Technische Universität Darmstadt

Telecooperation/RBG

Java-based Generation of AV Content Using ANIMAL

Dr. Guido Rößling <roessling@acm.org>
TU Darmstadt

Overview

What is ANIMAL?

- ANIMAL: *Advanced Navigation and Interactive Modeling for Animations in Lectures*
- Developed 1998-2001 Uni Siegen, since 2001 TU Darmstadt
- Several features such as arbitrary rewind, I18N, …
- Example animation window:
  - Speed control
  - Zoom – from „0%“ to 500% (currently 83%)
  - Display area, including code and counters
  - Bidirectional navigation
  - Kiosk mode (run without interaction)
  - Manual step control

Basic Concepts of AV in ANIMAL

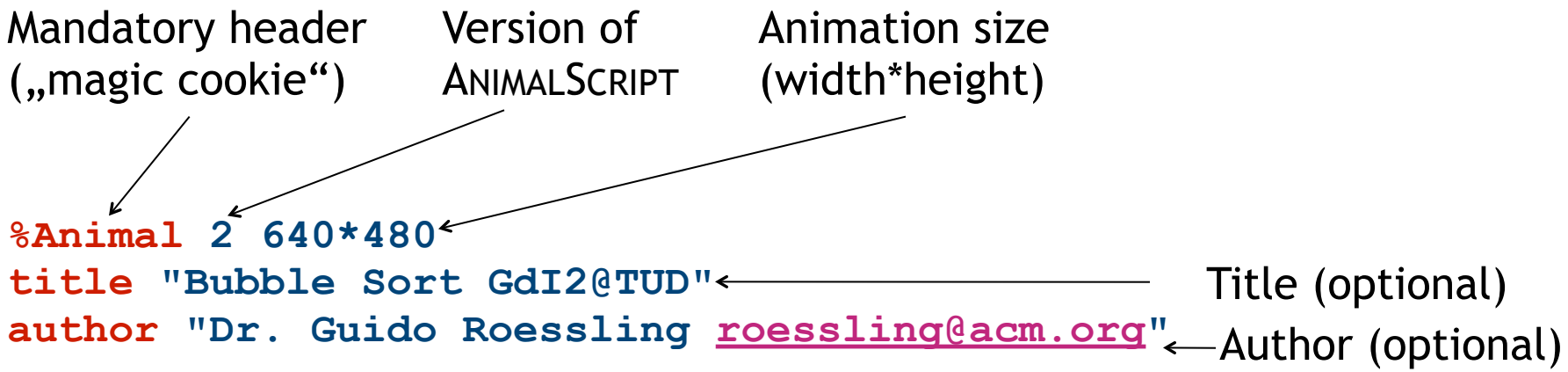
- ANIMAL distinguishes between the following elements:
  - A *graphical object* (or "*primitive*") has a **state** and can **paint** itself
  - An *animation effect* (or "*animator*") transforms objects
  - An *animation step* contains an arbitrary number of *animators*
  - An *animation* contains an arbitrary number of *animation steps*
- All animation effects can be timed:
  - *Offset* between the start of the step and the effect's start
  - *Duration* of the effect
- This is a simplified view; ANIMAL is more complex
  - But you do not need to know / understand it better for the API!
- ANIMAL URL: http://www.algoanim.info/Animal2
  - Feedback is always welcome at <roessling@acm.org> ☺

A Brief Introduction to ANIMALSCRIPT

- ANIMALSCRIPT is ANIMAL's built-in scripting language
  - Editable in any text editor
  - As well as in an Eclipse plug-in called "Animalipse"
  - Can also be generated in programs, e.g. using *System.out.println(…)*
  - Is also generated by the API shown in this presentation
- Basic Ideas:
  - Each line (apart from the header) contains at most *one* command
  - Each command appears in an *animation step* (with one command)
  - If you want multiple commands in one step, surround them with **{ }**
- Basic Commands:
  - Defining *graphical objects*, e.g. a new square, array, …
  - Defining *animation effects*, e.g. "move the following objects like this"
  - Comments (**#** equals "**//**" in Java); braces **{ }**; debug output

A Brief Introduction to ANIMALSCRIPT

- Typical file header:



A Brief Introduction to AnimalScript

- A brief example for graphical primitives:

```
{
  text "title" "Bubble Sort GdI2@TUD" (20,35) color (0,0,0)
    depth 1 font SansSerif size 20 bold
  rectangle "headerRect" offset (-5,-5) from "title" NW
    offset (5,5) from "title" SE color (0,255,0) depth 3 filled
    fillColor (192,192,192)
  text "descrHd" "Bubble Sort: Synopsis" (20,80)
}
```

- Visual output in the animation:
  - Black Sans-Serif text, bold
  - Green rectangle filled with dark grey surrounds
  - Second text is placed below the rectangle

- In the above, *optional* arguments are shown now outlined
  - As you can see, they make up much of the content!
  - AnimalScript will use the last fitting definition (“same font as before”)

A Brief Introduction to ANIMALSCRIPT

- Changing objects is easy, too…:

```
color "descrHd" red within 10 ticks
hide "descrHd"
array "array" (30,150) length 8 "1" "3" "7" "5" "2" "6" "8" "4"
highlightArrayElem on "array" position 3
highlightArrayElem on "array" position 2
arraySwap on "array" position 2 with 3 within 10 ticks
```

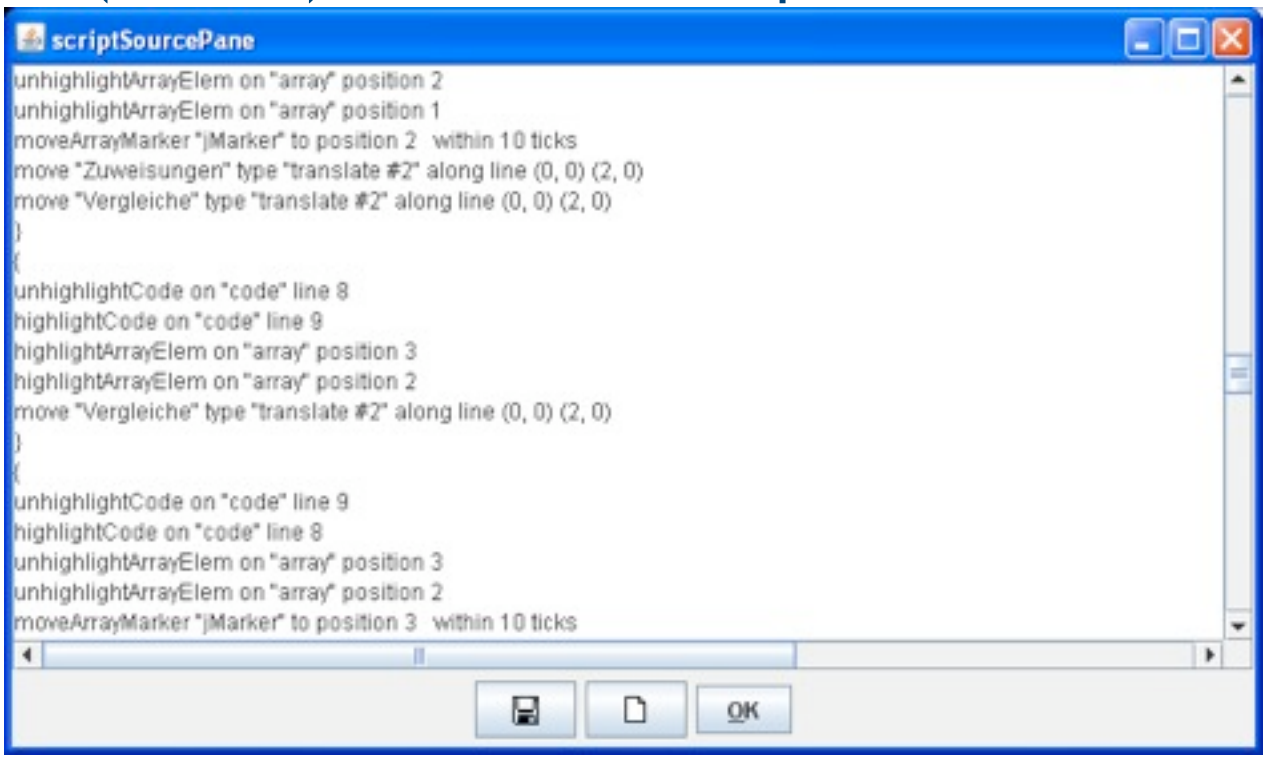
- First, change the color of the heading to red
- Then hide the element
- Now show an array of size 8 with values {1, 3, 7, 5, 2, 6, 8, 4}
- Highlight the array element at position 3 (→ value 5)
- Highlight the array element at position 2 (→ value 7)
- Swap the two positions within 10 animation frames („ticks“)

A Brief Introduction to ANIMALSCRIPT

- ANIMALSCRIPT supports the following basic primitives:
  - Point, square, rectangle, triangle, polygon, line, text, arc, circle, …
- It also offers some CS-specific data structures
  - Arrays with „array markers“ (pointers that indicate positions)
  - List elements
  - Code blocks with indentation and highlighting
- A BNF description of ANIMALSCRIPT is included in ANIMAL itself
  - Select *Help* → *AnimalScript Definition* from the menu bar
- Animators are very flexible
  - Good timing interpolation
  - Can receive “method name” to distinguish different operations
    - Move type “translate” → move the complete object
    - Move type “translate #2” → move only node #2 of the object(s)

A Brief Introduction to AnimalScript

- Here‘s an (almost) readable excerpt of ANIMALSCRIPT



- You should get a rough idea of what is happening here
  - ANIMALSCRIPT is rather readable!

The ALGOANIM API: Basic Concepts and Approach

- Dumping ANIMALSCRIPT code inside an application is messy

```
int oldDist = -1;
sb.append("  text \"dist\" \"dist:\" offset (40, 20) from \"array\" E");
for (dist = 1; dist < a.length / 9; dist = 3 * dist + 1) {
  toggleStep();
  if (oldDist != -1) {
    sb.append("  hide \"dist").append(oldDist).append("\"");
  }
  sb.append("  text \"dist").append(dist).append("\" offset (10, 0)");
  sb.append(" from \"dist\" baseline");
  oldDist = dist;
}
```

- This code comes from ShellSort
  - You can only tell this by how "**`dist`**" is modified in the **`for`** loop
- The actual code becomes difficult to read
- We provide a Java API for generating ANIMALSCRIPT

The ANIMALSCRIPT API: Basic Concepts and Approach

- Basic Concepts and Goals:
  - Use good software engineering
    - Interfaces, abstract classes
    - *Factory* Design Pattern
  - Provide support for extension to other languages and new object types
- Factory class *Language* is the central element:

```
// Generate a new Language instance for content creation
// Parameter: Animation title, author, width, height
Language lang = new AnimalScript("Quicksort Animation",
                                 "Dr. Guido Roessling", 640, 480);
// Activate step control
lang.setStepMode(true);
```

- Step control:
  - If activated, animation steps are changed by calling *nextStep()*
  - If not activated, each command will create a new step

Defining Graphical Objects

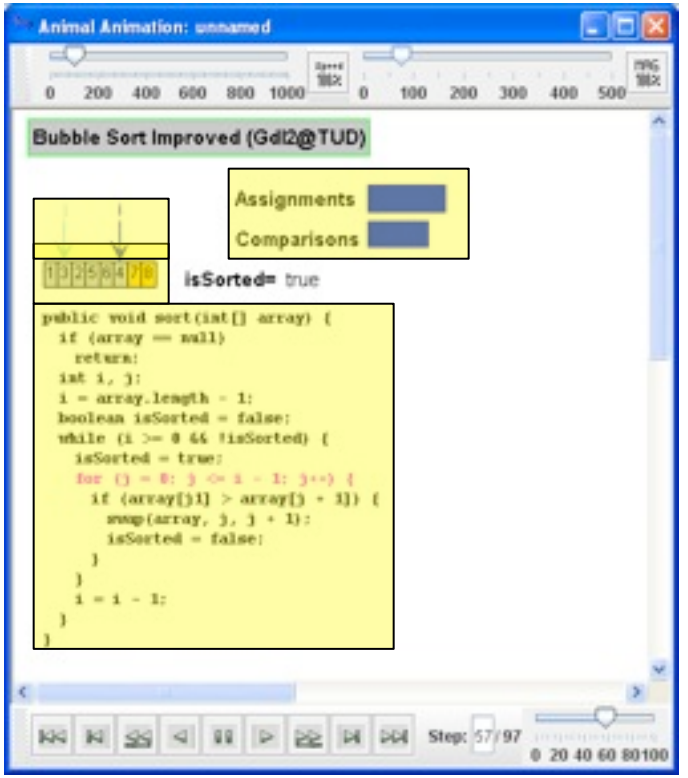
- The *Language* Factory can also create objects
- This usually requires the following parameters:
  - Location – as an absolute coordinate or relative to other objects;
  - Value - a **`String`** for text objects, **`int[]`** for int arrays, …
  - Name of the object for later access,
  - Display Options - delayed, hidden, or **`null`** (for default visibility)
  - Visual Properties – defined in a few slides, ignore for now
- Here's how you can create a visual Int-Array :

```
// Create a new int[] object (will normally exist before)
int[] arrayContents = new int[] { 1, 13, 7, 2, 11};
// Parameter: Location, Value, Name, Display Opt., Visual P.
IntArray array = lang.newIntArray(new Coordinates(10, 30),
                    arrayContents, "array", null, arrayProps);
```

- Currently supported:

Defining Graphical Objects

- The API supports several data types relevant for CS:
  - int[], String[], int[][], String[][], Graph, List elements, Code blocks
- Of course, this includes the “appropriate” operations
  - *put*, *swap*, *link*, …
  - Highlighting elements, positions,….
- See the example to the right:
  - **`int[]`** highlighting the sorted elements
  - Code block highlighting the current line
  - Two index pointers on the array
  - „Step counters“ for assignments and comparisons



Object Placement

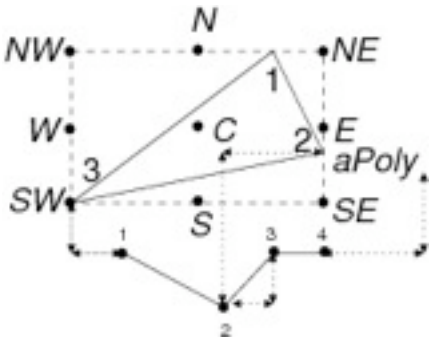
- Placement using absolute coordinates:
  - `new Coordinates(x, y)`
- Relative to another object:
  - `new Offset(dx, dy, reference, direction)`
  - `dx, dy` are the offsets used
  - `reference` is a reference to an existing primitive (Java API object)
  - `direction` is one of the following constants:

AnimalScript.DIRECTION_NW	AnimalScript.DIRECTION_SW
AnimalScript.DIRECTION_N	AnimalScript.DIRECTION_S
AnimalScript.DIRECTION_NE	AnimalScript.DIRECTION_SE
AnimalScript.DIRECTION_W	AnimalScript.DIRECTION_BASELINE_START
AnimalScript.DIRECTION_C	AnimalScript.DIRECTION_BASELINE_END
AnimalScript.DIRECTION_E	

  - The following slide shows an example for the direction types

Object Placement

- Offsets refer to the *bounding box* of the referenced object
  - Bounding box: smallest rectangle covering the complete object
  - To the right, you see a polygon called „aPoly“ with nodes 1 to 3
  - The bounding box is shown as a dashed box
  - The text codes NW, …, SE represent the
    - Corners of the bounding box – NW, NE, SW, SE
    - Centers of the sides of the bounding box – N, W, E, S
    - Center of the bounding box – C
  - Additionally, the baseline can be used for texts
    - BASELINE_START, BASELINE_END
    - Useful if the text has characters such as y, q, p ☺



- The other object can then be defined relative to these points
  - E.g., line node #1 is at

```
new Offset(10, 10, aPoly, AnimalScript.DIRECTION_SW)
```

Defining and Sharing Visual Properties

- The API uses templates to set the visual properties of objects
  - Similarly to CSS, visual property objects are assigned to objects
  - These property objects can be reused as often as wanted
  - Usually, even the predefined values are good enough ☺
- Here's an example for defining visual properties:

```
// create array properties with default values
ArrayProperties arrayProps = new ArrayProperties();
// Redefine properties: border red, filled with gray
arrayProps.set(AnimationPropertiesKeys.COLOR_PROPERTY,
                        Color.RED); // Base color red (border)
arrayProps.set(AnimationPropertiesKeys.FILLED_PROPERTY,
                        true); // filled
arrayProps.set(AnimationPropertiesKeys.FILL_PROPERTY,
                        Color.GRAY); // fill color gray
```

Defining and Sharing Visual Properties

- Each object type has its own set of properties
- Each property type has a default value
  - E.g., colors will always be *java.awt.Color.BLACK* unless redefined
- The table on the following slide lists all properties
  - Each property has the following form:

    `AnimationPropertiesKeys.X_PROPERTY`
- Properties are set as shown in the previous slide
  1. Create the properties object *props* (or use an existing one)
  2. Invoke *props.set(AnimationPropertiesKeys.X_PROPERTY, value);*
  - *value* must match the type expected for the property
- Note: do not try to remember the table on the next slide
  - The information is also included in the online documentation!

Overview of Visual Properties

Property (AnimationPropertiesKeys.X_PROPERTY)				Primitives that support this property																
				Arc	Array	ArrayMarker	Circle	CircleSeg	Ellipse	Graph	ListElement	Matrix	Point	Polygon	Polyline	Rect	SourceCode	Square	Text	Triangle
X	Type	Range	Default																	
ANGLE	int	0..360	0	X				X												
BOLD	boolean	false, true	false														X			
BOXFILLCOLOR	Color	any Color	Color.BLACK								X									
BWARROW	boolean	false, true	false	X				X							X					
CASCADED	boolean	false, true	false		X															
CELLHIGHLIGHT	Color	any Color	Color.BLACK		X							X								
CENTERED	boolean	false, true	false																X	
CLOCKWISE	boolean	false, true	false	X				X												
CLOSED	boolean	false, true	false	X				X												
COLOR	Color	any color	Color.BLACK	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CONTEXTCOLOR	Color	any Color	Color.BLACK														X			
COUNTERCLOCKWISE	boolean	false, true	false	X				X												
DEPTH	int	0..Integer.MAX_VALUE	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DIRECTED	boolean	false, true	false							X										
DIRECTION	boolean	false, true	false		X															
EDGECOLOR	Color	any Color	Color.BLACK							X										
ELEMENTCOLOR	Color	any Color	Color.BLACK		X					X		X								
ELEMHIGHLIGHT	Color	any Color	Color.BLACK		X					X		X								
FILL	Color	any color	Color.BLACK	X	X		X	X	X	X		X		X		X		X		X
FILLED	boolean	false, true	false	X	X		X	X	X			X		X		X		X		X
FONT	Font	any Font	SansSerif 12														X		X	
FWARROW	boolean	false, true	false	X				X							X					
HIDDEN	boolean	false, true	false	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
HIGHLIGHTCOLOR	Color	any Color	Color.BLACK							X							X			
INDENTATION	int	0...30	1														X			
ITALIC	boolean	false, true	false														X			
LABEL	String	any String	""			X														
NODECOLOR	Color	any Color	Color.BLACK							X										
POINTERAREACOLOR	Color	any Color	Color.BLACK								X									
POINTERAREAFILLCOLOR	Color	any Color	Color.BLACK								X									
POSITION	int										X									
ROW	int	0...100	1														X			
SIZE	int	6...100	10														X			
STARTANGLE	int	0..360	0	X				X												
TEXT	String	any String	""								X									
TEXTCOLOR	Color	any Color	Color.BLACK								X									
WEIGHTED	boolean	false, true	false							X										

Defining Animation Effects

- All objects can also be animated
  - Show / hide, move, change color, rotate, …
  - There are special effects for certain types, e.g. *swap* for arrays
- Duration and Delay can be defined based on…
  - *milliseconds*: `new MsTiming(100) // 100 ms`
  - *animation frames*: `new TicksTiming(20) // 20 frames`
- There is also a special "method name" parameter
  - This allows using subtypes of a given effect, e.g. for color changing:
    - „color": Change the base color of the object(s)
    - „fillColor": Change the fill color of the object(s)
- The available method names depend on…
  - The type of the underlying object(s)
  - Certain properties of the object(s)
    - If the object is not *filled*, you cannot change its fill color

Defining Animation Effects

- In the following example, we change the fill color of a square

```
// Change the square's fill color to Orange, starting after a
//   delay of 100ms and using 20 frames
// Parameters: method name, target color, delay (or null),
//   duration
square.changeColor("fillColor", Color.ORANGE,
                   new MsTiming(100), new TicksTiming(20));
```

- If you are unsure what effects are available:
  - load the code into ANIMAL (or draw an appropriate object)
  - Select the proper animation effect in the “Animation Overview”
  - Select the object and check the entries in the “method” list

A Longer Example

- In the following slides, we will look at Selection Sort
- We will transform this into a full-fledged animation
- We start with the original Java code
- We then perform the following steps:
  1. Visualize the *array* operations
  2. Visualize the *array index variables*
  3. Show the underlying source code
  4. Highlight the current code lines
- But first, let us look at the surrounding class code
  - Copy this from the web page to save typing ☺

Preparation

- Your wrapping class code should look roughly like this:
  - Note: you have to add the “proper” import statements etc.!

```
public class SelectionSortDemo {
  protected Language lang;
  private ArrayProperties arrayProps;
  public void init() { // initialize the main elements
  // Generate a new Language instance for content creation
  // Parameter: Animation title, author, width, height
  lang = new AnimalScript("Quicksort Animation", "Dr. Guido Roessling", 640, 480);
  // Activate step control
  lang.setStepMode(true);

  // create array properties with default values
  arrayProps = new ArrayProperties();
  // Redefine properties: border red, filled with gray
  arrayProps.set(AnimationPropertiesKeys.ELEMENTCOLOR_PROPERTY, Color.RED); // color red
  arrayProps.set(AnimationPropertiesKeys.FILLED_PROPERTY, true); // filled
  arrayProps.set(AnimationPropertiesKeys.FILL_PROPERTY, Color.GRAY); // fill color gray
}
```

Preparation

```
public static void main(String[] args) {
    SelectionSortDemo ssd = new SelectionSortDemo();
    ssd.init();
    int[] original = new int[] { 1, 13, 7, 2, 11};
    ssd.selectionSort(original);
    System.out.println(ssd.lang.toString());
  }

  public void selectionSort(int[] array) {
    // to be done
  }
}
```

- After execution, ANIMALSCRIPT code is dumped to System.out
- You can redirect it to a file and load it in in Animal…
- …or copy it from Console to *Windows → AnimalScript Code*

Example: Selection Sort I – Java Code

Here is the original Java code without animation support:

```
public void selectionSort(int[] inputArray) {
  int i, j, minIndex;
  for (i = 0; i < inputArray.length - 1; i++) {
    minIndex = i;
    for (j = i + 1; j < inputArray.length; j++)
      if (inputArray[j] < inputArray[minIndex])
        minIndex = j;
    int tmp = inputArray[i]; // triangular swap
    inputArray[i] = inputArray[minIndex];
    inputArray[minIndex] = tmp;
  }
}
```

Transformation Step 1

- In this step, we replace the Java type int[]
- We use the API-provided type IntArray
  - Found in package *algoanim.primitives*
  - Generated via the *Language* instance
- Steps to be taken:
  - Convert int[] passed in to a IntArray object

```
IntArray array = lang.newIntArray(new Coordinates(10, 30),
        inputArray, "array", null, arrayProps);
```

  - Convert read access to int[] to a getData() call on the IntArray
    - `inputArray[i]` → array.getData(i) etc.
  - Convert write accesses to put() call on the IntArray
    - No write access in this example
  - Convert triangular exchange to swap() call on the IntArray

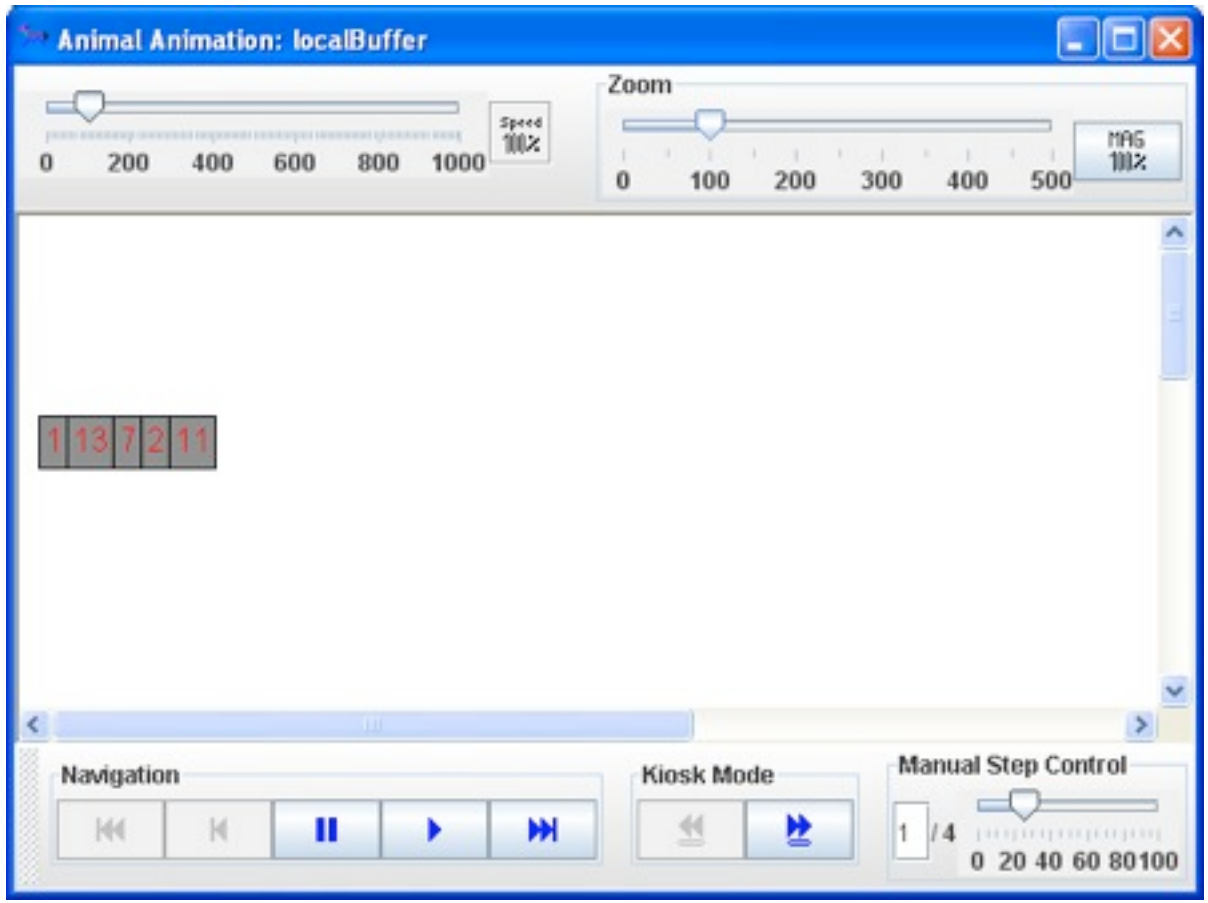
→ `array.swap(i, minIndex, null, timing);`

Result of Transformation Step 1

```
public void selectionSort(int[] inputArray) {
  // wrap int[] to IntArray instance
  IntArray array = lang.newIntArray(new Coordinates(10, 30),
                   inputArray, "array", null, arrayProps);
  // declare a default duration for swap effects
  Timing defaultTiming = new TicksTiming(15);
  int i, j, minIndex;
  for (i = 0; i < array.getLength() - 1; i++) {
    minIndex = i;
    for (j = i + 1; j < array.getLength(); j++)
      if (array.getData(j) < array.getData(minIndex))
        minIndex = j;
    array.swap(i, minIndex, null, defaultTiming); // Swap
  }
}
```

Visual Result of Step 1

- This is what your output should look like (roughly):



Step 2: Visualize the *array index variables*

- Now we can also replace the variables *i*, *j*, *minIndex*
- We use the API-provided type *ArrayMarker*
  - Also found in package *algoanim.primitives*
  - Generated via the *Language* instance
- Steps to be taken:
  - Convert i, j, minIndex declarations to ArrayMarker objects

```
ArrayMarker i = lang.newArrayMarker(array, 0, "i", null);
```

  - Convert read accesses to getPosition() calls on the ArrayMarker
  - Convert increment operations increment(offset, duration) calls on the ArrayMarker
  - Adapt the step control so things happen at different times
    - Add lang.nextStep() after (nearly) all operations

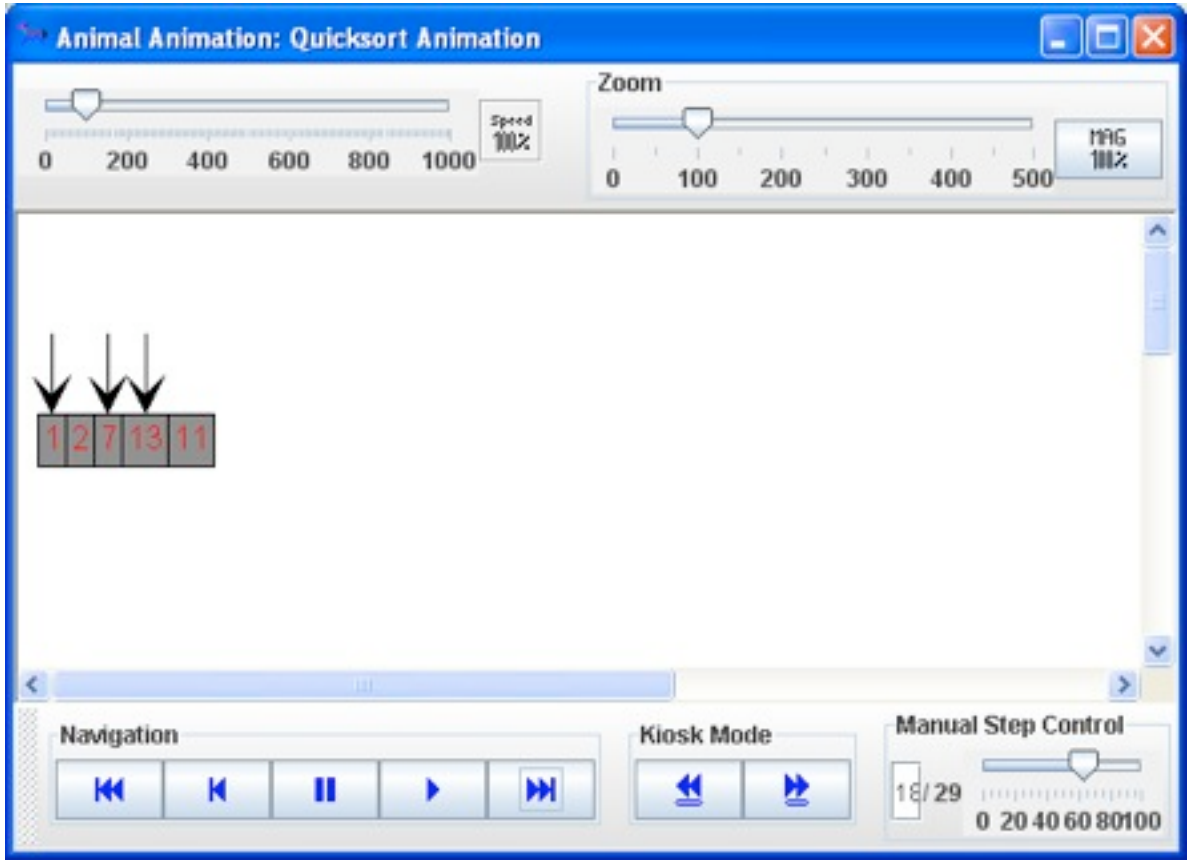
Result of Transformation Step 2 [Part 1]

```
public void selectionSort(int[] inputArray) {
  IntArray array = lang.newIntArray(new Coordinates(10, 30),
                   inputArray, "array", null, arrayProps);
  Timing defaultTiming = new TicksTiming(15);
  lang.nextStep(); // to show array without markers

  ArrayMarker i = lang.newArrayMarker(array, 0, "i", null);
  lang.nextStep(); // to show i marker
  ArrayMarker j = lang.newArrayMarker(array, 0, „j", null);
  lang.nextStep(); // to show j marker
  ArrayMarker min = lang.newArrayMarker(array, 0, "min", null);

  for (; i.getPosition() < array.getLength() - 1;
         i.increment(null, defaultTiming)) {
    lang.nextStep(); // for increment
```

Result of Transformation Step 2 [Part 2]

```
        min.move(i.getPosition(), null, defaultTiming);
        lang.nextStep(); // move min
        for (j.move(i.getPosition() + 1, null, defaultTiming);
             j.getPosition() < array.getLength();
             j.increment(null, defaultTiming)) {
          if (array.getData(j.getPosition())
              < array.getData(min.getPosition()))
            min.move(j.getPosition(), null, defaultTiming);
        } // end for
        array.swap(i.getPosition(), min.getPosition(),
             null, defaultTiming);
        lang.nextStep(); // change step
    }
}
```

Visual Result of Step 2



- Why are there no labels for the array markers?
  - We forgot to do something (see next slide)!

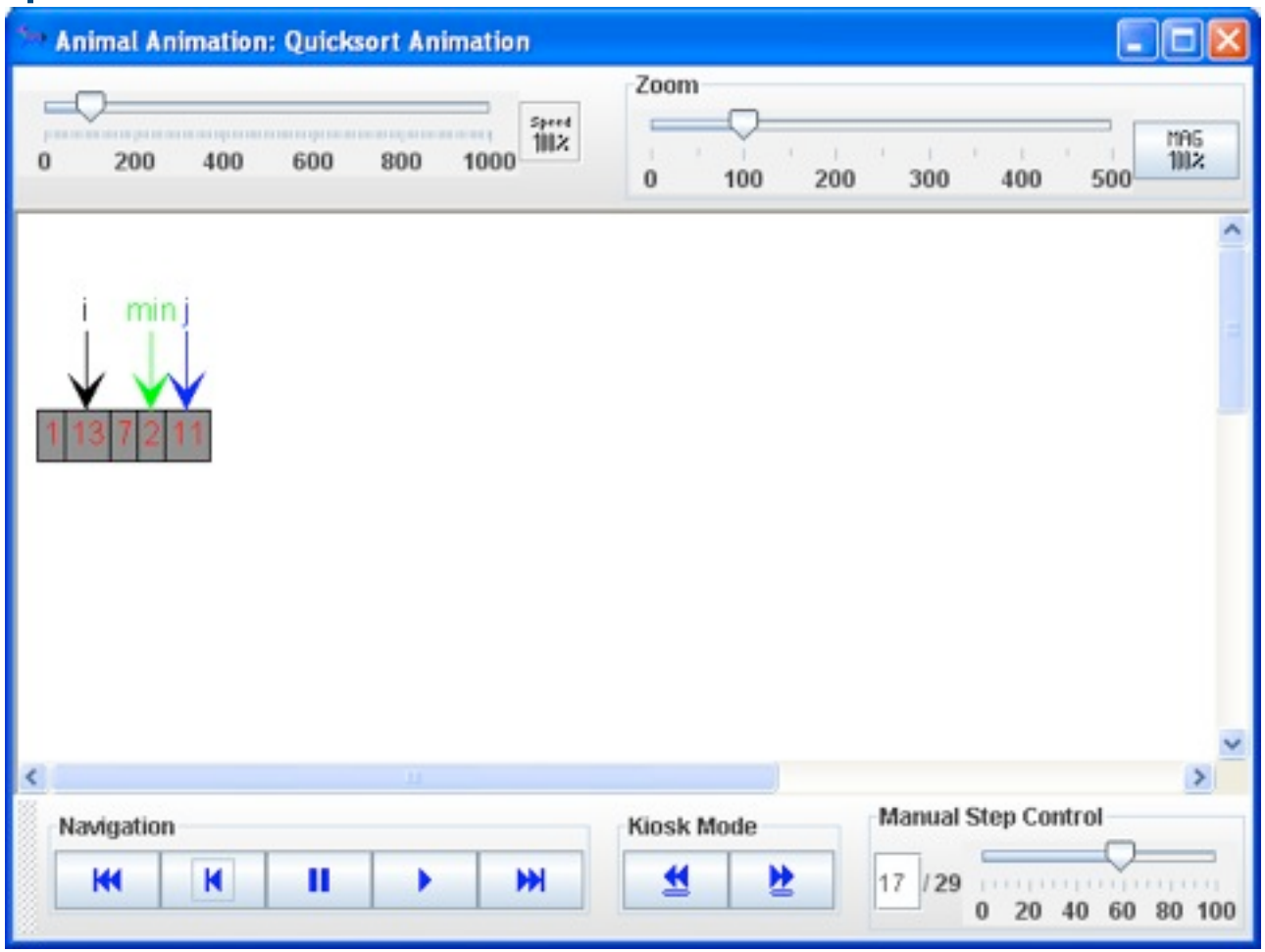
Fixing the missing labels

Add the following to the end of your „init“ method:

```
 // marker for i: black with label „i“
      ami = new ArrayMarkerProperties();
  ami.set(AnimationPropertiesKeys.COLOR_PROPERTY, Color.BLACK);
  ami.set(AnimationPropertiesKeys.LABEL_PROPERTY, "i");
 // marker for j: blue with label „j“
  amj = new ArrayMarkerProperties();
  amj.set(AnimationPropertiesKeys.COLOR_PROPERTY, Color.BLUE);
  amj.set(AnimationPropertiesKeys.LABEL_PROPERTY, "j");
  // marker for min: green with label „min“
  amMin = new ArrayMarkerProperties();
  amMin.set(AnimationPropertiesKeys.COLOR_PROPERTY, Color.GREEN);
  amMin.set(AnimationPropertiesKeys.LABEL_PROPERTY, "min");
}
// declare the marker properties…
private ArrayMarkerProperties ami, amj, amMin; // pass these as additional (last) argument to
 // the generation of the ArrayMarker instances I, j, min :-), e.g.
 // ArrayMarker i = lang.newArrayMarker(array, 0, “i”, null, ami);
```

Visual Result of Step 2 [Part 2]

- The improved version now looks like this:



Step 3: Show the underlying source code

- Add the following method to your class:

```
public void showSourceCode() {
  // first, set the visual properties for the source code
   SourceCodeProperties scProps = new SourceCodeProperties();
   scProps.set(AnimationPropertiesKeys.CONTEXTCOLOR_PROPERTY, Color.BLUE);
   scProps.set(AnimationPropertiesKeys.FONT_PROPERTY,
      new Font("Monospaced", Font.PLAIN, 12));
   scProps.set(AnimationPropertiesKeys.HIGHLIGHTCOLOR_PROPERTY, Color.RED);
   scProps.set(AnimationPropertiesKeys.COLOR_PROPERTY, Color.BLACK);


   // now, create the source code entity
   SourceCode sc = lang.newSourceCode(new Coordinates(40, 140), "sourceCode",
      null, scProps);
   // add a code line
   // parameters: code itself; name (can be null); indentation level; display options
   sc.addCodeLine("public void selectionSort(int[] data) {", null, 0, null);
```

Step 3: Show the underlying source code [II]

```
  sc.addCodeLine("int i, j, minIndex;", null, 1, null);
  sc.addCodeLine("for (i = 0; i < inputArray.length - 1; i++) {", null, 1, null);
  sc.addCodeLine("minIndex = i;", null, 2, null);
  sc.addCodeLine("for (j = i + 1; j < inputArray.length; j++)", null, 2, null);
  sc.addCodeLine("if (inputArray[j] < inputArray[minIndex])", null, 3, null);
  sc.addCodeLine("minIndex = j;", null, 4, null);
  sc.addCodeLine("swap(inputArray, i, j); // swap", null, 2, null);
  sc.addCodeLine("}", null, 1, null);
  sc.addCodeLine("}", null, 0, null);
}
private SourceCode sc; // to ensure it is visible outside the method…
```

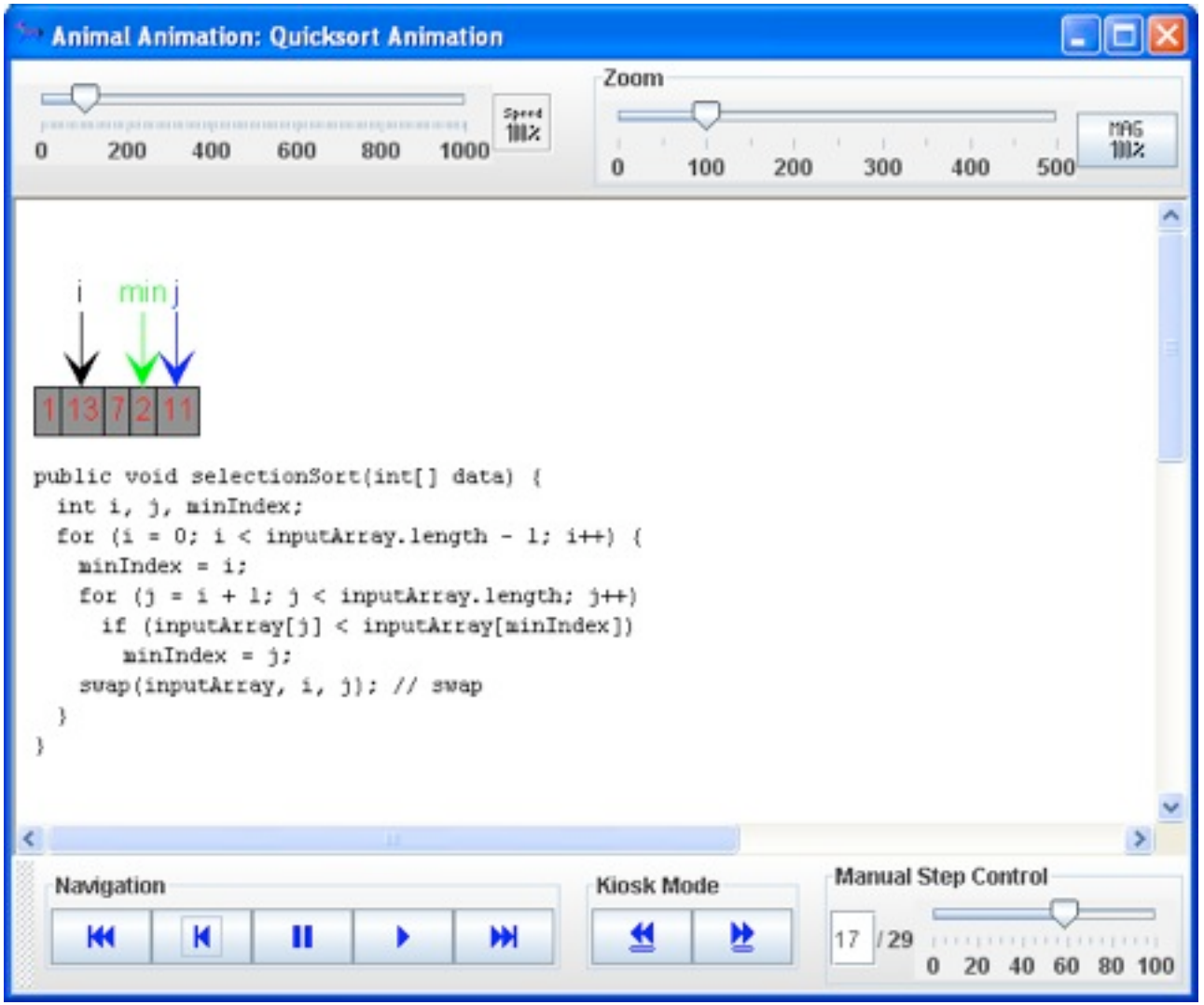
- While this is much typing work, the code itself is easy
  - Method declaration and final brace are at “top level” – indentation 0
  - The declarations of i, j, minIndex, outer loop and brace are at level 1
  - The reset of minIndex, inner *for* and the *swap* are at level 2
  - The *if* is at level 3, and its body is at level 4

Step 3: Show the underlying source code [III]

- Make sure your new method is called inside your code
  - After the initial IntArray is generated, and before the *lang.nextStep()*

```
public void selectionSort(int[] inputArray) {
  IntArray array = lang.newIntArray(new Coordinates(10, 30),
                   inputArray, "array", null, arrayProps);
  Timing defaultTiming = new TicksTiming(15);
  showSourceCode(); // show the source code
  lang.nextStep(); // to show array without markers
  // ... code continues as before...
```

Visual Result of Step 3



Step 4: Highlight the current code lines

- We now want to highlight the current code lines
- We can use *sc.highlight(lineNr)* and *sc.unhighlight(lineNr)*
- For the lazy: use *sc.toggleHighlight(oldLineNr, newLineNr)*
- We need to find the right places to do this
- The code is shown on the next two slides
  - It just keeps growing… ☺
  - Only the new elements will be highlighted
- Note: ensuring the right line is highlighted can be tricky
  - If we enter the *if*, we will highlight "min=", then unhighlight "min="
  - If we do *not* enter the if, we should not unhighlight "min="
    - But we have to unhighlight the *if*!

Results of Step 4

```
public void selectionSort(int[] arrayContents) {
    IntArray array = lang.newIntArray(new Coordinates(10, 100),
        arrayContents, "array", null, arrayProps);
    Timing defaultTiming = new TicksTiming(15);
    showSourceCode();
    sc.highlight(0); // method head
    lang.nextStep(); // to show array without markers
    sc.toggleHighlight(0, 1); // jump from header to int...
    ArrayMarker i = lang.newArrayMarker(array, 0, "i", null, ami);
    lang.nextStep(); // to show i marker
    ArrayMarker j = lang.newArrayMarker(array, 0, "j", null, amj);
    lang.nextStep(); // to show j marker
    ArrayMarker min = lang.newArrayMarker(array, 0, "min", null, amMin);


    sc.unhighlight(1); // unhighlight the declarations
```

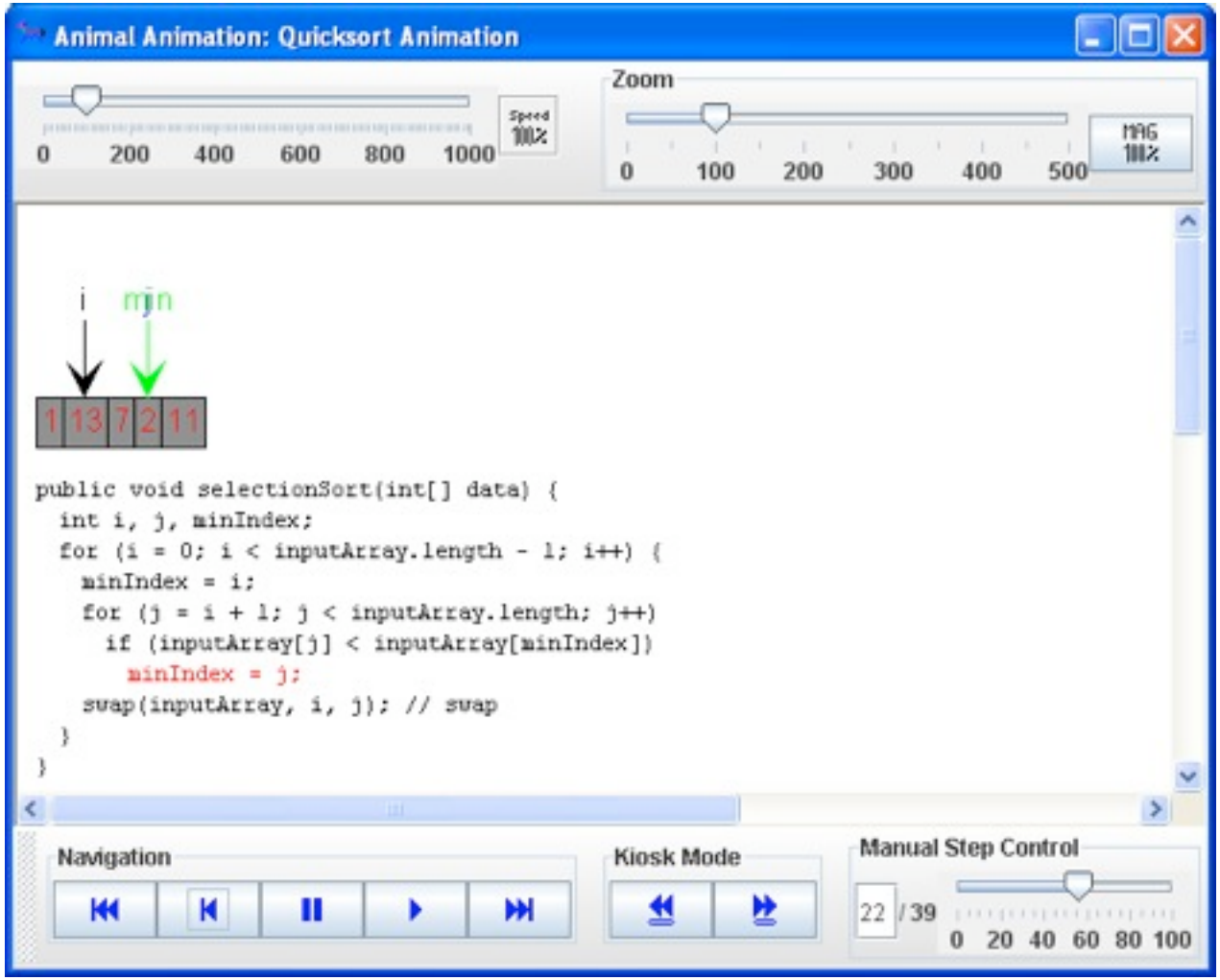
Results of Step 4 [II]

```
for (; i.getPosition() < array.getLength() - 1;
       i.increment(null, defaultTiming)) {
  sc.highlight(2); // now at line 2 (for loop)

  lang.nextStep(); // for increment
  sc.toggleHighlight(2, 3); // for --> min =

  min.move(i.getPosition(), null,defaultTiming);
  lang.nextStep(); // move min
  sc.unhighlight(3); // min= done
  for (j.move(i.getPosition() + 1, null, defaultTiming);
       j.getPosition() < array.getLength();
       j.increment(null, defaultTiming)) {
    sc.highlight(4); // for j...
    lang.nextStep(); // wait for the update of j
    sc.toggleHighlight(4, 5); // for loop entered, change to „if“
```

Results of Step 4 [III]

```
if (array.getData(j.getPosition()) < array.getData(min.getPosition())) {
  lang.nextStep();
  sc.toggleHighlight(5, 6); // change from „if“ to min =
  min.move(j.getPosition(), null, defaultTiming);
  lang.nextStep(); // wait for update of j
  sc.unhighlight(6); // remove highlight on „min="
  } else { // added for highlighting to ensure „if“ is no longer highlighted
    lang.nextStep();
    sc.unhighlight(5); // unhighlight “if”
  }
 }
 sc.highlight(7); // highlight swap
 array.swap(i.getPosition(), min.getPosition(), null, defaultTiming);
 lang.nextStep(); // change step
 sc.unhighlight(7); // unhighlight swap
 }
}
```

Visual Result of Step 4



Add-Ons

- If you wanted to have a really nice animation, you could…
  - Add interactive prediction
  - Add a header (which is currently missing)
  - Add counters
- However, what we have done so far is probably “enough”!

Adding Your Materials to ANIMAL

- ANIMAL has a built-in set of animation content generators
  - Use *File* → *Generate* to access them
- How can you add your new generator to this?
- It's not as difficult as you may think…

1. Declare your class to *implement Generator*
   - You should also do an *import generator.Generator;*
   - Let Eclipse auto-complete the missing methods
2. Implement the missing methods
   - Shown on the next slide for the "more obvious ones" ☺
3. Adapt the input data to data provided by the user
   - Shown in a short moment…

Implementing the Missing Methods

```
public String getCodeExample() {
  return "Straightforward SelectionSort Algorithm"; // to give readers an impression
}
public Locale getContentLocale() {
  return Locale.US; // US-English
}
public String getDescription() {
  return "Animates SelectionSort with Source Code + Highlighting"; // description
}
public String getFileExtension() {
  return ".asu"; // file extension for „AnimalScript, uncompressed“
}
public GeneratorType getGeneratorType() {
  return new GeneratorType(GeneratorType.GENERATOR_TYPE_SORT); // this is about sorting!
}
public String getName() {
  return "SelectionSortDemo"; // the title to be displayed
}
```

Adapting the Input Data

- Implement the missing „generate“ method as follows:

```
public String generate(AnimationPropertiesContainer props,
                       Hashtable<String, Object> primitives) {
  init(); // ensure all properties are set up :-)
  int[] arrayData = (int[])primitives.get("array");
  // adapt the COLOR to whatever the user chose
  // you could do this for all properties if you wanted to...
  arrayProps.set(AnimationPropertiesKeys.COLOR_PROPERTY,
     props.get("array", AnimationPropertiesKeys.COLOR_PROPERTY));

  // call the selection sort method
  selectionSort(arrayData);
  return lang.toString();
}
```

- This will extract the array the user has generated and pass it to *selectionSort*
- It also adapts one color setting to show how this is done.

Creating the Property File

- To create an XML definition file for your new generator:
  - Run java -cp Animal-x.jar generator.PropertiesGUI
    - Replace "-x" with the correct version number!
  - Create a "New Primitive", type int[], named "array"
  - Create the following "New Properties":
    - **`array --`** type ArrayProperties
    - **`i --`** type ArrayMarkerProperties
    - **`j --`** type ArrayMarkerProperties
    - **`min --`** type ArrayMarkerProperties
    - **`sourceCode --`** type SourceCodeProperties
  - Confirm your choice
    - potential I18N bug: use "Bestätigen" if there is no "Confirm" button
  - Adjust the settings if you like, then save the file
  - It should go in a folder "generatorImplementations"

Last Steps

- Re-Package your class to ***generatorImplementations***
- Your directory ***generatorImplementations*** should contain:
  - SelectionSortDemo.java // your Java source
  - SelectionSortDemo.class // compiled Java class
  - SelectionSortDemo.ptm // Generator specification
- Run ***java -cp Animal-x.jar;. animal.main.Animal***
  - If generatorImplementations is not in your current directory, adapt the path
- Select "File → Generate" and then "Sorting Algorithms"
- Search for "SelectionSortDemo" on the left, then "Confirm"
- Adapt the input data, Confirm
- Instead of typing in a file name, you can also press Confirm to directly see the animation ☺

Restrictions and "To Dos"

- Some things still need to be done
  - For me, not for you!
- This includes…:
  - A sensible way to include support for interactive prediction
  - A better documentation for some of the features
  - A more up-to-date documentation on the web
- Any comments you have are welcome!
  - Just send them to roessling@acm.org
  - But please do not expect an immediate bug fix ☺
- Note: Animal-2.3.20+ is a developer version
  - It may show a couple of output messages that are *not* relevant to you

Summary

- This presentation has introduced the ALGOANIM API
- As you can tell from the longer example, content generation can be time-consuming
- But it gets much faster the "second time around"
- For more features etc., please consult the documentation
- You can find it at http://www.algoanim.info/Animal2/
- There you will also find the ANIMAL downloads