Lazarus: Project Fresnel

Michaël Van Canneyt

December 26, 2022

Abstract

Part of the Lazarus roadmap is a new initiative called Project Fresnel: a replacement for the LCL (Lazarus Component Library). In this article we explain the goals of this new project.

1 Introduction

The LCL - as its Delphi counterpart VCL - is old. The foundations of this UI framework were laid 20 years ago, when it was conceived as a free version of the VCL - the UI framework used in Delphi. Its design reflects the UI principles of that time, and more precisely, the Windows UI toolkit of that time, the Win32 API, sported for the first time in Windows 95. One could say therefore that the LCL is a 27-year old technology. That one can take 20 year old code and still create a working application today is thanks to the great efforts to keep things backwards compatible.

But meanwhile, the world has moved on.

The internet has taken the world by storm, and the UI development in the browser is based on 2 standards: HTML (the DOM) and CSS. CSS offers a lot of possibilities in terms of UI customization, and the possibilities of CSS still grow.

The appearance of other devices created challenges for the UI developers: while in the 90-ies, the PC was the dominant platform, today this is no longer the case: mobile phones are ubiquitous, and they come in all shapes and colors. The old desktop PC is no longer the dominant form factor.

CSS has enabled web developers to handle these various form factors: it is difficult to imagine a modern website which has not been designed with responsiveness in mind.

In response to these new platforms, Firemonkey was introduced in Delphi: an alternative to the VCL which can be used on multiple platforms. It has styles, which can be used to cater for the different form factors, and can be seen as a replacement for CSS. Firemonkey can handle different backends, for example SKIA can be used as a backend or OpenGL. But at its core, firemonkey still uses the positioning paradigm used in the VCL, and the styles mechanism uses a format resembling the DFM format.

2 Introducing Project Fresnel

Instead of introducing a new format to update the UI layer of Lazarus, it seems like a logical choice to reuse a successful and standardized mechanism: CSS.

The paradigms of the VCL must be completely thrown overboard: this means no more obligatory positioning, no more obligatory `Top`, `Left`, `Width` and `Height` properties. Instead, all matters UI and layout must be delegated to CSS properties.

In short: The aim of Project Fresnel is to make a set of controls (or widgets) which are guided by CSS. These controls will obey the following architectural principles:

- The widgets must be streamable: That means the widgets are descendents of `TComponent` and must use published properties. This requirement means a form file can be made, and the lazarus IDE can be used to manipulate the widgets.
- The layout and looks are determined completely by CSS. This means every widget has a property that represents the CSS to be used. Additional properties for manipulation in an Object Inspector can of course be introduced.
- Multiple drawing back-ends must be supported. Since Lazarus is multi-platform, it goes without saying that the various platforms will provide a specialized back-end, much as the custom-drawn widgetset or the FPGUI-based widgetset offers today.
- The widgets will be independent of the LCL: no dependency on the LCL must exist.
- The widgets must be able to co-exist together with an LCL framework: An application must be able to handle LCL forms together with CSS-based forms. In effect, the LCL will provide one of the possible backends for the Fresnel widgets.

The name Fresnel is chosen in honour of the French Physicist Augustin-Jean Fresnel who made important contributions to the theory of light: what is a UI if not a clever play of light?

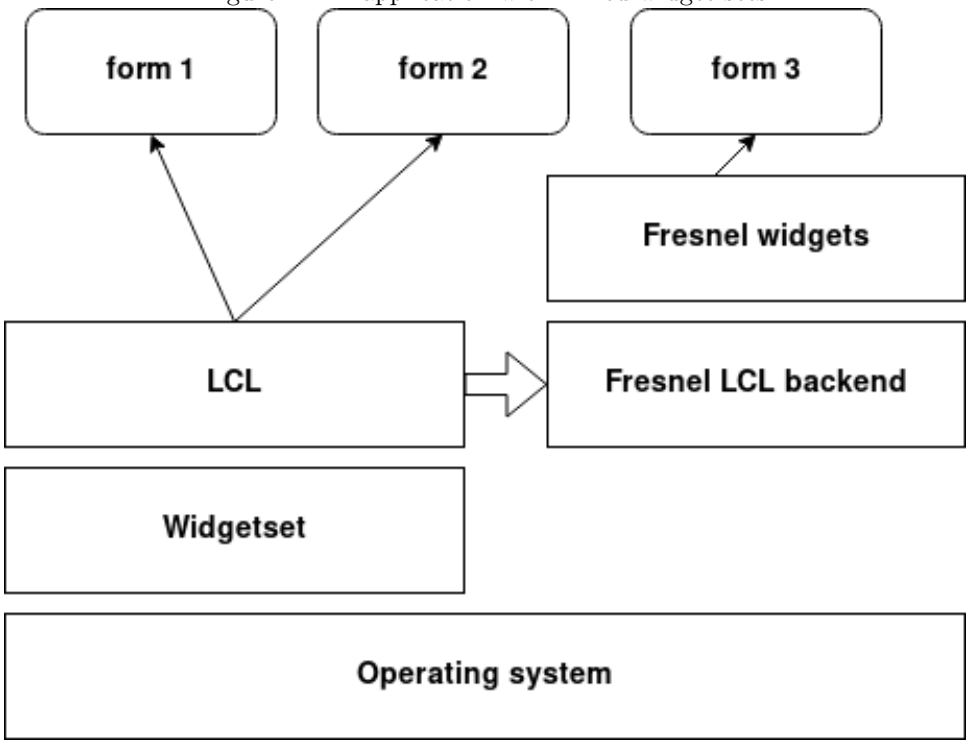
The situation where the LCL is used to back up the Fresnel widgets (as will be the case in the IDE) is depicted in figure 1 on page 3

3 Architectural consequences of CSS

Ideally, a project made with the Fresnel widgets should be able to take an existing CSS framework such as Bootstrap, Foundation or Tailwind CSS and apply it to a form. That means we need to provide several things:

- a CSS parser and a mechanism to translate the CSS to native structures.
- a CSS resolver: this mechanism determines what CSS rules apply to a given widget.
- a drawing mechanism that knows all supported CSS properties.
- a layouter. CSS has several means to determine the layout: float, flex, grid etc, as determined by the `display` CSS property.
- Some widgets that mimic standard HTML elements: the CSS framework assume the existence of certain tag names (div/span/p/label etc.) so we need to provide widgets that react to these names. (see also below).

Figure 1: An application with mixed widget sets



The work on this has been started: a CSS parser and resolver have been made. A drawing backend based on the LCL has been started, and a first layouter has been written.

A CSS rule can select elements based on tag names, element IDs, (pseudo-)attributes and tag hierarchies. For object IDs, it seems natural to use the component name as the attribute. This removes the need for the HTML equivalent of `GetElementByID`, since every element will be a component defined in the form.

A tag hierarchy maps naturally on a parent-child relation such as it exists in the VCL/LCL.

The mapping between tag names and attributes has been abstracted in an interface. It is up to the widget what tags and attributes it reports to the CSS engine.

4 Consequences of not using the LCL

The VCL or LCL handles lots of tasks. It is not only responsible for drawing the controls, but also handles messages coming from the OS: mouse messages, keyboard messages, drag and drop. The clipboard is also handled.

By requiring that the Fresnel Widgets are independent of the LCL, all these things must be re-implemented, although some may be reused from the LCL.

Since the Fresnel widgets do not need to obey the LCL and VCL heritage, this means we can redesign the messaging and event handling system. The LCL/VCL is limited to 1 event handler per message. Messages are sent in windows style, using 2 integers. In old times, performance of the computers necessitated the use of such a limited system. Today, this restriction is no longer needed (as evidenced by the

browser).

The Fresnel widgets will re-imagine this: use records (as in FPGui) or objects (as e.g. in Javascript) to represent the messages, allow multiple event handlers for a given event, but still allow event handlers to be set in the object inspector.

Since the LCL will be used as one possible back-end, obviously a translation mechanism between LCL messages and the new Fresnel messages needs to be implemented.

5 Co-existence with the LCL

The IDE needs to be able to draw and manipulate the Fresnel Widgets. Since the IDE is built using the LCL, it follows that there must be a bridge between the LCL and the Fresnel widgets: The Fresnel widgets should be able to use the LCL as a drawing backend.

This coincides with an architectural requirement put forward in the beginning: the LCL and Fresnel must be able to co-exist in a single application. The idea for this requirement is partially born out of the necessity to be able to manipulate the widgets in the IDE, but more importantly to allow people to port existing applications on a form-by-form basis to the new widget framework. For many applications, the need to port all of the application at once would be simply a too big hurdle to take.

Initially, the resulting application structure would be slightly more complicated: Fresnel widgets, on top of an LCL backend, with the backend based on LCL components, where the LCL components are built on top of a widgetset. That situation was depicted in figure 1 on page 3.

But the final goal is of course an application structure which is simpler: Fresnel widgets, on top of a backend based directly on the OS. This situation is depicted in figure 2 on page 5.

6 Why not simply the browser ?

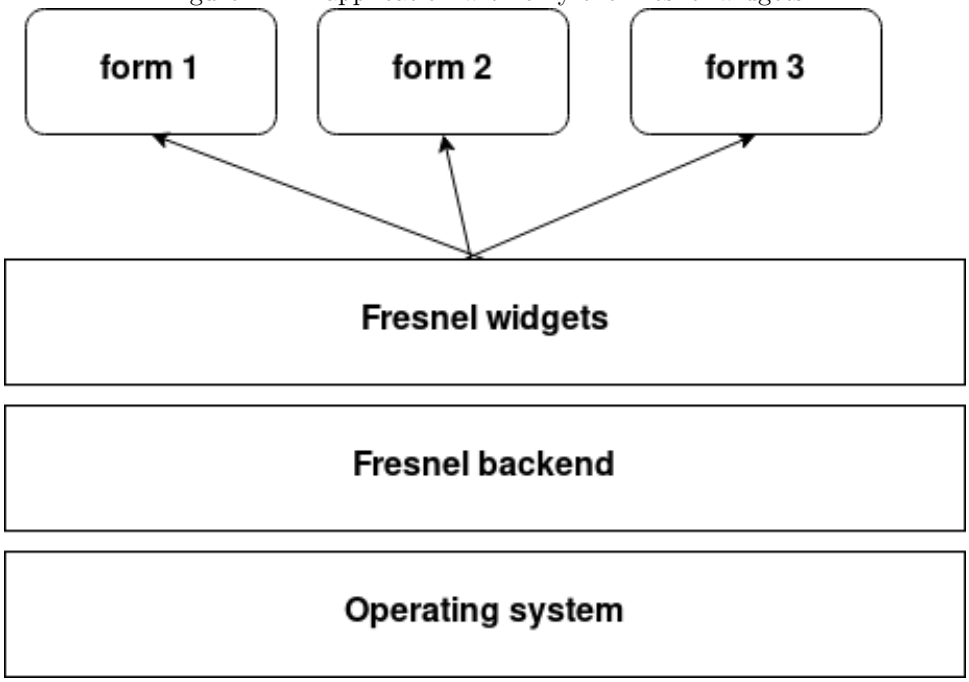
Since we're re-imagining widgets based on CSS, creating a new messaging model which has some characteristics of Javascript DOM: why not simply use the browser or some system like Electron (a packaged browser) or Chromium? As a pascal programmer, you can after all target these platforms with Pas2js or TMS Web Core.

The browser, however powerful, is still a limited environment with its own set of rules. There is no doubt that the browser shines in UI development - which is why Project Fresnel uses CSS as its mechanism - but native applications are still faster, and as a programmer you enjoy more freedom than in a browser environment: access to the file system and other hardware. The browsers also impose an asynchronous model on many operations, which makes it a little harder for the beginning programmer.

What is more, porting to the browser is an all-or-nothing decision: it is not possible to have part of the application in the browser, and part as native. It is of course possible to rewrite an application form by form to a browser application, but interaction between the native and browser part will be very limited.

The fact that Project Fresnel uses CSS also does not mean it is limited to the HTML elements: it will still be perfectly possible to create your own widgets; but you will

Figure 2: An application with only the Fresnel widgets



need to interrogate the CSS mechanism to find out about borders, colors and such properties more.

The goal is also not to create a browser. Although it would be possible to 'import' a HTML file and create a form based on the HTML tags, creating a full-fledged browser is not the goal.

7 Conclusion and the road ahead

Project Fresnel is a large project to rejuvenate the UI possibilities of Pascal: there is a lot of work ahead. But the project has been put in motion, and the first results have been obtained: a proof of concept using the LCL as a backend has been created (see figure 3 on page 6).

The code can be found in a Git repository on Gitlab:

`https://gitlab.com/freepascal.org/lazarus/fresnel`

It currently needs FPC trunk to compile due to the dependency on the CSS framework, but this may change in the future so the release version of FPC can be used (and, who knows ? Delphi...).

The Free Pascal and Lazarus foundation strongly believes in this project and will do everything to back up development of this widgetset. But as said, it is a large project, and the success of the project will depend on the uptake in, and the support of, the whole Pascal community.

Figure 3: The proof of concept