Paint.Web

- Marius and Mihai Şucan
- University of ”Aurel Vlaicu”, Arad, Romania

November 2008

Paint.Web

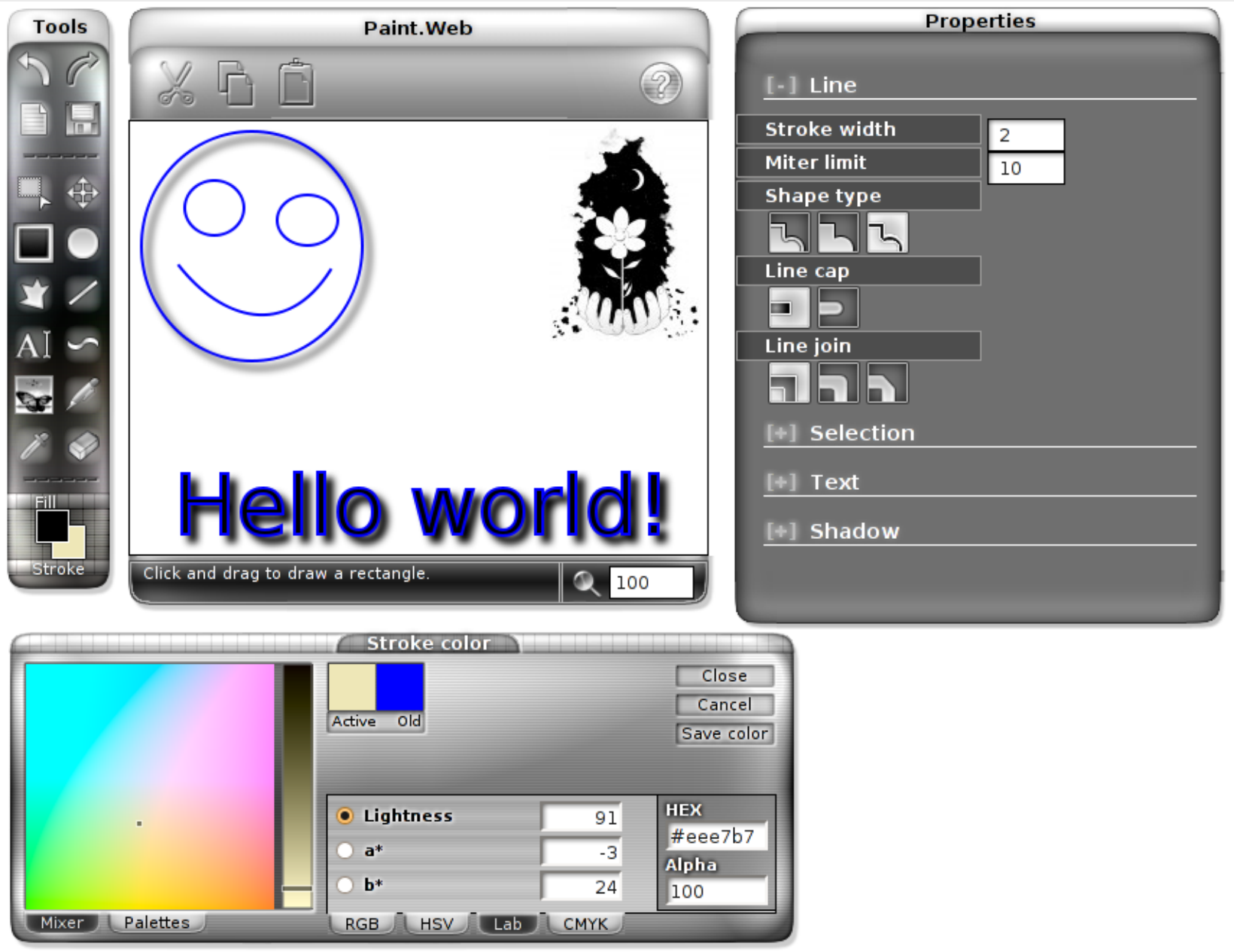
Presentation outline

- Introduction
- The interface
- Features
- Usage
- Structure
- HTML 5
- JavaScript
- The color editor
- Compatibility
- What can be better
- Plans
- Canvas in the future

Paint.Web – Introduction

- Online Web application for painting. Open-source project, GPL v3, hosted at http://code.google.com/p/paintweb.
- This is a technical demonstration of **<canvas>** from HTML5, together with Web Forms 2.
- It makes use of the 2D canvas context. This API allows you to quickly paint on a bitmap 2D surface using basic functions.
- Canvas was first implemented in Safari by Apple.

Paint.Web – The interface



Paint.Web – Features

- Drawing:
  - selection, rectangle, ellipse, line, Bézier curve, polygon, crayon, text, images, and an eraser.
- Options:
  - save images as PNG, history (undo/redo), canvas resize, color picker, zoom.
- Properties:
  - fill color, stroke color, stroke width, line joins and caps, filled shapes and stroke-only shapes, shadows (offset, blur and color), font name, font size, text alignment, and text styling (bold and italic).

Paint.Web – Usage

- Intuitive interface displaying contextual information.
- Keyboard shortcuts for each tool and button.
- During drawing some tools allow the use of keyboard modifiers, such as Shift.
  - For example, rectangles become squares, ellipses become circles, etc.

Paint.Web – Structure

- HTML for the interface.
- CSS for layout/design + PNG images.
- JavaScript for interactivity.
- Strict delimitation: in JS code you won't fine CSS/HTML, nor will you find CSS/JS in HTML.
- Absolutely no server-side code (PHP, Perl, Python, etc.). You can run Paint.Web without a Web server.

Paint.Web – HTML 5

index.html

- <!DOCTYPE html>
  - The DOCTYPE relevance is given by the CSS strict rendering mode activation.
- New attributes:
  - <meta charset=”utf-8”>
  - Web Forms 2:
    - Input type=number/range min=0 max=100 step=0.1
    - ... and others
- Invalid code according to validator.w3.org.
  - Try www.validator.nu

Paint.Web – JavaScript

index.js

- Everything is stored in a single JS object which is executed when loading the main document.
  - No added DOM global properties/methods.
  - Easy to integrate into other projects, without any conflicting function or variable names.
- 6300 lines of code with comments for each function explaining the choices I made and how it all works.
- Easy to translate: all the messages are grouped into only two JS objects.

Paint.Web – JavaScript: Structure

Overview

- We will use ”app” to name the main JS object of the Web application.
- ```
  app = function () {
      ... elems,          inputs,
      img,                img_temp,
      messages,           status_texts,
      init,               init_tools,
      init_keys,          tool_activate,
      ev_canvas,          ev_keypress,
      tools,              kshortcuts, ... };
  ```
- ```
  window.onload = app;
  ```

Paint.Web – JavaScript: Running

- All the references to important DOM nodes are stored in **app.elems** and in **app.inputs**.
- The 2D context references are stored in **app.img** and in **app.img_temp**.
  - During the use of any tool, the temporary drawing is executed in the temporary buffer (**img_temp**).
  - At the end, the tool code calls **app.img_update()**. This method copies the image from **.img_temp** to **.img**, and adds a new history step by calling **app.history_add()**.

Paint.Web – JavaScript: Running

- All the messages used in the application are stored in **app.messages** and in **.status_texts**.
  - Messages can be associated to DOM nodes by adding their ID to **.status_texts**.
- **app.kshortcuts** stores the global keyboard shortcuts with their associated actions.
  - Each shortcut can activate a tool and/or call a function.
- **app.tools** stores the code of each drawing tool.
  - Each tool can have code associated for initialization and for each mouse/keyboard event.

Paint.Web – JavaScript: Running

- The **app.init()** method prepares the canvas contexts and adds the majority of event listeners in the application.
  - It also executes other initialization methods: **init_tools()**, **init_properties()**, **coloreditor.init()** and **init_keys()**.
- **app.init_tools()** does the following:
  - It adds the event listeners for all the drawing tools.
  - It activates the default tool (**app.tool_default**).

Paint.Web – JavaScript: Running

- The **app.init_keys()** method adds each keyboard shortcut to the title of all the associated buttons.
  - This is done to inform the user on the available keyboard shortcuts.
  - For example: **Undo [ctrl-z]**
- The **app.ev_keypress()** method is the event handler for keypress.
  - It looks into **app.kshortcuts** for the keyboard shortcut. Any shortcut can have a function/tool associated, which is automatically activated.

Paint.Web – JavaScript: Running

- The **app.ev_canvas()** method is the event handler for all canvas-related events (mostly mouse events).
  - The function checks if the active tool has an associated event handler for the current **event.type**. If yes, it is executed.
  - The function determines the exact mouse position in relation to the canvas, irrespective of zoom and scroll. The coordinates are given to the tool-specific event handler (**ev._x** şi **ev._y**).

Paint.Web – JavaScript: Running

- The **app.tool_activate(id, ev)** is the method which allows tool activation.
  - It takes two arguments: the tool ID and an optional DOM event object.
  - The tool object construction code can cancel the activation.
    - For example the ”Add image” tool cancels activation if the user cancels the URL prompt.
  - **app.tool** holds the object of the active tool.

Paint.Web – JavaScript

Minimal code sample from the rectangle tool

- You can find the implementation of all tools in **app.tools.**
- ```
  app.tools = {
    'rect' : function () {
      var _tool = this;
      _tool.mousedown = function (ev) {
        _tool.x0 = ev._x
        _tool.y0 = ev._y;
        _tool.start = true;
      };
      _tool.mouseup = function () {
        app.img_update();
        _tool.start = false;
      };
  ..................................
  ```
  ```

Paint.Web – JavaScript

Minimal code sample from the rectangle tool

```
      _tool.mousemove = function (ev) {
        if (!_tool.start) return;

        var x = Math.min(ev._x,  _tool.x),
            y = Math.min(ev._y,  _tool.y),
            w = Math.abs(ev._x - _tool.x),
            h = Math.abs(ev._y - _tool.y);

        app.img_temp.clearRect(0, 0,
                               app.imgW, app.imgH);
        app.img_temp.fillRect(x, y, w, h);
      };
    } // app.tools.rect
  }; // app.tools
```

Paint.Web – JavaScript

- While implementing each tool we tried to ensure each user action has real-time visualization.
  - For example, the Bézier curve tool shows the curve starting from the first two points, without waiting for all the points, like other applications.
- The selection tool uses the third canvas element for storing the selected pixels.
  - Unlike other similar applications, Paint.Web allows the manipulation of pixels and the selection itself (resize and drag).

Paint.Web – The color editor

- Everything related to the color editor is stored in the **app.coloreditor** object:
  - Methods for converting colors between color spaces: RGB, HSV, CIE Lab şi CMYK.
  - The **draw_chart()** and **draw_slider()** methods deal with the color space visualization All drawing is done in a separate canvas element.
  - The predefined color palettes are imported from Photoshop.

Paint.Web – Compatibility

Opera 9.5 (Kestrel)

- Due to some error, the mouse pointer does not update its shape soon enough, when using the selection tool.
- On Mac and Windows, the **drawImage()** method darkens transparencies.
- The Canvas shadows and text-related APIs are unimplemented.

Paint.Web – Compatibility

Opera 9.2 (Merlin)

- Same problems like in Kestrel (Opera 9.5) and more.
- No implementation for **get/putImageData**, thus Undo/Redo don't work. The CIE Lab color space visualization doesn't work either.
- No implementation for **globalCompositeOperation 'lighter'**. The RGB color space visualization doesn't work.
- Some CSS 3 Selectors are unsupported, breaking the rendering of the color editor.

Paint.Web – Compatibility

Safari 3+

- Safari 3 is similar to Opera 9.2, having no support for **get/putImageData**, shadows, nor text drawing.
- Latest SVN trunk builds of Webkit implement text and **get/putImageData**.
- Webkit seems to be the fastest renderer.

Paint.Web – Compatibility

Firefox

- Version 2:
  - Slow renderer and extremely slow rendering when it comes to **get/putImageData**.
  - The Eraser tool doesn't work.
  - No support for drawing text and shadows.
- Version 3 is much faster, being similar to Opera 9.5.
  - No support for drawing text and shadows, either.
- Version 3.1 (beta) the text and shadows API.
  - This is the only Web browser which has complete support for Paint.Web.

Paint.Web – Compatibility

Other Web browsers

- Konqueror 4+
  - I was surprised to see that the KDE developers made their own **<canvas>** implementation.
  - It implements all the Canvas 2D context API, except text rendering.
  - At the moment, the entire browser is a bit unstable, but the problems are quickly solved.
- Microsoft Internet Explorer
  - It has no support for canvas.
  - There are some efforts organized by other groups to implement canvas in IE.

Paint.Web – What can be better

- The current history (Undo/Redo) stores only images for each step.
  - A hybrid history system would be more efficient.
- The selection tool needs rethinking for ”larger scale” use.
  - The purpose of the reimplementation would be to have selections of other shapes as well.
- Performance optimizations are much needed.
- A more dynamic interface: side panels and floating panels.

Paint.Web – Plans

... or some ”ideas”

- Add more important features:
  - Filters, layers, gradients, patterns, ”smart objects”, plugins, and even SVG.
- Animated interface: CSS Animations or SVG.
- Integration into much bigger Web applications.
- More HTML 5:
  - Offline Web Application
  - Client-side storage and database storage
  - Drag and drop
  - Server-sent events

Paint.Web – Canvas in the future

- Canvas can have multiple contexts (not at the same time).
- Besides the 2D context, main browsers are implementing the 3D context as well.
- Hardware acceleration with OpenGL/DirectX for both contexts.
- For the 2D context we will have more methods.
  - For example, while working on Paint.Web some browsers added support for rendering text and shadows.

Paint.Web – Merci

- Thanks for your time.
- To test Paint.Web go to:
  - www.robodesign.ro/paint.web
  - http://code.google.com/p/paintweb