Implement new-generation UI for OpenCV samples and Demos

- Author: Vadim Pisarevsky
- Link: The feature request
- Status: **Draft**
- Platforms: **All**
- Complexity: a few man-months.

Introduction and Rationale

The highgui is one of the oldest and one of the most used modules in OpenCV. It lets users to experiment with computer vision algorithms very conveniently and produce samples or even demos that work on every major desktop platform, be it Linux, Windows or macOS.

Actually, there are two APIs, the basic one with GTK+, Win32 and Cocoa backends, and the advanced one that is built on top of Qt. This second advanced UI offers a lot of useful stuff, but it can be problematic to use because it relies on quite heavy library that is not that easy to install on Windows and Mac. Besides, it's LGPL library, so if we link it statically into highgui, it will become LGPL library as well (besides, it will become a huge LGPL library). A part of this proposal is to port most essential parts of Qt-based HighGUI to other backends. This way we will get much more beautiful OpenCV-based demos by default. But it outlines some more desired features.

Proposed solution

- Base everything on OpenGL (or maybe Vulkan + Metal)

The main idea is to use OpenGL with proper (double) buffering for visualization, at least on Linux and Mac. On Windows we could use DirectX. In the case of UMat we could try to use more efficient code path to display the image without copying it back and forth between CPU and GPU memory.

Some of the UI functionality that Qt-based backend implements, such as buttons etc., we can implement using OpenCV drawing primitives, see cvui for example.

A separate interesting topic is font rendering. Currently embedded into OpenCV Hershey fonts suck. Their only good property is that they are always available. And they are rendered reasonably fast too. That's it. At the same time, Qt-based backend offers the method to render truetype and other supported fonts, which gives very nice-looking text. We can probably store the compressed bitmaps of some pre-rendered free font and then adjust those bitmaps to the proper text size and render them.

Let's list some useful features that the new UI may implement:

- memorize window positions and size and then restore it. Currently, only Windows backend supports it (?)
- automatic resize of the window when a large image is displayed in it. Currently, when we display a large image in a window without autoresize option, the window gets huge (exceeds the screen size).
- play/pause control (some embedded shortcuts in `cv::waitKey()` ?)
- fixed cross-platform keycodes for the popular keys (arrows, ESC, etc.)
- zoom in/out, 100% view, 300/500% view (to visualize pixels)
- support high-resolution/retina displays
- display pixel values
- capture screenshots and store them to desktop
- capture video and store it to desktop
- select rectangular area with a mouse (already implemented?)
- place trackbars as well as other controls (buttons, checkboxes etc.) in a separate control panel
- memorize the state of checkboxes.
- better text rendering

- add extra drawing functions (maybe to imgproc module) to visualize various useful objects: dashed rectangles and other figures, fancy borders and targets (like 4-corner target), arrows etc.
- overlay figures (like sprites) on top of displayed image (via special HighGUI calls) to mark detected objects, selections etc.
- combine several windows in one (tiled view)?
- automatically create some highgui "console" window (probably attached to one of the other windows) that will display stdout/stderr and will react on cv::waitKey(). Currently, users have to switch to the console to see the output (printf's) and then hotkeys stop working until they switch back to the highgui windows.
- **Q:** should we add elements of viz into this new UI module?
- ...

Impact on existing code, compatibility

Since the basic UI, which is mostly used nowadays, is very basic, it's unlikely that users will miss something. The Qt backend can be dropped once this new backend is ready.

Possible alternatives

because this is gradual extension to the current UI, we can prioritise the new features and implement them step by step for one platform, then for another etc.

References

1. The current Qt-based UI API and appearance
2. VIZ module tutorial (in Russian)