What is VRL-Studio?

Overview

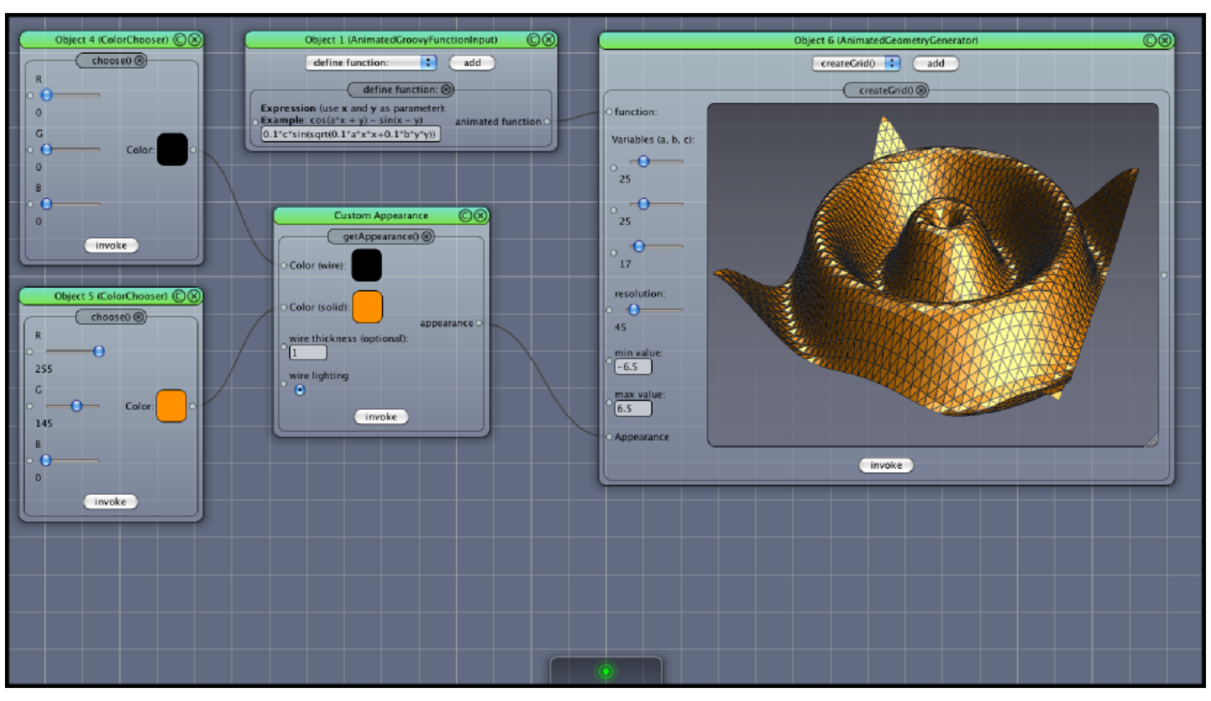
VRL-Studio is an innovative Integrated Development Environment (IDE) based on the Java Platform that combines both visual and text-based programming. In contrast to many other Development Environments VRL-Studio Projects are fully functional programs that are developed at runtime. This makes it an ideal Environment for rapid prototyping, teaching and experimentation.

A VRL-Project is basically a Java Library (.jar-File) that contains visually defined workflows, source code, compiled classes and optionally resources. In addition to that, each project is under revision control and contains its complete history.

VRL-Studio is based on the Visual Reflection Library (VRL). VRL enables declarative and fully automated creation of graphical user interfaces from Java objects. Therefore, it uses the information accessible via the Java Reflection API.

In addition to the automatic generation of user interfaces VRL-Studio supports the definition of *data-flow* and *control-flow*.

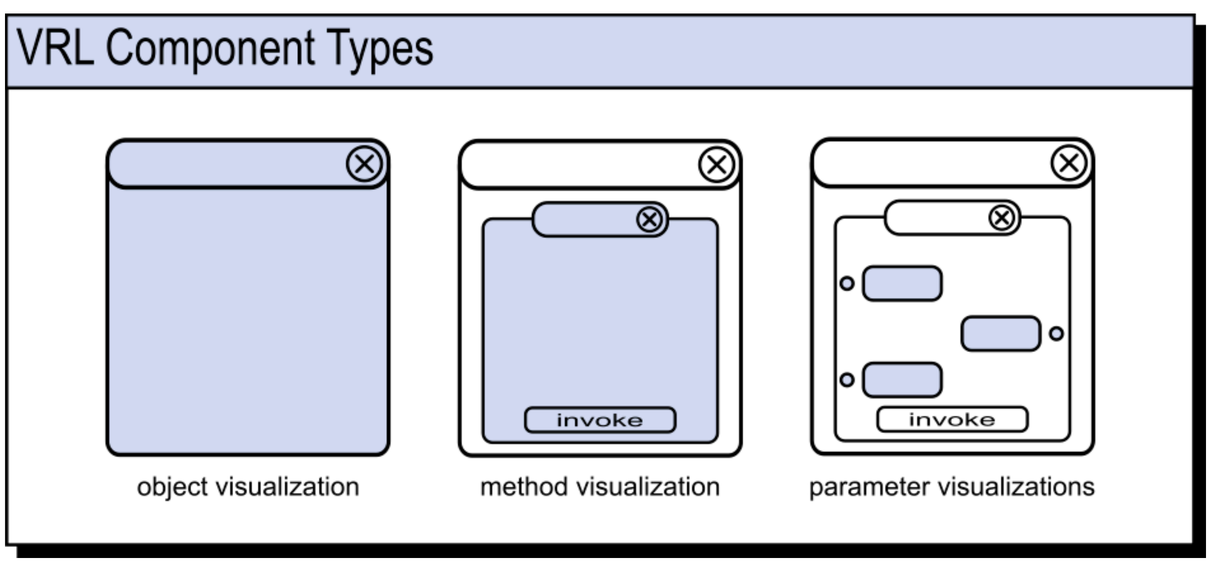
Figure Function Plotter demonstrates the capabilities of VRL.



Function Plotter

VRL & Declarative UI Development

VRL provides three types of visual components (see Figure Component Types) . An object representation is a container, comparable to a program window that can group several child components. A method representation is a container component inside an object representation. It can also group child components and provides elements for calling the represented method. To represent variable data VRL provides type representations. In most cases they allow interaction with the visualized data.



Component Types

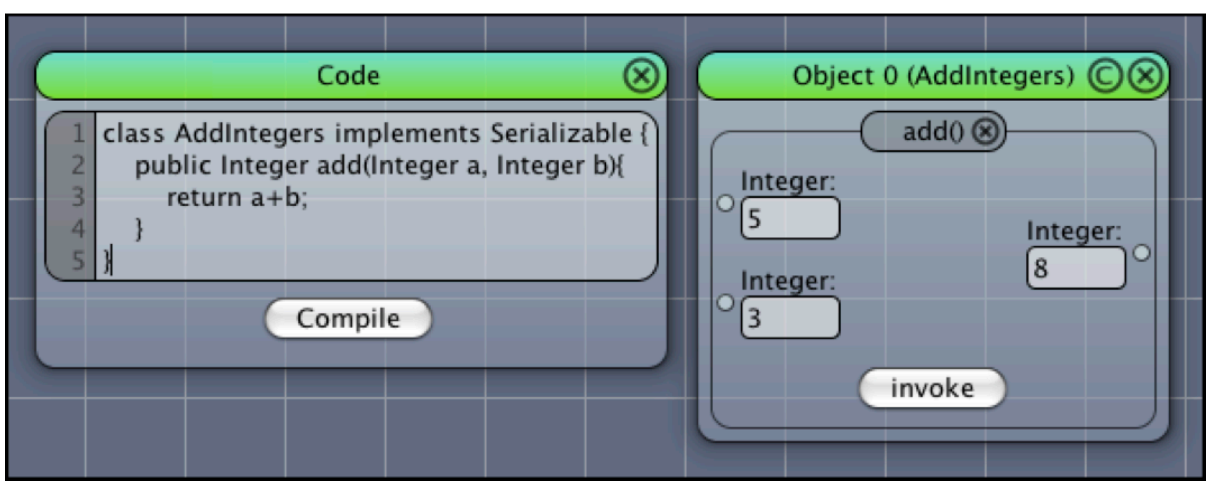
To create a graphical user interface in VRL it is only necessary to specify its functionality.

Example

The following code defines a simple Java/Groovy class that provides a method that adds two integer numbers.

```
class AddIntegers {
    public Integer add( Integer a , Integer b) {
      return a+b;
    }
}
```

To create a VRL interface for this class no additional code is required.



```
class AddIntegers implements Serializable {
    public Integer add(Integer a, Integer b){
        return a+b;
    }
}
```

Integer: 5

Integer: 3

Integer: 8

Add Integers

Figure Add Integers shows how to define the interface in VRL-Studio.

Manually creating the user interface with Java is much more complicated. Equivalent Java code [1]:

```
public class Main {
    public static void main( String [] args ) {
        javax.swing.JFrame frame = new javax.swing.JFrame(”Add Integers”);
        frame.setLayout(new java.awt.GridLayout());
        final javax.swing.JTextField input1 = new javax.swing.JTextField();
        final javax.swing.JTextField input2 = new javax.swing.JTextField();
        final javax.swing.JTextField output = new javax.swing.JTextField();
        frame.add(new javax.swing.JLabel(”Integer”));
        frame.add( input1 );
        frame.add(new javax.swing.JLabel(”Integer”));
        frame.add( input2 );
        frame.add(new javax.swing.JLabel(”Result”));
        frame.add( output );
        javax.swing.JButton btn = new javax.swing.JButton(”invoke”);
        btn.addActionListener (new java.awt.event.ActionListener() {
            public void actionPerformed ( java.awt.event.ActionEvent e ) {
                output.setText (new Integer (new Integer( input1.getText()) +
                new Integer(input2.getText())).toString());
            }
        });
        frame.add( btn ) ;
        frame.pack( ) ;
        frame.setVisible (true );
    }
}
```

For real examples look at the VRL tutorial template: `File->New Project from Template->VRL-Tutorial-1`.

1. not necessarily the shortest possible implementation ↩