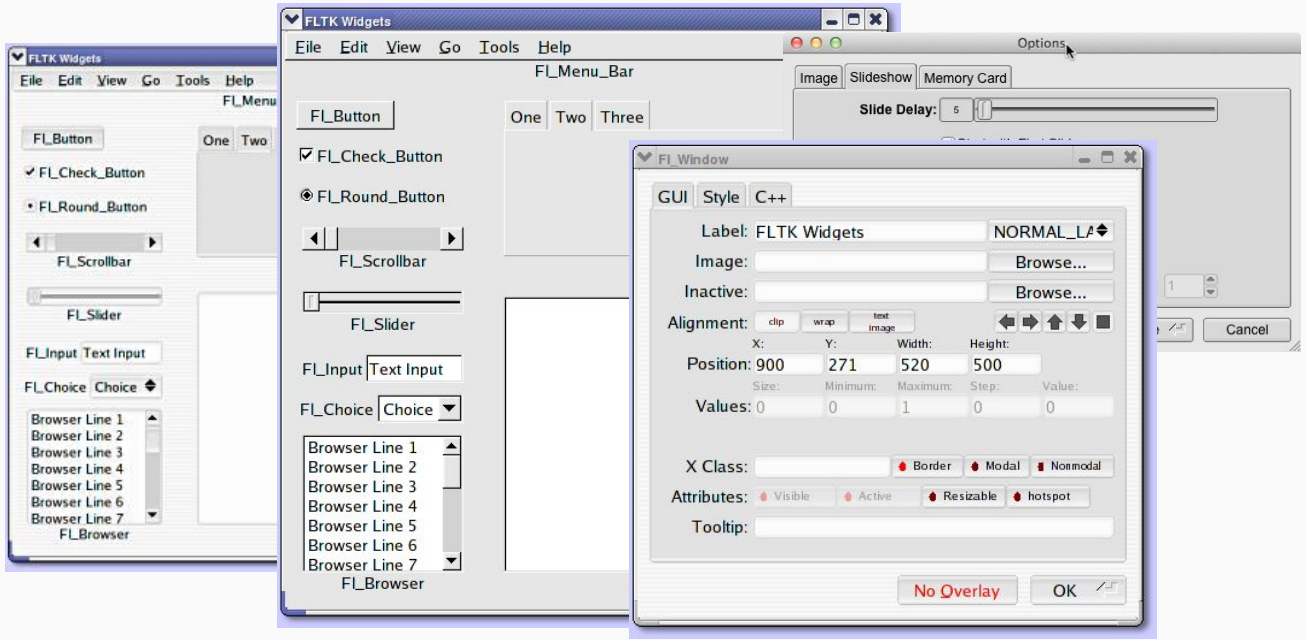
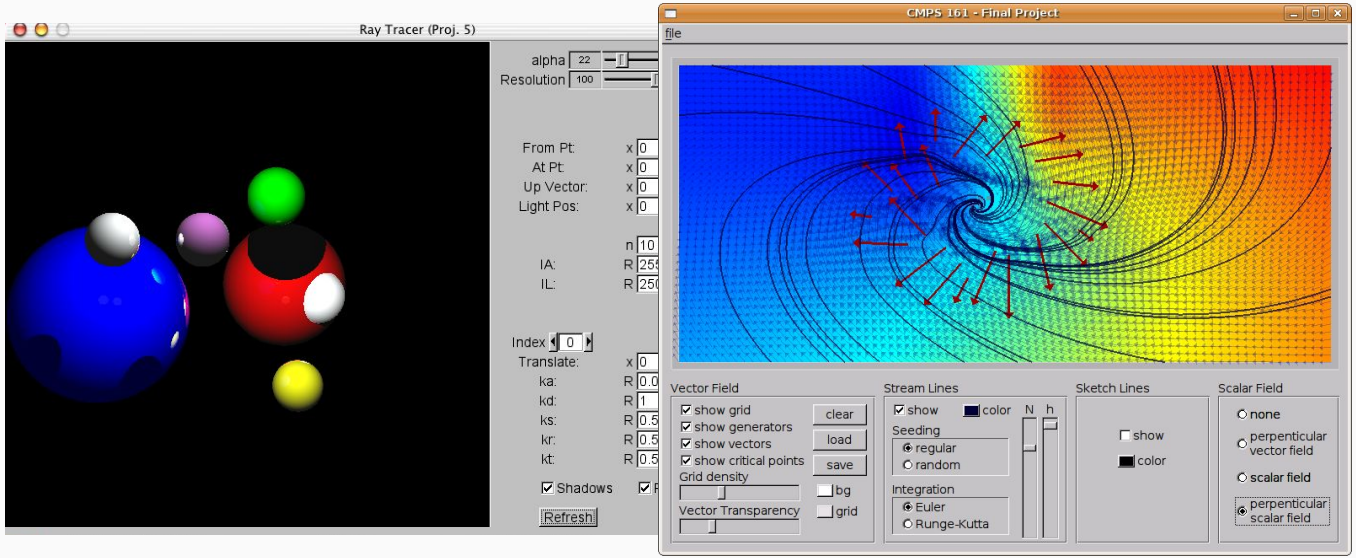
How I built a native Dart VM extension for 

by Herman Bergwerf

What is FLTK?


FLTK Widgets
File Edit View Go Tools Help
FL_Button
FL_Check_Button
FL_Round_Button
FL_Scrollbar
FL_Slider
FL_Input Text Input
FL_Choice Choice
Browser Line 1
Browser Line 2
Browser Line 3
Browser Line 4
Browser Line 5
Browser Line 6
Browser Line 7
FL_Browser
One Two Three
Options
Image Slideshow Memory Card
Slide Delay:
Cancel
FL_Window
GUI Style C++
Label: FLTK Widgets
Image:
Inactive:
Alignment:
Position: 900 271 520 500
Values:
X Class:
Attributes:
Tooltip:
Border Modal Nonmodal
Visible Active Resizable hotspot
No Overlay
OK

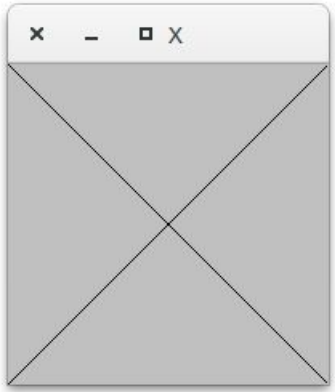
Great with OpenGL



Why FLTK?

- Compiles the first time you try!
- Simple class structure
- Straightforward API
- I’m already familiar with FLTK

```cpp
class XWidget : public Fl_Widget {
 public:
  /// Constuctor
  XWidget(int x, int y, int w, int h) : Fl_Widget(x, y, w, h, 0) {}

  /// Draws the lines
  void draw() {
    fl_color(FL_BLACK);
    int x1 = x(), y1 = y();
    int x2 = x() + w() - 1, y2 = y() + h() - 1;
    fl_line(x1, y1, x2, y2);
    fl_line(x1, y2, x2, y1);
  }
};

int main() {
  auto window = new Fl_Double_Window(200, 200, "X");
  auto x = new XWidget(0, 0, window -> w(), window -> h());
  window -> resizable(x);
  window -> show();
  return Fl::run();
}
```



FLTK-Dart example

C++

```cpp
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Box.H>
#include <FL/fl_draw.H>

int main(int argc, char **argv) {
  Fl::scheme("gleam");
  auto window = new Fl_Window(350, 180, "FLTK");
  auto box = new Fl_Box(20, 40, 310, 100, "Hello, World!");
  box -> box(FL_UP_BOX);
  box -> labelsize(36);
  box -> labelfont(FL_BOLD + FL_ITALIC);
  box -> labeltype(FL_SHADOW_LABEL);
  box -> labelcolor(FL_YELLOW);
  box -> color(FL_RED);
  window -> end();
  window -> show(argc, argv);
  return Fl::run();
}
```

Dart

```dart
import 'package:color/color.dart';
import 'package:fltk/fltk.dart' as fl;

int main() {
  fl.scheme = 'gleam';
  var window = new fl.Window(350, 180, 'FLTK');
  var box = new fl.Box(20, 40, 310, 100, 'Hello, World!');
  box.box = fl.UP_BOX;
  box.labelsize = 36;
  box.labelfont = fl.BOLD + fl.ITALIC;
  box.labeltype = fl.SHADOW_LABEL;
  box.labelcolor = fl.YELLOW;
  box.color = fl.toColor(new HexColor('#ff0000'));
  window.end();
  window.show();

  return fl.run();
}
```

FLTK-Dart example

C++ *Dart*

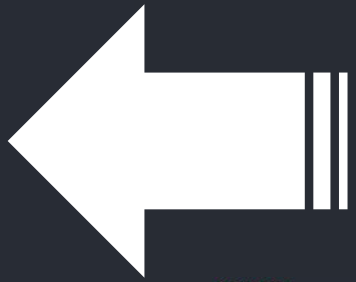
```cpp
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Box.H>
#include <FL/fl_draw.H>

int main(int argc, char **argv)
  Fl::scheme("gleam");
  auto window = new Fl_Window(35
  auto box = new Fl_Box(20, 40,
  box -> box(FL_UP_BOX);
  box -> labelsize(36);
  box -> labelfont(FL_BOLD + FL_
  box -> labeltype(FL_SHADOW_LAB
  box -> labelcolor(FL_YELLOW);
  box -> color(FL_RED);
  window -> end();
  window -> show(argc, argv);
  return Fl::run();
}
```


```dart
/color.dart';
fltk.dart' as fl;

;
.Window(350, 180, 'FLTK');
x(20, 40, 310, 100, 'Hello, World!');
;

BOLD + fl.ITALIC;
SHADOW_LABEL;
.YELLOW;
lor(new HexColor('#ff0000'));
window.end();
window.show();

return fl.run();
}
```

Dart	C++	generated C++
Objects for all widgets with native calls	Extension setup	Function wrappers
Dart flavoured Streams for events	Dart API utilities *(like handling Unt8List data blocks)*	FLTK widget wrappers
Integrations with third party libraries	Special cases *(Cairo graphics, Fl_Text_Buffer callbacks…)*	Method wrappers *(stores class instance pointers in Dart handle)*

Code generator

```
void Fl_Labeltype_labeltype(Dart_NativeArguments arguments) {
  Fl_Widget_Wrapper *_ref;
  Dart_EnterScope();
  _ref = (Fl_Widget_Wrapper*)getptr(arguments, 0);
  int64_t _tmp = static_cast<int64_t>(_ref -> labeltype());
  Dart_Handle _ret = Dart_NewInteger(_tmp);
  Dart_SetReturnValue(arguments, _ret);
  Dart_ExitScope();
}

void void_labeltype(Dart_NativeArguments arguments) {
  Fl_Widget_Wrapper *_ref;
  int64_t type;
  Dart_EnterScope();
  _ref = (Fl_Widget_Wrapper*)getptr(arguments, 0);
  HandleError(Dart_IntegerToInt64(getarg(arguments, 1), &type));
  _ref -> labeltype(static_cast<Fl_Labeltype>(type));
  Dart_Handle _ret = Dart_Null();
  Dart_SetReturnValue(arguments, _ret);
  Dart_ExitScope();
}
```

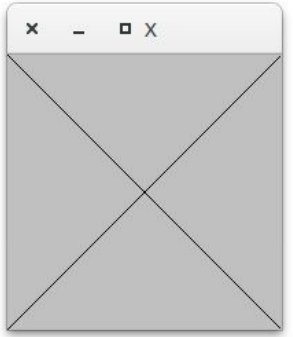


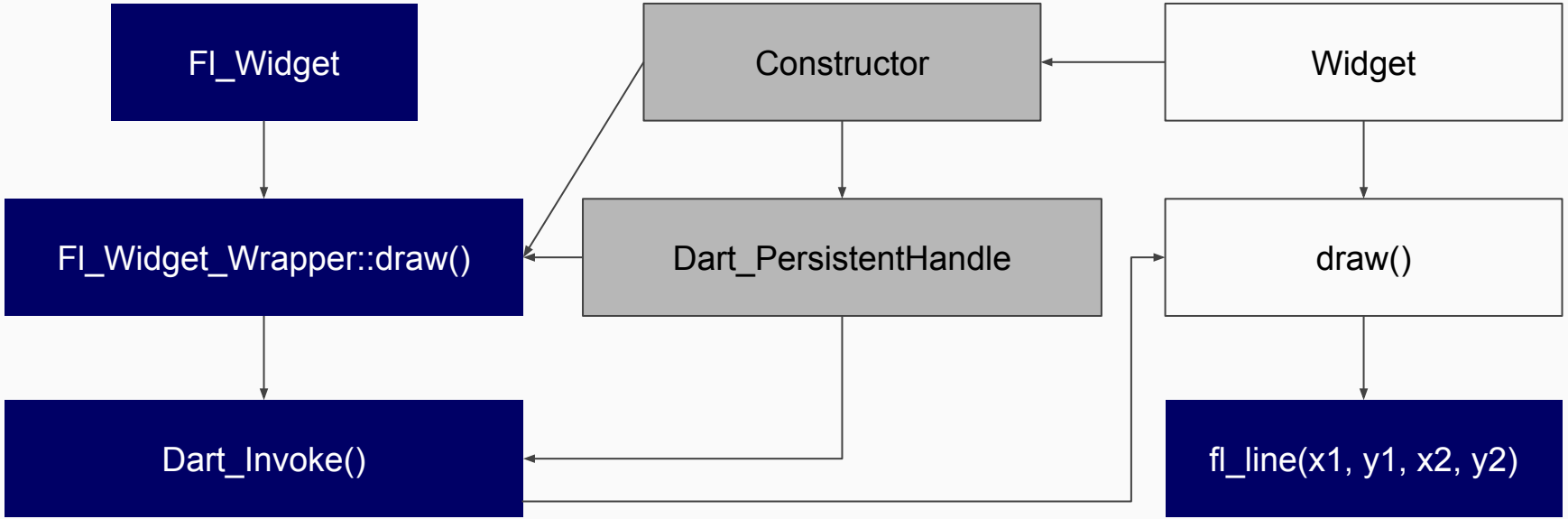
```
cname: Fl_Widget
dartname: Widget

constructors:
- void (int x, int y, int w, int h, String l)

methods:
- int x()
- int y()
- int w()
- int h()

- void activate()
- void deactivate()
- int active()
- int active_r()
- void show()
- void hide()
- void redraw()
- int visible()
- int visible_r()

- String label()
- void label(String text)
- Fl_Font labelfont()
- void labelfont(Fl_Font f)
- Fl_Fontsize labelsize()
- void labelsize(Fl_Fontsize pixels)
- Fl_Labeltype labeltype()
- void labeltype(Fl_Labeltype type)
- Fl_Color labelcolor()
- void labelcolor(Fl_Color c)

enums:
- Fl::Fl_Option
- Fl_Labeltype
- Fl_Boxtype
- Fl_Mode

typedefs:
  Fl_Color: int64_t      #
  Fl_Align: int64_t      #
  Fl_Fontsize: int64_t   #
  Fl_Font: int64_t       #
```

Third party integrations

`dartgl`:	OpenGL extension on Pub
`cairodart`:	Cairo graphics extension on Pub
`image`:	Image data decoding, can be used to provide widget images
`color`:	Color library

Drawing the X in Dart





Dart Streams, yay!

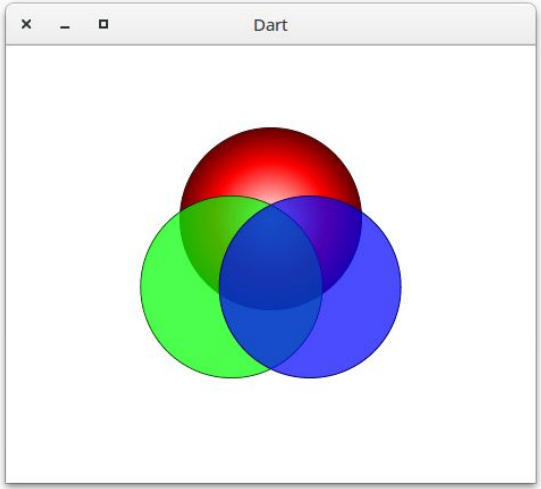
```
var window = new fl.Window(300, 200, 'Click the button...');
var button = new fl.Button(0, 0, window.w(), window.h(), 'ON');
button.onCallback
    .listen((_) => button.label = button.label == 'ON' ? 'OFF' : 'ON');
window.end();
window.show();
```

```
editor.textBuffer.onModify.listen((data) {
  if (data.nInserted != 0 || data.nDeleted != 0) {
    updateEditorSyntax();
  }
});
```

Cool apps in FLTK Dart!

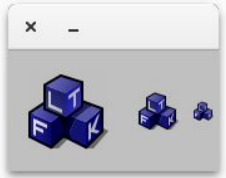
```dart
import 'dart:io';

import 'package:image/image.dart';
import 'package:fltk/fltk.dart' as fl;

int main() {
  var window = new fl.DoubleWindow(64, 64);
  var button = new fl.Box(0, 0, 64, 64);
  button.image = decodeImage(new File('example/image.png').readAsBytesSync());
  window.end();
  window.show();
  return fl.run();
}
```

```dart
import 'package:cairodart/cairodart.dart';
import 'package:fltk/fltk.dart' as fl;

int main() {
  var window = new fl.CairoWindow(300, 300, 'Dart');

  window.resizable = window;
  window.color = fl.WHITE;

  // Set cairo draw callback.
  window.drawCallback = (_, Context ctx) {
    ctx.lineWidth = 1.0;

    // Final variables
    final r1 = 80, r2 = 40, cx = window.w() / 2, cy = window.h() / 2;

    // Red circle
    var a = PI / 2;
    final redGradient = new RadialGradient(0, 0, 0, 0, 0, 100);
    redGradient.addColorStop(new ColorStop(new Color.rgb(1, 1, 1), 0.0));
    redGradient.addColorStop(new ColorStop(new Color.rgb(1, 0, 0), 0.5));
    redGradient.addColorStop(new ColorStop(new Color.rgb(0, 0, 0), 1.0));
```

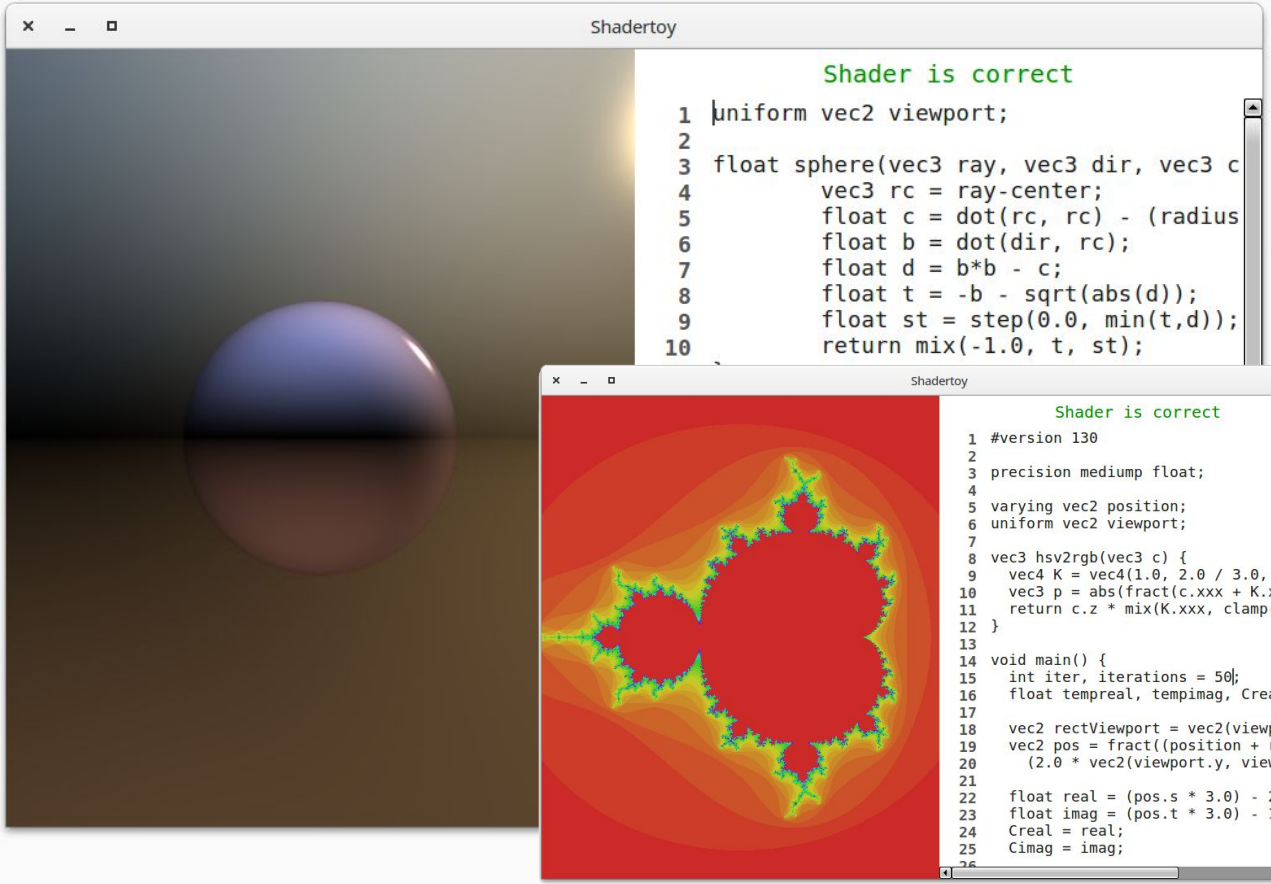


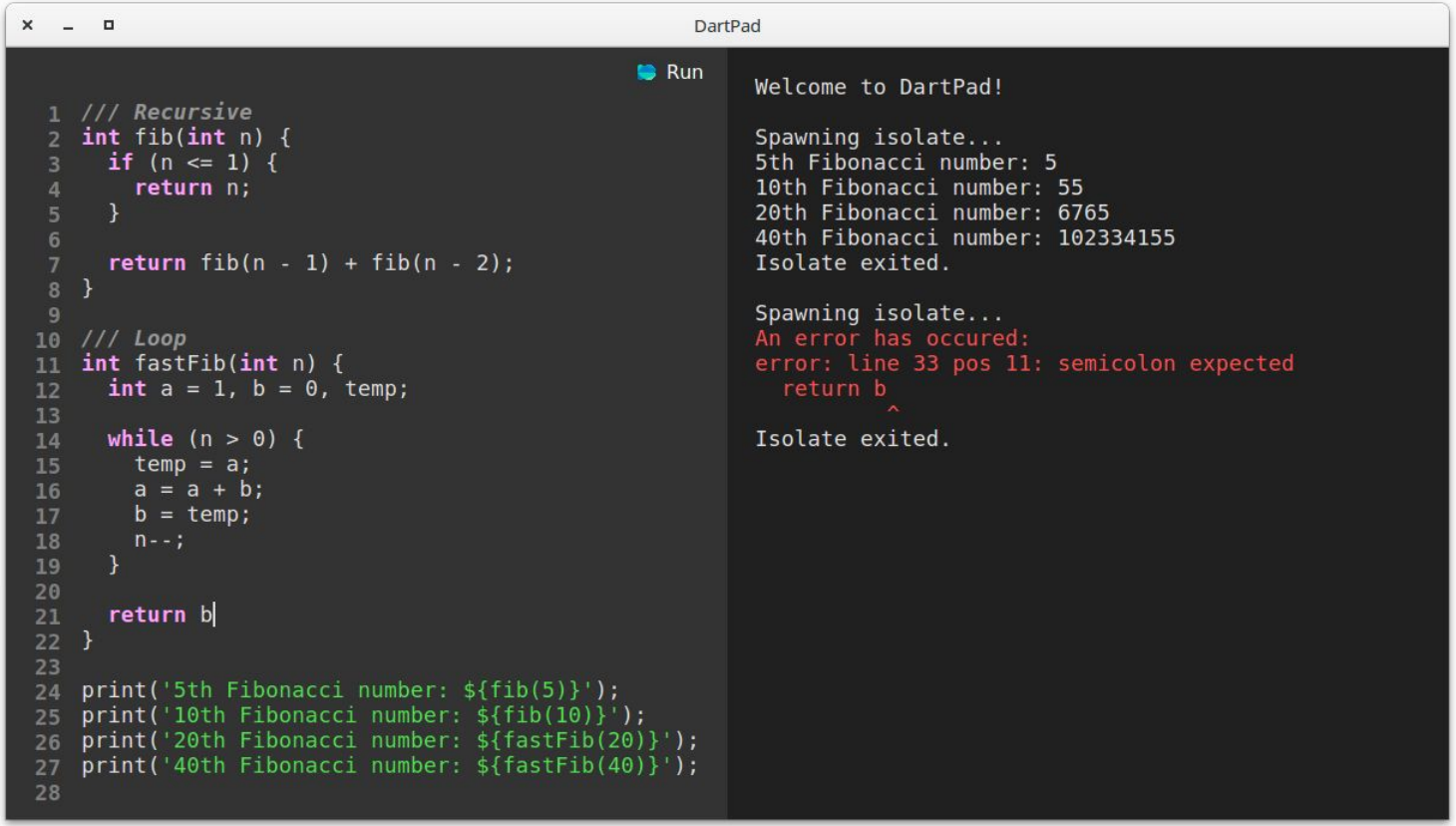
I wrote some Dart code to parse HVIF files and render them in Cairo!

Bitmap	SVG	HVIF
1,024 byte + 256 byte	7,192 byte	768 byte



```
24      surface.addHandler((event, data) {
25        switch (event) {
26          case fl.Event.PUSH:
27            // Update position.
28            x = data.x;
29            y = data.y;
30            xo = x;
31            yo = y;
32            return true;
33
34          case fl.Event.DRAG:
35            // Draw line segment
36            ctx
37              ..moveTo(xo, yo)
38              ..lineTo(x, y)
39              ..lineTo(data.x, data.y)
40              ..setSourceRgb(x / surface.w(), y / surface.h(), blue)
41              ..stroke();
42            surface.redraw();
```


Shadertoy
Shader is correct
1 uniform vec2 viewport;
2
3 float sphere(vec3 ray, vec3 dir, vec3 c
4 vec3 rc = ray-center;
5 float c = dot(rc, rc) - (radius
6 float b = dot(dir, rc);
7 float d = b*b - c;
8 float t = -b - sqrt(abs(d));
9 float st = step(0.0, min(t,d));
10 return mix(-1.0, t, st);
Shadertoy
Shader is correct
1 #version 130
2
3 precision mediump float;
4
5 varying vec2 position;
6 uniform vec2 viewport;
7
8 vec3 hsv2rgb(vec3 c) {
9 vec4 K = vec4(1.0, 2.0 / 3.0, 1.0 /
10 vec3 p = abs(fract(c.xxx + K.xyz) *
11 return c.z * mix(K.xxx, clamp(p - K
12 }
13
14 void main() {
15 int iter, iterations = 50;
16 float tempreal, tempimag, Creal, Ci
17
18 vec2 rectViewport = vec2(viewport.y
19 vec2 pos = fract((position + rectVi
20 (2.0 * vec2(viewport.y, viewport.
21
22 float real = (pos.s * 3.0) - 2.0;
23 float imag = (pos.t * 3.0) - 1.5;
24 Creal = real;
25 Cimag = imag;


DartPad
Run
/// Recursive
int fib(int n) {
if (n <= 1) {
return n;
}
return fib(n - 1) + fib(n - 2);
}
/// Loop
int fastFib(int n) {
int a = 1, b = 0, temp;
while (n > 0) {
temp = a;
a = a + b;
b = temp;
n--;
}
return b
}
print('5th Fibonacci number: ${fib(5)}');
print('10th Fibonacci number: ${fib(10)}');
print('20th Fibonacci number: ${fastFib(20)}');
print('40th Fibonacci number: ${fastFib(40)}');
Welcome to DartPad!
Spawning isolate...
5th Fibonacci number: 5
10th Fibonacci number: 55
20th Fibonacci number: 6765
40th Fibonacci number: 102334155
Isolate exited.
Spawning isolate...
An error has occured:
error: line 33 pos 11: semicolon expected
return b
^
Isolate exited.

Can you use this today?

Not really…

- No compile scripts for Windows or macOS
- Only a small subset of FLTK widgets are implemented

Source code:

github.com/hermanbergwerf/fltk-dart

pub.dartlang.org: **fltk**