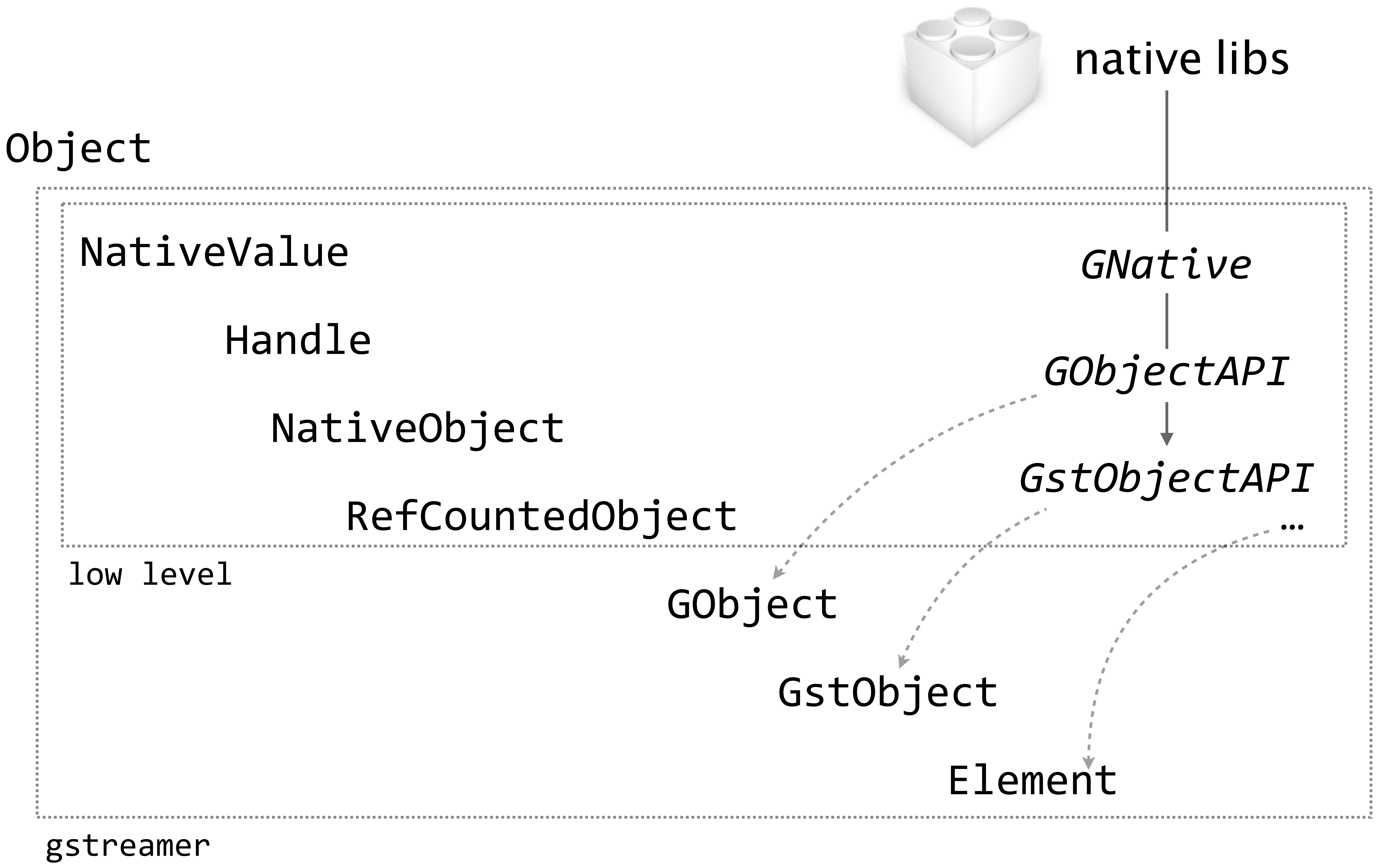
Mass Rhythm: https://vimeo.com/105419121

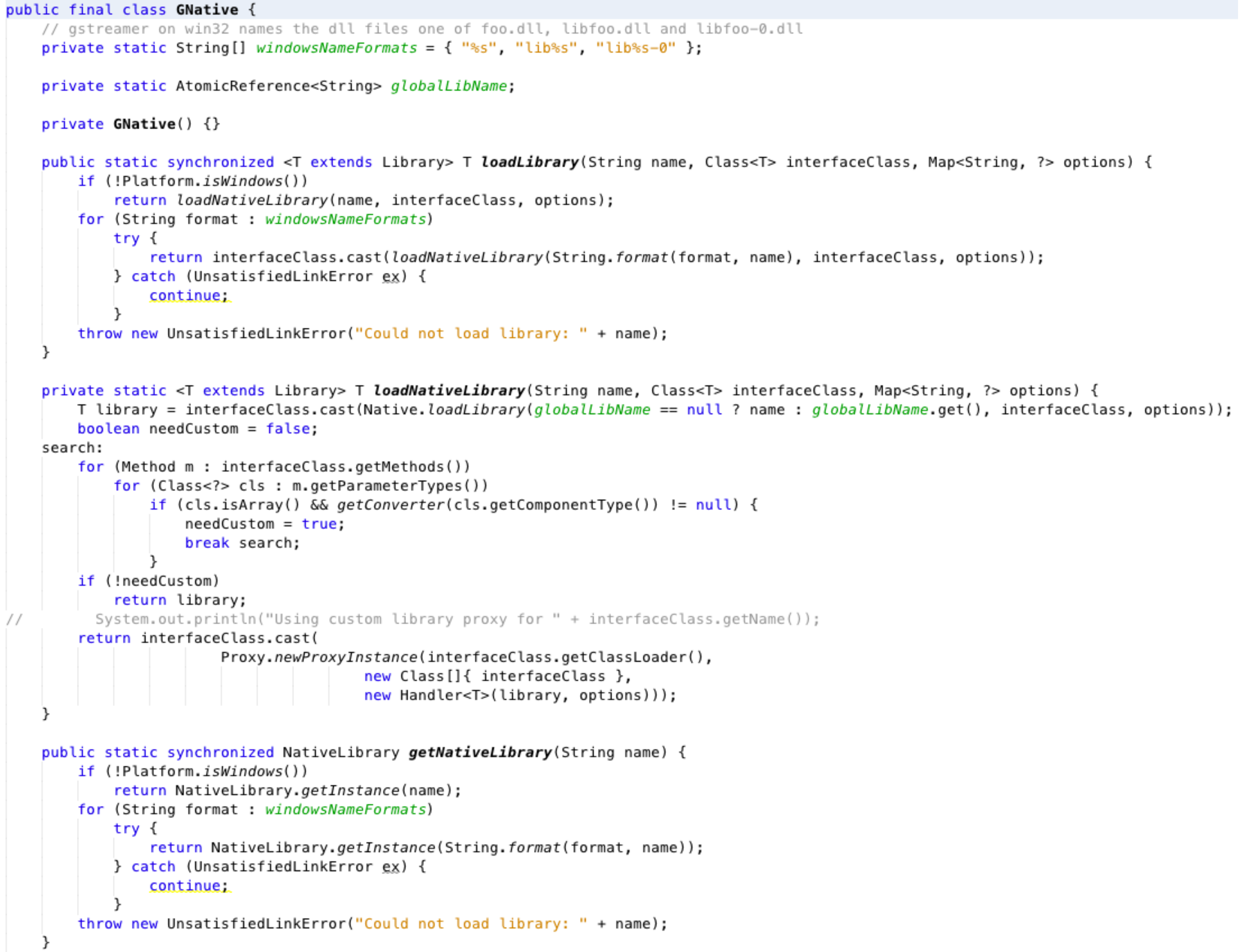
Facial Distortions: https://vimeo.com/3688931

Incarceration-vacation: https://vimeo.com/34311454

Let’s take a deeper look into gstreamer-java

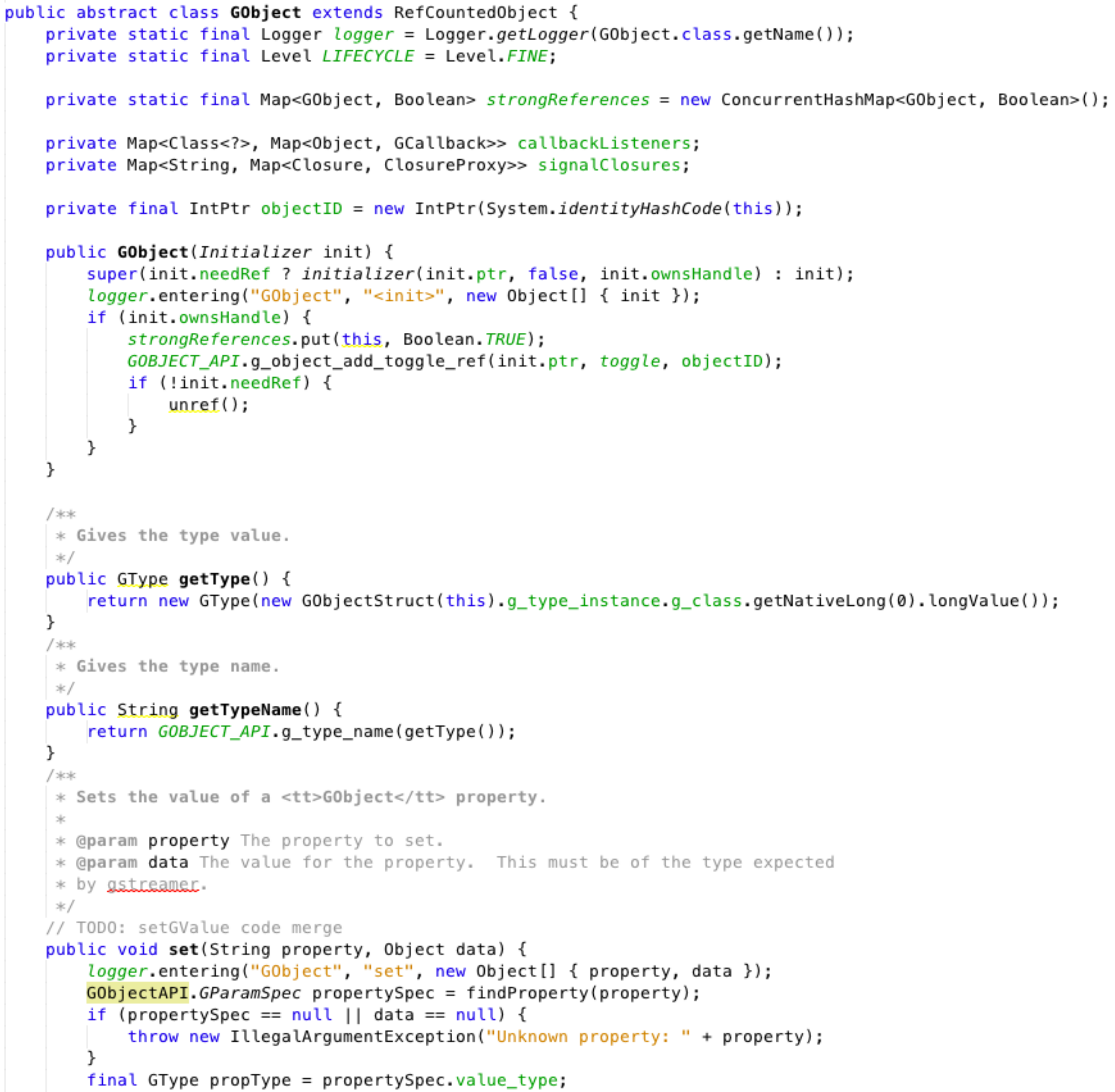
- It uses JNA, so no need to write glue code in C, like it is the case with JNI
- However, the object hierarchy in GStreamer, starting from GLib, is duplicated manually in Java
- It comprises of around 300 Java classes.


native libs
Object
NativeValue
Handle
NativeObject
RefCountedObject
low level
GNative
GObjectAPI
GstObjectAPI
…
GObject
GstObject
Element
gstreamer



```java
public final class GNative {
    // gstreamer on win32 names the dll files one of foo.dll, libfoo.dll and libfoo-0.dll
    private static String[] windowsNameFormats = { "%s", "lib%s", "lib%s-0" };

    private static AtomicReference<String> globalLibName;

    private GNative() {}

    public static synchronized <T extends Library> T loadLibrary(String name, Class<T> interfaceClass, Map<String, ?> options) {
        if (!Platform.isWindows())
            return loadNativeLibrary(name, interfaceClass, options);
        for (String format : windowsNameFormats)
            try {
                return interfaceClass.cast(loadNativeLibrary(String.format(format, name), interfaceClass, options));
            } catch (UnsatisfiedLinkError ex) {
                continue;
            }
        throw new UnsatisfiedLinkError("Could not load library: " + name);
    }

    private static <T extends Library> T loadNativeLibrary(String name, Class<T> interfaceClass, Map<String, ?> options) {
        T library = interfaceClass.cast(Native.loadLibrary(globalLibName == null ? name : globalLibName.get(), interfaceClass, options));
        boolean needCustom = false;
    search:
        for (Method m : interfaceClass.getMethods())
            for (Class<?> cls : m.getParameterTypes())
                if (cls.isArray() && getConverter(cls.getComponentType()) != null) {
                    needCustom = true;
                    break search;
                }
        if (!needCustom)
            return library;
//          System.out.println("Using custom library proxy for " + interfaceClass.getName());
        return interfaceClass.cast(
                        Proxy.newProxyInstance(interfaceClass.getClassLoader(),
                                        new Class[]{ interfaceClass },
                                        new Handler<T>(library, options)));
    }

    public static synchronized NativeLibrary getNativeLibrary(String name) {
        if (!Platform.isWindows())
            return NativeLibrary.getInstance(name);
        for (String format : windowsNameFormats)
            try {
                return NativeLibrary.getInstance(String.format(format, name));
            } catch (UnsatisfiedLinkError ex) {
                continue;
            }
        throw new UnsatisfiedLinkError("Could not load library: " + name);
    }
```

```java
public interface GObjectAPI extends Library {
    GObjectAPI GOBJECT_API = GNative.loadLibrary("gobject-2.0", GObjectAPI.class,
        new HashMap<String, Object>() {{
            put(Library.OPTION_TYPE_MAPPER, new GTypeMapper());
        }});

    GType g_object_get_type();
    void g_param_value_validate(GParamSpec spec, GValue data);
    void g_object_set_property(GObject obj, String property, GValue data);
    void g_object_get_property(GObject obj, String property, GValue data);
    void g_object_set(GObject obj, String propertyName, Object... data);
    void g_object_get(GObject obj, String propertyName, Object... data);
    Pointer g_object_class_list_properties(Pointer oclass, IntByReference size);

    Pointer g_object_new(GType object_type, Object... args);

    interface GClosureNotify extends Callback {
        void callback(Pointer data, Pointer closure);
    }
    NativeLong g_signal_connect_data(GObject obj, String signal, Callback callback, Pointer data,
            GClosureNotify destroy_data, int connect_flags);
    void g_signal_handler_disconnect(GObject obj, NativeLong id);
    boolean g_object_is_floating(GObject obj);
    interface GToggleNotify extends Callback {
        void callback(Pointer data, Pointer obj, boolean is_last_ref);
    }
    void g_object_add_toggle_ref(Pointer object, GToggleNotify notify, Pointer data);
    void g_object_remove_toggle_ref(Pointer object, GToggleNotify notify, Pointer data);
    void g_object_add_toggle_ref(Pointer object, GToggleNotify notify, IntPtr data);
    void g_object_remove_toggle_ref(Pointer object, GToggleNotify notify, IntPtr data);
    interface GWeakNotify extends Callback {
        void callback(IntPtr data, Pointer obj);
    }
    void g_object_weak_ref(GObject object, GWeakNotify notify, IntPtr data);
    void g_object_weak_unref(GObject object, GWeakNotify notify, IntPtr data);
    Pointer g_object_ref(GObject object);
    void g_object_unref(GObject object);

    GParamSpec g_object_class_find_property(GObjectClass oclass, String property_name);
    Pointer g_object_class_find_property(Pointer oclass, String property_name);
    GQuark g_quark_try_string(String string);
    GQuark g_quark_from_static_string(String string);
    GQuark g_quark_from_string(String string);
    String g_quark_to_string(GQuark quark);

    String g_intern_string(String string);
    String g_intern_static_string(String string);

    void g_type_init();
    void g_type_init_with_debug_flags(int flags);
```



```java
public abstract class GObject extends RefCountedObject {
    private static final Logger logger = Logger.getLogger(GObject.class.getName());
    private static final Level LIFECYCLE = Level.FINE;

    private static final Map<GObject, Boolean> strongReferences = new ConcurrentHashMap<GObject, Boolean>();

    private Map<Class<?>, Map<Object, GCallback>> callbackListeners;
    private Map<String, Map<Closure, ClosureProxy>> signalClosures;

    private final IntPtr objectID = new IntPtr(System.identityHashCode(this));

    public GObject(Initializer init) {
        super(init.needRef ? initializer(init.ptr, false, init.ownsHandle) : init);
        logger.entering("GObject", "<init>", new Object[] { init });
        if (init.ownsHandle) {
            strongReferences.put(this, Boolean.TRUE);
            GOBJECT_API.g_object_add_toggle_ref(init.ptr, toggle, objectID);
            if (!init.needRef) {
                unref();
            }
        }
    }

    /**
     * Gives the type value.
     */
    public GType getType() {
        return new GType(new GObjectStruct(this).g_type_instance.g_class.getNativeLong(0).longValue());
    }
    /**
     * Gives the type name.
     */
    public String getTypeName() {
        return GOBJECT_API.g_type_name(getType());
    }
    /**
     * Sets the value of a <tt>GObject</tt> property.
     *
     * @param property The property to set.
     * @param data The value for the property.  This must be of the type expected
     * by gstreamer.
     */
    // TODO: setGValue code merge
    public void set(String property, Object data) {
        logger.entering("GObject", "set", new Object[] { property, data });
        GObjectAPI.GParamSpec propertySpec = findProperty(property);
        if (propertySpec == null || data == null) {
            throw new IllegalArgumentException("Unknown property: " + property);
        }
        final GType propType = propertySpec.value_type;
```

```java
public interface GstElementAPI extends com.sun.jna.Library {
    GstElementAPI GSTELEMENT_API = GstNative.load(GstElementAPI.class);

    GType gst_element_get_type();
    StateChangeReturn gst_element_set_state(Element elem, State state);
    StateChangeReturn gst_element_get_state(Element elem, State[] state, State[] pending, long timeout);
    StateChangeReturn gst_element_get_state(Element elem, State[] state, State[] pending, ClockTime timeout);
    boolean gst_element_set_locked_state(Element element, boolean locked_state);
    boolean gst_element_sync_state_with_parent(Element elem);
    boolean gst_element_query_position(Element elem, Format[] fmt, long[] pos);
    boolean gst_element_query_duration(Element elem, Format[] fmt, long[] pos);
    boolean gst_element_query(Element elem, Query query);
    boolean gst_element_seek(Element elem, double rate, Format format, int flags,
            SeekType cur_type, long cur, SeekType stop_type, long stop);
    boolean gst_element_seek_simple(Element elem, Format format, int flags, long pos);
    boolean gst_element_link(Element elem1, Element elem2);
    boolean gst_element_link_many(Element... elements);
    void gst_element_unlink_many(Element... elements);
    void gst_element_unlink(Element elem1, Element elem2);
    @CallerOwnsReturn Pad gst_element_get_pad(Element elem, String name);
    @CallerOwnsReturn Pad gst_element_get_static_pad(Element element, String name);
    // pads returned from get_request have to be freed via release_request_pad
    Pad gst_element_get_request_pad(Element element, String name);
    void gst_element_release_request_pad(Element element, Pad pad);
    boolean gst_element_add_pad(Element elem, Pad pad);
    boolean gst_element_remove_pad(Element elem, @IncRef Pad pad);
    boolean gst_element_link_pads(Element src, String srcpadname, Element dest, String destpadname);
    void gst_element_unlink_pads(Element src, String srcpadname, Element dest, String destpadname);
    boolean gst_element_link_pads_filtered(Element src, String srcpadname, Element dest, String destpadname,
            Caps filter);

    Pointer gst_element_iterate_pads(Element element);
    Pointer gst_element_iterate_src_pads(Element element);
    Pointer gst_element_iterate_sink_pads(Element element);
    /* factory management */
    ElementFactory gst_element_get_factory(Element element);
    @CallerOwnsReturn Bus gst_element_get_bus(Element element);
    boolean gst_element_send_event(Element element, @IncRef Event event);
    boolean gst_element_post_message(Element element, @IncRef Message message);

    boolean gst_element_implements_interface(Element element, NativeLong iface_type);
    /* clocking */
    Clock gst_element_get_clock(Element element);
    boolean gst_element_set_clock(Element element, Clock clock);
    void gst_element_set_base_time(Element element, ClockTime time);
    ClockTime gst_element_get_base_time(Element element);
    void gst_element_set_start_time(Element element, ClockTime time);
    ClockTime gst_element_get_start_time(Element element);

    public static final class GstElementDetails extends com.sun.jna.Structure {
```

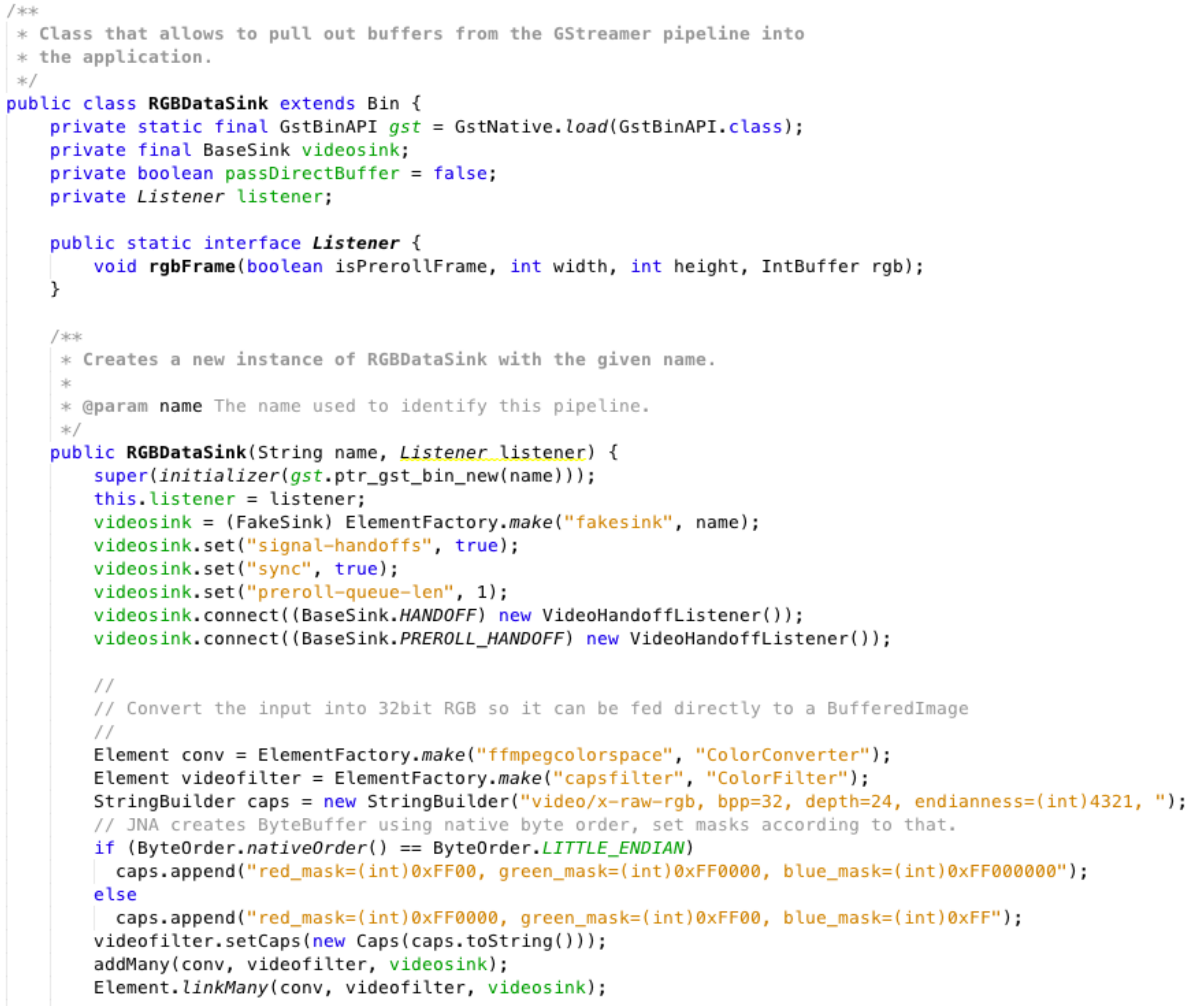
```java
public class Element extends GstObject {
    public static final String GTYPE_NAME = "GstElement";

    private static final GstElementAPI gst = GstNative.load(GstElementAPI.class);

    /**
     * Creates a new instance of Element.  This constructor is used internally.
     *
     * @param init internal initialization data.
     */
    public Element(Initializer init) {
        super(init);
    }

    /**
     * Creates an instance of the required element type, but does not wrap it in
     * a proxy.
     *
     * @param factoryName The name of the factory to use to produce the Element
     * @param elementName The name to assign to the created Element
     * @return a raw element.
     */
    protected static Initializer makeRawElement(String factoryName, String elementName) {
        return initializer(ElementFactory.makeRawElement(factoryName, elementName));
    }

    /**
     * Links this element to another element.
     * The link must be from source to destination; the other direction will not
     * be tried.
     * <p>
     * The function looks for existing pads that aren't linked yet.
     * It will request new pads if necessary. Such pads need to be released manually when unlinking.
     * If multiple links are possible, only one is established.
     *<p>
     * Make sure you have added your elements to a bin or pipeline with
     * {@link Bin#add} or {@link Bin#addMany} before trying to link them.
     *
     * @param dest The {@link Element} containing the destination pad.
     * @return true if the elements could be linked, false otherwise.
     */
    public boolean link(Element dest) {
        return gst.gst_element_link(this, dest);
    }

    /**
     * Chain together a series of elements, with this element as the first in the list.
     * <p>
     * Make sure you have added your elements to a bin or pipeline with
     * {@link Bin#add} or {@link Bin#addMany} before trying to link them.
```

Sample application using Swing

```java
public class VideoPlayer {
    public static void main(String[] args) {
        args = Gst.init("VideoPlayer", args);
        final PlayBin playbin = new PlayBin("VideoPlayer");
        playbin.setInputFile(new File(args[0]));

        SwingUtilities.invokeLater(new Runnable() {

            public void run() {
                VideoComponent videoComponent = new VideoComponent();
                playbin.setVideoSink(videoComponent.getElement());

                JFrame frame = new JFrame("VideoPlayer");
                frame.getContentPane().add(videoComponent, BorderLayout.CENTER);
                frame.setPreferredSize(new Dimension(640, 480));
                frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
                frame.pack();
                frame.setVisible(true);
                playbin.setState(State.PLAYING);
            }
        });
        Gst.main();
        playbin.setState(State.NULL);
    }
}
```

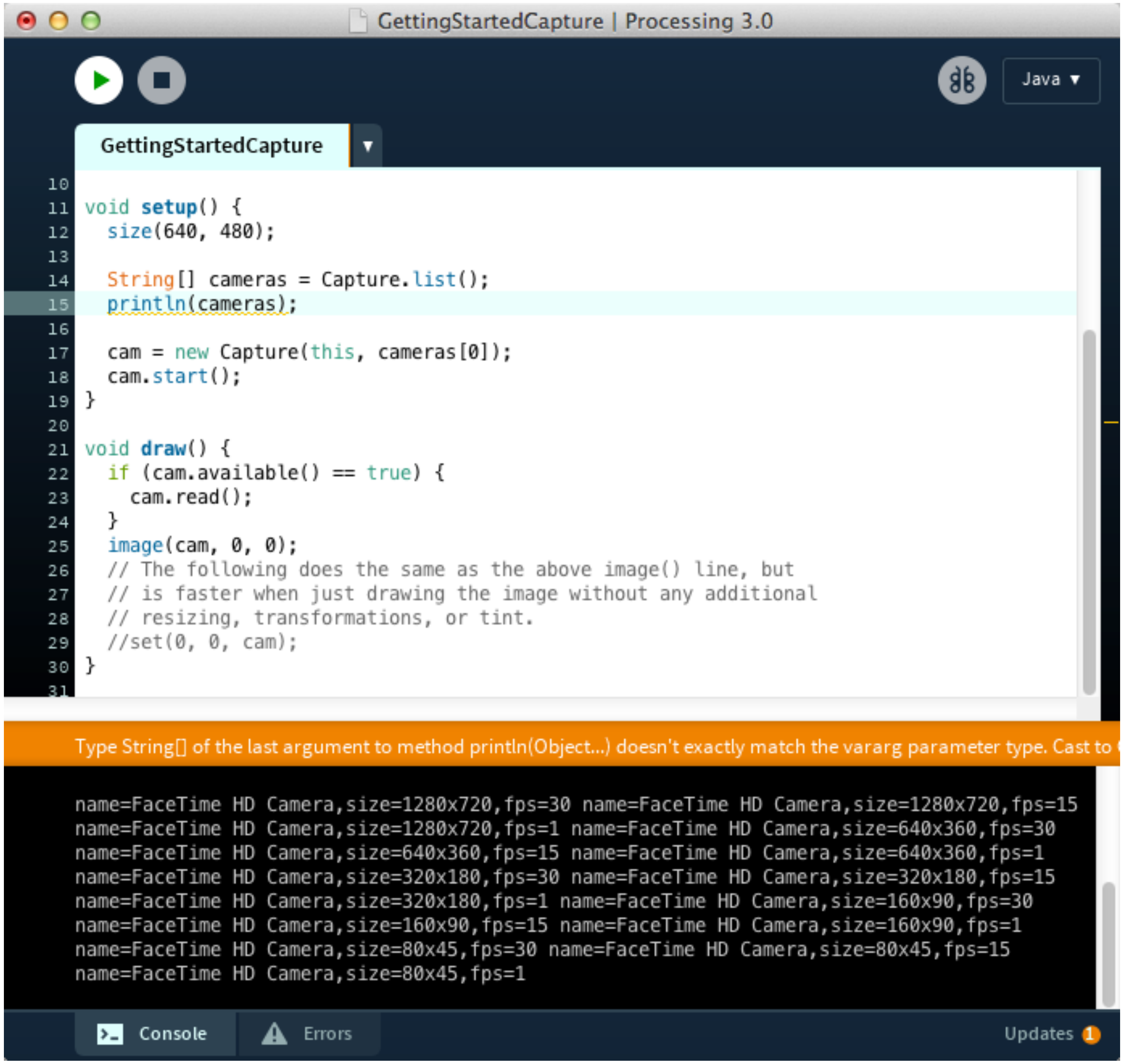
VideoComponent extends javax.swing.JComponent,and···

```java
private class RGBListener implements RGBDataSink.Listener {
    public void rgbFrame(boolean isPrerollFrame, int width, int height, IntBuffer rgb) {
        // If the EDT is still copying data from the buffer, just drop this frame
        //
        if (!bufferLock.tryLock()) {
            return;
        }

        //
        // If there is already a swing update pending, also drop this frame.
        //
        if (updatePending && !isPrerollFrame) {
            bufferLock.unlock();
            return;
        }
        try {
            final BufferedImage renderImage = getBufferedImage(width, height);
            int[] pixels = ((DataBufferInt) renderImage.getRaster().getDataBuffer()).getData();
            rgb.get(pixels, 0, width * height);
            updatePending = true;
        } finally {
            bufferLock.unlock();
        }

        int scaledWidth = currentImage.getWidth();
        if (keepAspect) {
            // Scale width according to pixel aspect ratio.
            Caps videoCaps = videoPad.getNegotiatedCaps();
            Structure capsStruct = videoCaps.getStructure(0);
            if (capsStruct.hasField("pixel-aspect-ratio")) {
                Fraction pixelAspectRatio = capsStruct.getFraction("pixel-aspect-ratio");
                scaledWidth = scaledWidth * pixelAspectRatio.getNumerator() / pixelAspectRatio.getDenominator();
            }
        }

        // Tell swing to use the new buffer
        update(scaledWidth, currentImage.getHeight());
    }
}
```



```java
/**
 * Class that allows to pull out buffers from the GStreamer pipeline into
 * the application.
 */
public class RGBDataSink extends Bin {
    private static final GstBinAPI gst = GstNative.load(GstBinAPI.class);
    private final BaseSink videosink;
    private boolean passDirectBuffer = false;
    private Listener listener;

    public static interface Listener {
        void rgbFrame(boolean isPrerollFrame, int width, int height, IntBuffer rgb);
    }

    /**
     * Creates a new instance of RGBDataSink with the given name.
     *
     * @param name The name used to identify this pipeline.
     */
    public RGBDataSink(String name, Listener listener) {
        super(initializer(gst.ptr_gst_bin_new(name)));
        this.listener = listener;
        videosink = (FakeSink) ElementFactory.make("fakesink", name);
        videosink.set("signal-handoffs", true);
        videosink.set("sync", true);
        videosink.set("preroll-queue-len", 1);
        videosink.connect((BaseSink.HANDOFF) new VideoHandoffListener());
        videosink.connect((BaseSink.PREROLL_HANDOFF) new VideoHandoffListener());

        //
        // Convert the input into 32bit RGB so it can be fed directly to a BufferedImage
        //
        Element conv = ElementFactory.make("ffmpegcolorspace", "ColorConverter");
        Element videofilter = ElementFactory.make("capsfilter", "ColorFilter");
        StringBuilder caps = new StringBuilder("video/x-raw-rgb, bpp=32, depth=24, endianness=(int)4321, ");
        // JNA creates ByteBuffer using native byte order, set masks according to that.
        if (ByteOrder.nativeOrder() == ByteOrder.LITTLE_ENDIAN)
          caps.append("red_mask=(int)0xFF00, green_mask=(int)0xFF0000, blue_mask=(int)0xFF000000");
        else
          caps.append("red_mask=(int)0xFF0000, green_mask=(int)0xFF00, blue_mask=(int)0xFF");
        videofilter.setCaps(new Caps(caps.toString()));
        addMany(conv, videofilter, videosink);
        Element.linkMany(conv, videofilter, videosink);
```

Device probing to list cameras in Processing



```

void setup() {
  size(640, 480);

  String[] cameras = Capture.list();
  println(cameras);

  cam = new Capture(this, cameras[0]);
  cam.start();
}

void draw() {
  if (cam.available() == true) {
    cam.read();
  }
  image(cam, 0, 0);
  // The following does the same as the above image() line, but
  // is faster when just drawing the image without any additional
  // resizing, transformations, or tint.
  //set(0, 0, cam);
}
```

Type String[] of the last argument to method println(Object...) doesn't exactly match the vararg parameter type. Cast to

```
name=FaceTime HD Camera,size=1280x720,fps=30 name=FaceTime HD Camera,size=1280x720,fps=15
name=FaceTime HD Camera,size=1280x720,fps=1 name=FaceTime HD Camera,size=640x360,fps=30
name=FaceTime HD Camera,size=640x360,fps=15 name=FaceTime HD Camera,size=640x360,fps=1
name=FaceTime HD Camera,size=320x180,fps=30 name=FaceTime HD Camera,size=320x180,fps=15
name=FaceTime HD Camera,size=320x180,fps=1 name=FaceTime HD Camera,size=160x90,fps=30
name=FaceTime HD Camera,size=160x90,fps=15 name=FaceTime HD Camera,size=160x90,fps=1
name=FaceTime HD Camera,size=80x45,fps=30 name=FaceTime HD Camera,size=80x45,fps=15
name=FaceTime HD Camera,size=80x45,fps=1
```

```java
static protected ArrayList<String> listDevices(String sourceName,
                                               String propertyName) {
  ArrayList<String> devices = new ArrayList<String>();
  try {
    // Using property-probe interface
    Element videoSource = ElementFactory.make(sourceName, "Source");
    PropertyProbe probe = PropertyProbe.wrap(videoSource);
    if (probe != null) {
      Property property = probe.getProperty(propertyName);
      if (property != null) {
        Object[] values = probe.getValues(property);
        if (values != null) {
          for (int i = 0; i < values.length; i++) {
            if (values[i] instanceof String) {
              devices.add((String)values[i]);
            } else if (values[i] instanceof Integer) {
              devices.add(((Integer)values[i]).toString());
            }
          }
        }
      }
    }
  } catch (IllegalArgumentException e) {
    if (PApplet.platform == LINUX) {
      // Linux hack to detect currently connected cameras
      // by looking for device files named /dev/video0, /dev/video1, etc.
      devices = new ArrayList<String>();
      String dir = "/dev";
      File libPath = new File(dir);
      String[] files = libPath.list();
      if (files != null) {
        for (int i = 0; i < files.length; i++) {
          if (-1 < files[i].indexOf("video")) {
            devices.add("/dev/" + files[i]);
          }
        }
      }
    } else {
      PGraphics.showWarning("The capture plugin does not support " +
                            "device query!");
      devices = new ArrayList<String>();
    }
  }
  return devices;
}
```

```java
static protected ArrayList<String> listResolutions(String sourceName,
                                                   String propertyName,
                                                   Object propertyValue) {
  // Creating temporary pipeline so that we can query
  // the resolutions supported by the device.
  Pipeline testPipeline = new Pipeline("test");
  Element source = ElementFactory.make(sourceName, "source");
  source.set(propertyName, propertyValue);

  BufferDataAppSink sink = new BufferDataAppSink("sink", "",
      new BufferDataAppSink.Listener() {
        public void bufferFrame(int w, int h, Buffer buffer) { }
      });
  testPipeline.addMany(source, sink);
  Element.linkMany(source, sink);

  // Play/pause sequence (with getState() calls to to make sure
  // all async operations are done) to trigger the capture momentarily
  // for the device and obtain its supported resolutions.
  testPipeline.play();
  testPipeline.getState();
  testPipeline.pause();
  testPipeline.getState();

  ArrayList<String> resolutions = new ArrayList<String>();
  addResFromSource(resolutions, source);

  testPipeline.stop();
  testPipeline.getState();

  if (sink != null) {
    sink.removeListener();
    sink.dispose();
  }

  testPipeline.dispose();
  return resolutions;
}
```

Last, but not least, we needed GStreamer binaries for Windows and Mac OS X

- We managed to build them using MacPorts on Mac, cross-compilation with mingw64 on Linux to output Windows .dlls
- These builds of 0.10 have been in use since Processing 1.x until today…

corbis

GStreamer 1.0: The future is now…

We have 3 options to transition to GStreamer 1.x:

1. Writing a minimal binding only for Processing, using JNI
2. We update gstreamer-java so it works with 1.x
3. We auto-generate the bindings form the .gir files!

https://github.com/gstreamer-java

1. Minimal JNI binding, only for Processing

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <jni.h>
#include <gst/gst.h>
#include <gst/app/gstappsink.h>   // for GstAppSink, part of gstreamer-plugins-base
#include "impl.h"
#include "iface.h"

GThread *thread;
GMainLoop *loop;

#define MAX_VIDEOS 10
video videos[MAX_VIDEOS];


JNIEXPORT jboolean JNICALL Java_processing_simplevideo_SimpleVideo_gstreamer_1init(JNIEnv *env, jclass cls)
{
  GError *err = NULL;

  gst_init_check(NULL, NULL, &err);
  if (err != NULL) {
    g_print("Could not initialize library: %s\n", err->message);
    g_error_free(err);
    return FALSE;
  }

  thread = g_thread_new("simplevideo-mainloop", simplevideo_mainloop, NULL);
  return TRUE;
}


static void* simplevideo_mainloop(void *data) {
  loop = g_main_loop_new(NULL, FALSE);
  g_main_loop_run(loop);
  return NULL;
}
```

2. Updated JNA bindings, gstreamer1.x-java

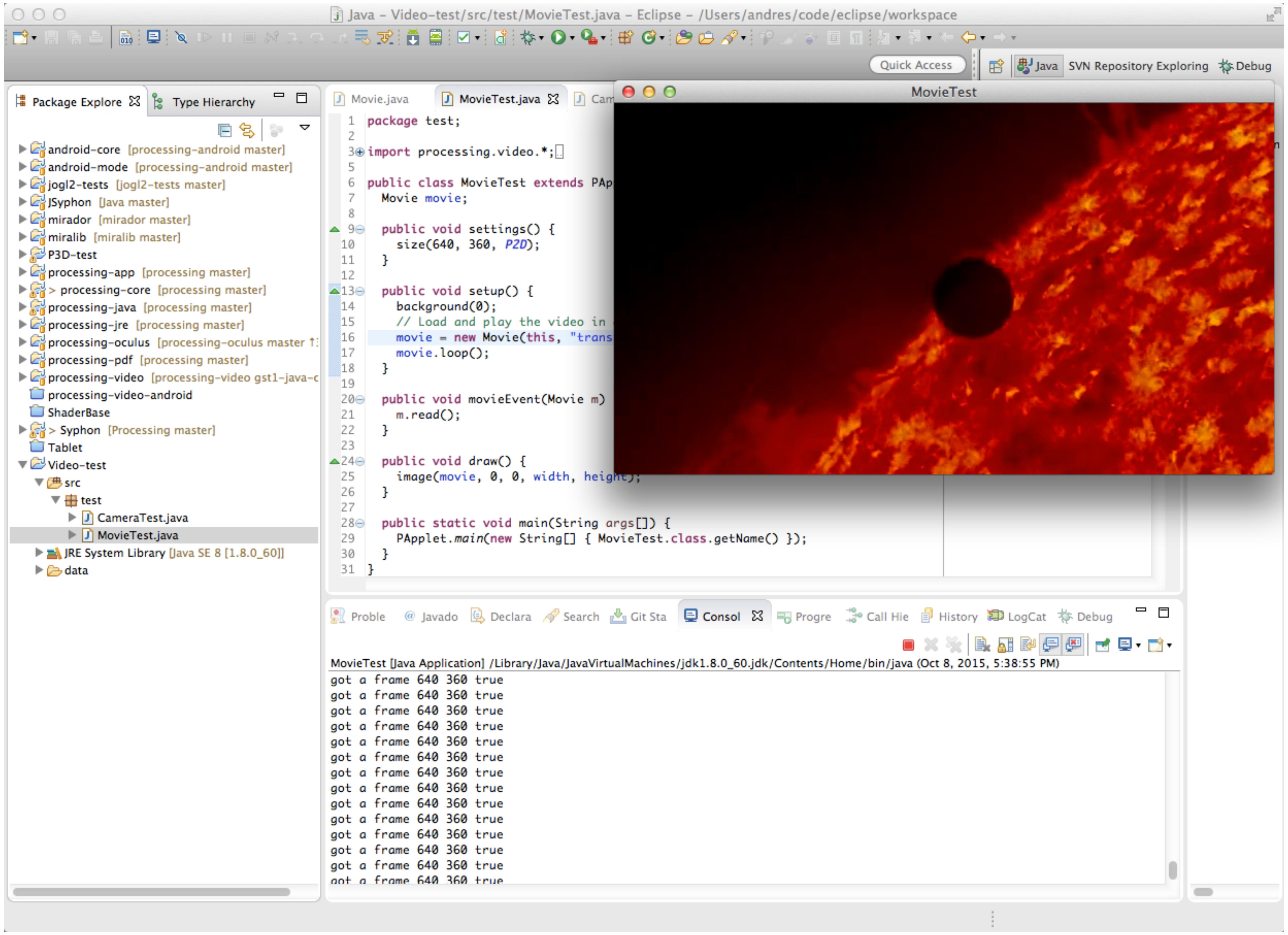
- One year ago, GitHub user armouroflight pushed a version of gstreamer-java mostly compatible with GStreamer 1.0/1.2:

  https://github.com/armouroflight/gstreamer1.x-java

- Neil C. Smith, also a Processing enthusiast, took armouroflight's code and create a more stable version:

  https://github.com/gstreamer-java/gst1-java-core

We already have a version of the Processing video library based on gst1-java-core with movie and capture support:



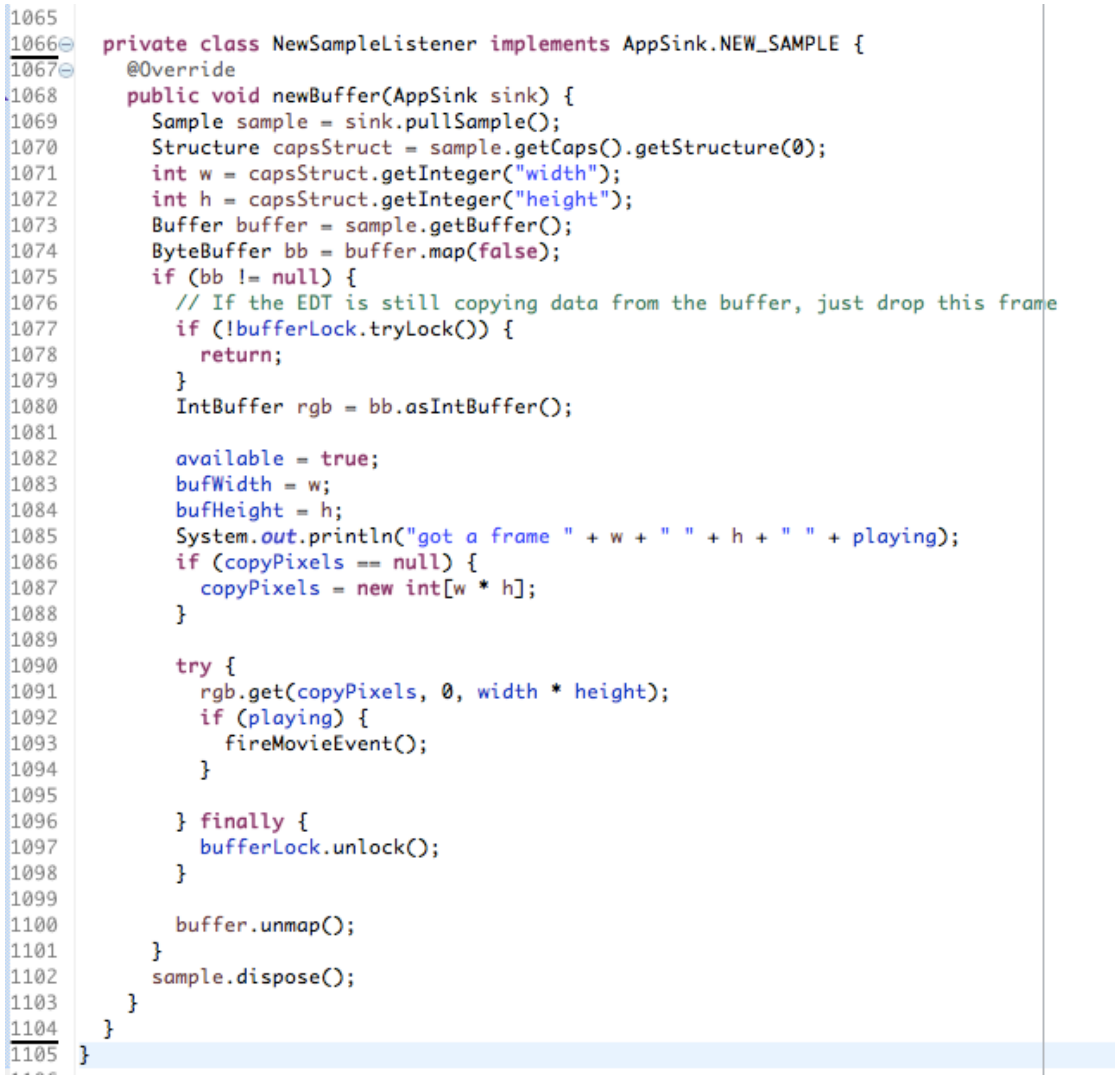
Java – Video-test/src/test/MovieTest.java – Eclipse – /Users/andres/code/eclipse/workspace

```
▶ android-core  [processing-android master]
▶ android-mode  [processing-android master]
▶ jogl2-tests  [jogl2-tests master]
▶ JSyphon  [Java master]
▶ mirador  [mirador master]
▶ miralib  [miralib master]
▶ P3D-test
▶ processing-app  [processing master]
▶ > processing-core  [processing master]
▶ processing-java  [processing master]
▶ processing-jre  [processing master]
▶ processing-oculus  [processing-oculus master ↑
▶ processing-pdf  [processing master]
▶ processing-video  [processing-video gst1-java-c
  processing-video-android
  ShaderBase
▶ > Syphon  [Processing master]
  Tablet
▼ Video-test
  ▼ src
    ▼ test
      ▶ CameraTest.java
      ▶ MovieTest.java
  ▶ JRE System Library [Java SE 8 [1.8.0_60]]
  ▶ data
```

```java
package test;

import processing.video.*;

public class MovieTest extends PAp
  Movie movie;

  public void settings() {
    size(640, 360, P2D);
  }

  public void setup() {
    background(0);
    // Load and play the video in
    movie = new Movie(this, "trans
    movie.loop();
  }

  public void movieEvent(Movie m)
    m.read();
  }

  public void draw() {
    image(movie, 0, 0, width, height);
  }

  public static void main(String args[]) {
    PApplet.main(new String[] { MovieTest.class.getName() });
  }
}
```

MovieTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Oct 8, 2015, 5:38:55 PM)

```
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
got a frame 640 360 true
```



```
  private class NewSampleListener implements AppSink.NEW_SAMPLE {
    @Override
    public void newBuffer(AppSink sink) {
      Sample sample = sink.pullSample();
      Structure capsStruct = sample.getCaps().getStructure(0);
      int w = capsStruct.getInteger("width");
      int h = capsStruct.getInteger("height");
      Buffer buffer = sample.getBuffer();
      ByteBuffer bb = buffer.map(false);
      if (bb != null) {
        // If the EDT is still copying data from the buffer, just drop this frame
        if (!bufferLock.tryLock()) {
          return;
        }
        IntBuffer rgb = bb.asIntBuffer();

        available = true;
        bufWidth = w;
        bufHeight = h;
        System.out.println("got a frame " + w + " " + h + " " + playing);
        if (copyPixels == null) {
          copyPixels = new int[w * h];
        }

        try {
          rgb.get(copyPixels, 0, width * height);
          if (playing) {
            fireMovieEvent();
          }

        } finally {
          bufferLock.unlock();
        }

        buffer.unmap();
      }
      sample.dispose();
    }
  }
}
```

3. Automated binding generation, using gir2java

- This is a project started during GSoC 2014 by Roland Elek, and co-mentored by Levente Farkas and myself
- It is parser and generator that generates the Java bindings automatically using GObject Instrospection .gir files
- It went pretty far: GLib, GObject, Gio, GModule, Gst-1.0, GstBase-1.0, and GstVideo-1.0 compile without errors on a recent Linux distro, and it is capable of running a simple video playback pipeline

https://github.com/gstreamer-java/gir2java

```java
package generated.gstreamer10.gst;

import generated.glib20.glib.GCond;
import generated.glib20.glib.GList;
import generated.glib20.glib.GRecMutex;
import org.bridj.BridJ;
import org.bridj.IntValuedEnum;
import org.bridj.Pointer;
import org.bridj.ann.Field;
import org.bridj.ann.Library;
import org.bridj.ann.Ptr;

@Library("gstreamer-1.0")
public class GstElement
    extends GstObject
{


    static {
        BridJ.register();
    }

    public GstElement() {
        super();
    }

    public GstElement(Pointer pointer) {
        super(pointer);
    }

    protected static native boolean gst_element_register(
        @Ptr
        long plugin,
        @Ptr
        long name, long rank, long type);

    public static boolean register(Pointer plugin, Pointer name, long rank, long type) {
        return GstElement.gst_element_register(Pointer.getPeer(plugin), Pointer.getPeer(name), rank, type);
    }

    protected native void gst_element_abort_state(
        @Ptr
        long element);

    public void abort_state() {
        this.gst_element_abort_state(Pointer.pointerTo(this, GstElement.class).getPeer());
    }

    protected native boolean gst_element_add_pad(
        @Ptr
        long element,
        @Ptr
        long pad);

    public boolean add_pad(Pointer pad) {
        return this.gst_element_add_pad(Pointer.pointerTo(this, GstElement.class).getPeer(), Pointer.getPeer(pad));
    }

    protected native void gst_element_create_all_pads(
        @Ptr
        long element);

    public void create_all_pads() {
        this.gst_element_create_all_pads(Pointer.pointerTo(this, GstElement.class).getPeer());
    }
```

```java
public static void main(String[] args) {
    Gst.init(null, null);

    Pointer<GMainLoop> mainLoopPointer = GMainLoop._new(null, false).as(GMainLoop.class);
    Pointer<GstElement> playbin = (Pointer<GstElement>)GstElementFactory.make(
            Pointer.pointerToCString("playbin"),
            Pointer.pointerToCString("the_playbin")
    );

    if (playbin == null) {
        System.err.println("The playbin could not be created.");
        return;
    }

    String uri = "file://" + System.getProperty("user.dir") + "/test.ogv";
    System.out.println("Trying to play " + uri);
    GObject.set(
            playbin,
            Pointer.pointerToCString("uri"),
            Pointer.pointerToCString(uri),
            null);

    GstPipeline pipeline = playbin.as(GstPipeline.class).get();
    pipeline.set_state(GstState.GST_STATE_PLAYING);
    GMainLoop mainLoop = mainLoopPointer.get();
    System.out.println("Everything seems OK so far, starting main loop");

    GstBus bus = pipeline.get_bus().get();
    BusWatch busWatch = new BusWatch();
    bus.add_watch(Pointer.pointerTo(busWatch).as(GstBusFunc.class), mainLoopPointer);
    bus.gstobject_unref();

    mainLoop.run();
    //The fun stuff happens, then:
    System.out.println("Main loop finished");
    pipeline.set_state(GstState.GST_STATE_NULL);
    pipeline.gstobject_unref();
}
```

However, it is not finished…

- Parsing fails non-deterministically with type reference resolution within a single file.
- Callback parsing and generation not implemented.
- API is not Java-like, it needs another layer on top of the classes generated by the parser.
- Automatic reference counting is missing.

Things left To Do/Discuss

1. Which approach to choose (I lean towards JNA)
2. OpenGL integration (GL plugins!)
3. Raspberry Pi integration (OpenMAX acceleration?)
4. JavaFX support (what about lightweight GStreamer being bundled by Oracle with JavaFX)
5. Bundling GStreamer 1.x native libs for OS X and Windows with the Processing video library

Questions/Comments?

Thanks to the many contributors and members of the Processing and GStreamer communities around the world!

…and thank you!

PROCESSING FOUNDATION

gstreamer