Dezvoltarea şi integrarea aplicației de desenat PaintWeb în Moodle

- Marius şi Mihai Şucan
- Universitatea ”Aurel Vlaicu”, Arad, România

- Proiect Google Summer of Code 2009
- Mentor: Martin Langhoff

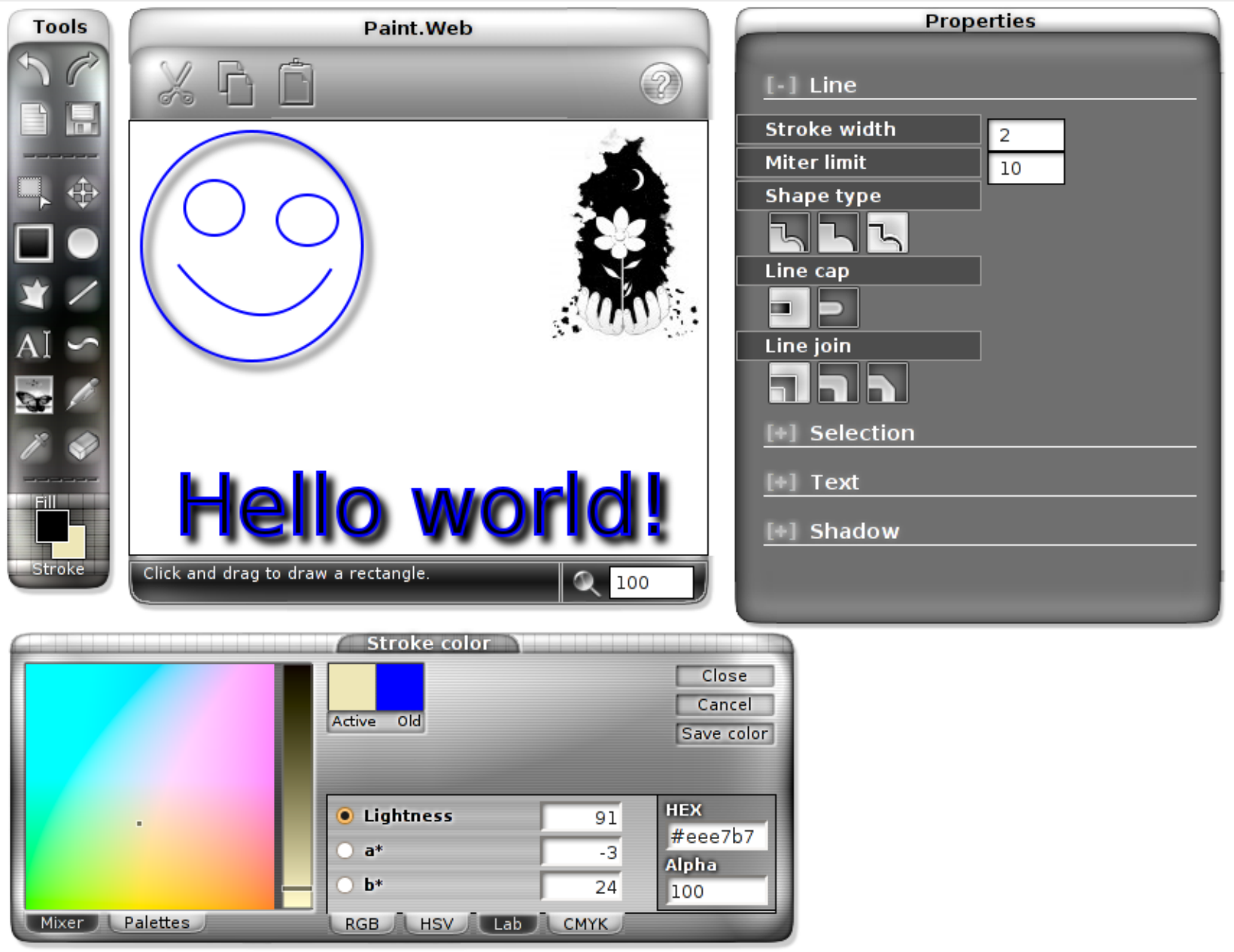
DESIGN ROBO
WE BRING YOU THE FUTURE

August 2009

PaintWeb – De la ce am început

- Anul trecut am dezvoltat o aplicație Web de desenat. Proiect open-source, GPL v3, găzduit pe Google Code.
- PaintWeb a fost o demonstrație a tehnologiei **<canvas>** (contextul 2D) din HTML 5, împreună cu Web Forms 2.
- Rezultatul:
  - Un singur script de aproximativ 6000 linii de cod, fără comentarii în format jsdoc.
  - Două tutoriale despre Canvas pentru Opera Software.

PaintWeb în noiembrie 2008



De ce PaintWeb?

- A început ca o demonstrație tehnică pentru tutoriale la Opera, despre HTML 5.
- A evoluat peste aşteptări.
  - Am decis lansarea lui open-source, în speranța de a face îmbunătățiri pe viitor.
- Deşi nu este evident, PaintWeb are capabilități ce depăşesc programe gen clasicul MS Paint.
- Originalitate: PaintWeb este singurul proiect open-source, aplicație de desenat sub formă de componentă, cu HTML 5 Canvas.
  - Există încercări cu Flash, Java, sau demouri de Canvas, dar nici unul la fel de evoluat şi nici unul cu această combinație de tehnologii.

Propunerea de proiect GSOC

3 Why Canvas [modifier]

Canvas is a new element in the HTML 5 specification, which provides a 2D context API.

- Drawing is bitmapped, therefore no additional DOM elements needed, Image data management/manipulation becomes simple and straight-forward.
- The entire API is native to the browser. No extensions needed, no plugins.
- "Communication" between the Web page and the Canvas element is trivial, not even worth mentioning. Fact is, Canvas lives in the native space of the browser. You do not have any sandboxing, like in plugins (Flash/Java Applets). This also contributes to improved performance.
- Loading and saving images is trivial. The Canvas 2D context API provides simple methods for such purposes.
- Communication between JS and the server can be done via AJAX. It's much easier than doing it from Flash/Java Applets.
- Drawing is faster than using HTML, Flash, and Java Applets. Not-so-distant future plans for major Web browsers include OpenGL/DirectX hardware acceleration for most operations which means chosing Canvas is a very good decision in the longer term.
- Even if drawing is done on a bitmapped 2D canvas, the specification allows future browsers to implement saving the image as SVG, not just in a bitmapped format. The availability of SVG export can come with no cost for the Moodle developers, given a browser implementation which retains sufficient information to allow exporting the image to SVG.
- Better keyboard accessibility. Unlike Flash, Java Applets and other plugins, Canvas does not steal away the keyboard focus (one of the most annoying problems with plugins). Given the OLPC XO laptop and children users, this is very important for them.
- Zero performance impact for loading the Canvas-based Web application. Java Applets load very slow, freezing the browser for a few seconds (on AMD Athlon XP 1800+), and they require a lot more memory, because you have to run the entire JVM inside the browser. Flash also requires more memory to run. Again, users of OLPC XO laptop will most likely disable Flash/Gnash entirely, because it crawls down the entire browser when many Flash banners are shown on a single page.
- Flash is known, even by Google, to cause numerous crashes, and I quote: "**Plugins are a major source of browser crashes. (Flash is in the callstack of a nontrivial percentage of Firefox crashes.)**" [3] It is, therefore, to be desired to avoid any kind of plugins, for stability reasons, not just performance.
- Canvas is a major browser feature being improved constantly. Each new version of each major Web browser improved the performance of the API.
- Almost no learning curve associated with developing a Canvas-based Web application. Existing JavaScript developers can further improve and fix the project at any time. They do not need Flash Professional (which is not free), they do not need to learn ActionScript, they do not need to learn Java. In general, it's better to keep the codebase compact, instead of each project using its own language. It's like using Python instead of PHP for some server-side parts.
- Cross-browser and cross-platform compatibility. Users do not need to install a different browser (it works in almost all major browsers, including, but not limited to Safari, Chrome, Opera, Firefox, Konqueror, Epiphany), and they do not need additional plugins, nor additional settings rather than the defaults. No specific operating system requirements are made either. Developers who want to work on the paint tool do not need to switch to Windows in order to edit some Flash project, and they do not need a completely different development stack for editing some Java project.
- Given the fact that all major browsers implement the Canvas 2D context API, except Microsoft Internet Explorer, makes it reasonable to expect that an upcoming release of MSIE will indeed support Canvas. Major companies already use Canvas in their projects (see Yahoo Pipes). Ultimately, it is in Microsoft's best interest to support a wider range of Web applications.
- The final version of Internet Explorer 8 was released in March. While it does not yet support the new Canvas element, this release has proven the renewed interest of Microsoft to implement new Web technologies, in an interoperable way. They have many standards-related improvements.
- Internet Explorer compatibility can be tackled in multiple ways. For example, very recently a new release of the ExplorerCanvas project from Google has been made available. Such layers of compatibility are simple to use and do not generally require any additional changes to the drawing tool itself. Once IE will natively implement the Canvas support, we can simply remove the compatibility layer for the new version.

I should also note that performance is critical not only for the OLPC XO. Many mobile devices like netbooks, notebooks and cell phones are getting connected to the Internet. All these devices have limited resources.

It's not very long time since Adobe released an up-to-date Flash player for Linux. Until then we were stuck with an outdated Flash player. However, now Flash player is much slower on Linux than the Flash player for Windows. We shouldn't be quick to forget the harm of closed source technologies.

Regarding the use of more JavaScript libraries: the Moodle community and developers have made their choice. They use the Yahoo UI library. I do not want to introduce yet another library which only make things worse and slower. Performance is important and we also don't want bugs between conflicting JS libraries. I will use the YUI library when I consider it's needed inside PaintWeb.

De ce Canvas?

- Un element <canvas>.
  - API în DOM, nativ navigatorului, fără extensii/plugins.
- Desenare 2D bitmap, fără DOM pentru elementele grafice.
- Accesibilitate mai bună din tastatură.
  - Este bine ştiut că Flash, Java şi alte pluginuri au probleme cu activarea ferestrei dorite.
- Încărcare cu impact aproape nul asupra performanței.
- Stabilitate mult mai bună.
  - Pluginurile (mai ales Flash) cauzează cel mai mare număr de blocări în navigatoarele Web.

PaintWeb – Ce am făcut

- Reorganizare completă a codului.
- Optimizări de performanță pentru OLPC XO.
- Definire API pentru extinderea lui PaintWeb.
- Îmbunătățiri de funcționalități.
- Interfață nouă.
- Împachetare.
- Integrare în TinyMCE 3 cu un plugin.
- Integrare în Moodle 1.9 şi 2.0.
- Documentație.

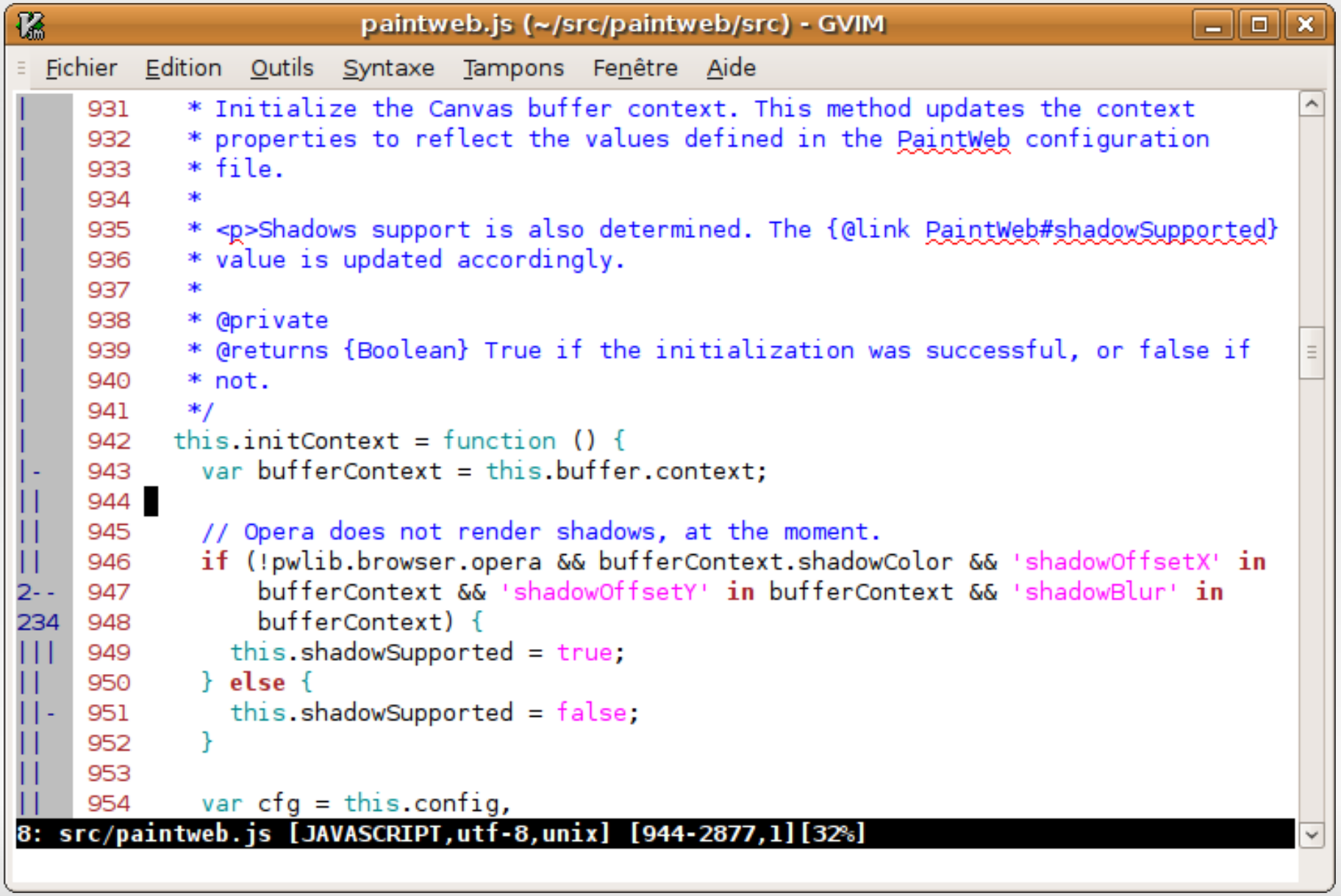
PaintWeb – Reorganizare cod

- Fişiere separate pentru fiecare unealtă de desenare, extensie şi limbă.
- Fişier nou de configurare.
- Cod mai formal, mai strict, cu mai multă consecvență în denumirea funcțiilor şi a variabilelor.
- Comentarii jsdoc pentru toate metodele şi proprietățile obiectelor definite.
- Fişiere separate pentru interfață: JavaScript, CSS şi XHTML.

PaintWeb – Organizare fişiere

- **build/** - PaintWeb împachetat.
- **demos/** - demonstrații de integrare PaintWeb.
- **docs/** - documentație.
- **ext/** - cod de integrare în proiecte externe.
- **scripts/** - scripturi folosite pentru împachetare.
- **src/** - codul sursă PaintWeb:
  - **extensions/**, **includes/**, **interfaces/**, **lang/** şi **tools/**;
  - **paintweb.js** – fişierul principal.
- **tests/** - diferite teste făcute pentru PaintWeb.

PaintWeb – Exemplu cod sursă



PaintWeb – Despre OLPC XO

- One Laptop Per Child School Server este un proiect ce include Moodle.
  - PaintWeb trebuie să ruleze bine pe OLPC XO-1.
- Detalii tehnice despre laptopul OLPC XO-1:
  - Procesor AMD de 433 Mhz, compatibil x86, 64 KB memorie tampon L1, 128 KB memorie tampon L2;
  - Memorie DRAM de 256 MB (dual DDR333 166 Mhz);
  - Disc de 1 GB sau 2 GB SLC NAND flash;
  - Ecran special de 7.5 țoli, TFT. Acesta are două straturi.
- Pe XO rulează un sistem Linux bazat pe Fedora 9.
  - Are un navigator Web bazat pe Gecko 1.9.0 cu interfață în Python.

PaintWeb – Optimizări de performanță

- **Problema:** Imaginile şi textele sunt redimensionate pe OLPC XO.
- În `about:config` valoarea `layout.css.dpi` este 134, în loc de 96 cum este implicit pe PC-uri.
- **Soluția:** anulăm redimensionarea canvasului.
- Fie DPI 200, fie `canvas.width=200` px, afişarea se face la 400 px. Anulăm redimensionarea cu `canvas.style.width="100px"`.
- Cu detectare de navigator pe OLPC XO, PaintWeb merge bine.
  - Folosesc CSS 3 Media query pentru Gecko 1.9.1+.

PaintWeb – Despre performanță

- Mit: JavaScript stă la baza performanței pe Web.
- Aplicațiile Web complexe sunt afectate prima dată de performanța de afişare.
  - CSS+HTML, imagini cu transparențe, etc.
- PaintWeb este încetinit de performanța de afişare CSS+HTML şi a Canvasului.
  - Nu este (încă) încetinit de JavaScript.
- Pe OLPC XO problemele de performanță s-au rezumat la redimensionarea de imagini şi la afişarea de imagini în fundalul Canvasului.

PaintWeb – API nou

- Înregistrare de extensii.
  - `PaintWeb.extensionRegister('id').`
- Înregistrare de unelte de desenat.
  - `PaintWeb.toolRegister('id').`
- Înregistrare de comenzi noi.
  - `PaintWeb.commandRegister('undo', undoFn).`
- Evenimente în cadrul aplicației.
  - `PaintWeb.events.add('imageSave', evFn).`
- … şi altele (vezi documentația).

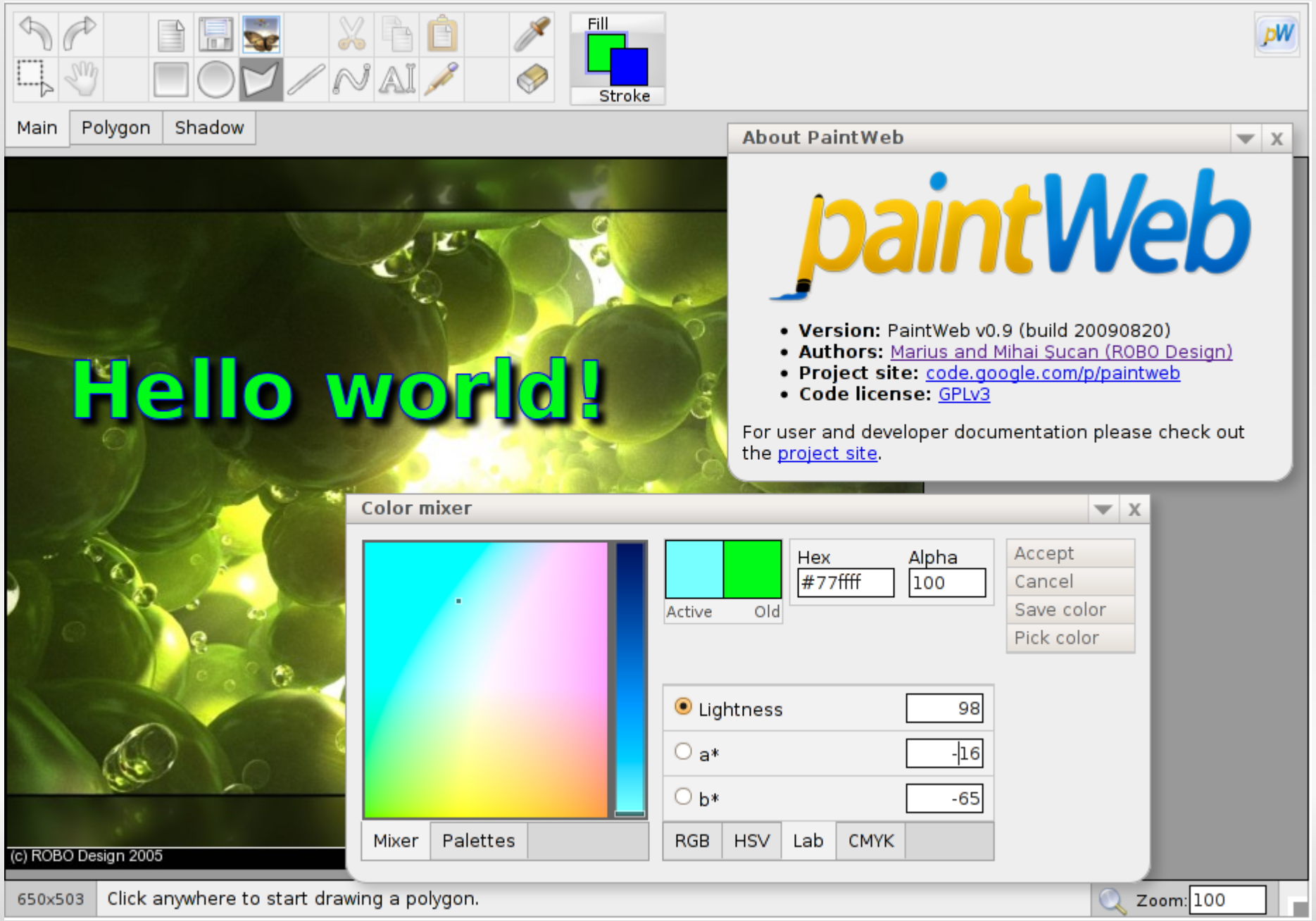
PaintWeb – Îmbunătățiri funcționalități

- Extensie nouă: MouseKeys. Utilizatorii pot desena fără mouse.
- Reimplementat suportul pentru comenzi din tastatură, cu compatibilitate mult mai bună între navigatoare.
- Multe îmbunătățiri şi erori eliminate pentru unealta de selecție pixeli.
- „Hand”: unealtă nouă de mutat imaginea în viewport.
- Îmbunătățiri pentru unealta „Eraser” (guma de şters).
- Suport mai bun pentru imagini mari, 6000 x 6000 px.
- Îmbunătățiri şi compatibilitate cu mai multe navigatoare Web pentru unealta de adăugare text.

PaintWeb – Interfața nouă

- Uşor de modificat, trei fişiere: CSS, XHTML şi JavaScript.
- Încărcare dinamică, la cerere.
- Atribute proprietare în codul HTML:
  - `<p data-pwCommand="imageSave">Save image</p>`
  - `<p data-pwTool="selection">Selection</p>`
  - `<input data-pwConfig="line.lineWidth" type="number" min="1" max="100" value="1">`
- Accesibilitate din tastatură mult mai bună.
- Interfață contextuală: opțiunile apar dinamic.

PaintWeb – Captură de ecran



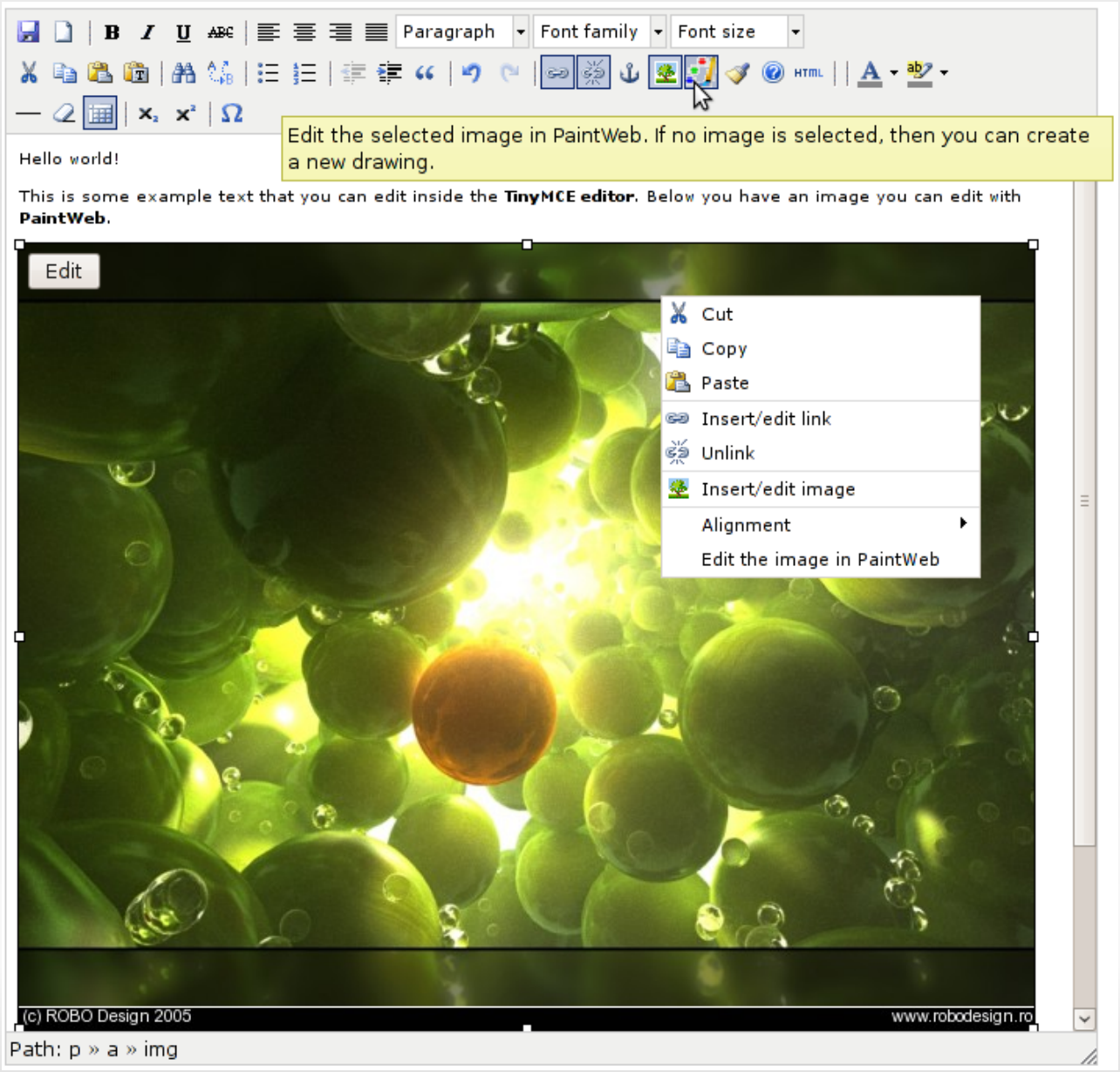
PaintWeb – Împachetare

- Codul este împachetat automat cu un Makefile şi nişte scripturi java, bash şi PHP.
- YUICompressor este folosit pentru comprimare JS şi CSS.
- Am făcut un script PHP propriu pentru image inlining, cu data URLs în CSS.
- Jsdoc-toolkit este folosit pentru generarea referințelor API.
- Rezultate:
  - De la 70 fişiere la 6 fişiere.
  - De la 700 KB la 460 KB sau 140 KB (cu gzipping).
  - Inițializare sub o secundă, cu Internet rapid.

PaintWeb – Integrare în TinyMCE

- Un plugin, uşor de instalat (copy/paste folder).
- Buton pe bara de unelte:
  - Dacă nici o imagine nu este selectată: se adaugă o imagine nouă cu dimensiunile dorite şi porneşte PaintWeb.
  - Dacă o imagine este selectată: se porneşte PaintWeb automat.
- Buton „Editare” afişat dinamic pe imaginea selectată în articol.
  - Sau dublu-click pe imagine pentru editare.
- Meniu contextual (click dreapta) ce oferă opțiunea de editare cu PaintWeb.

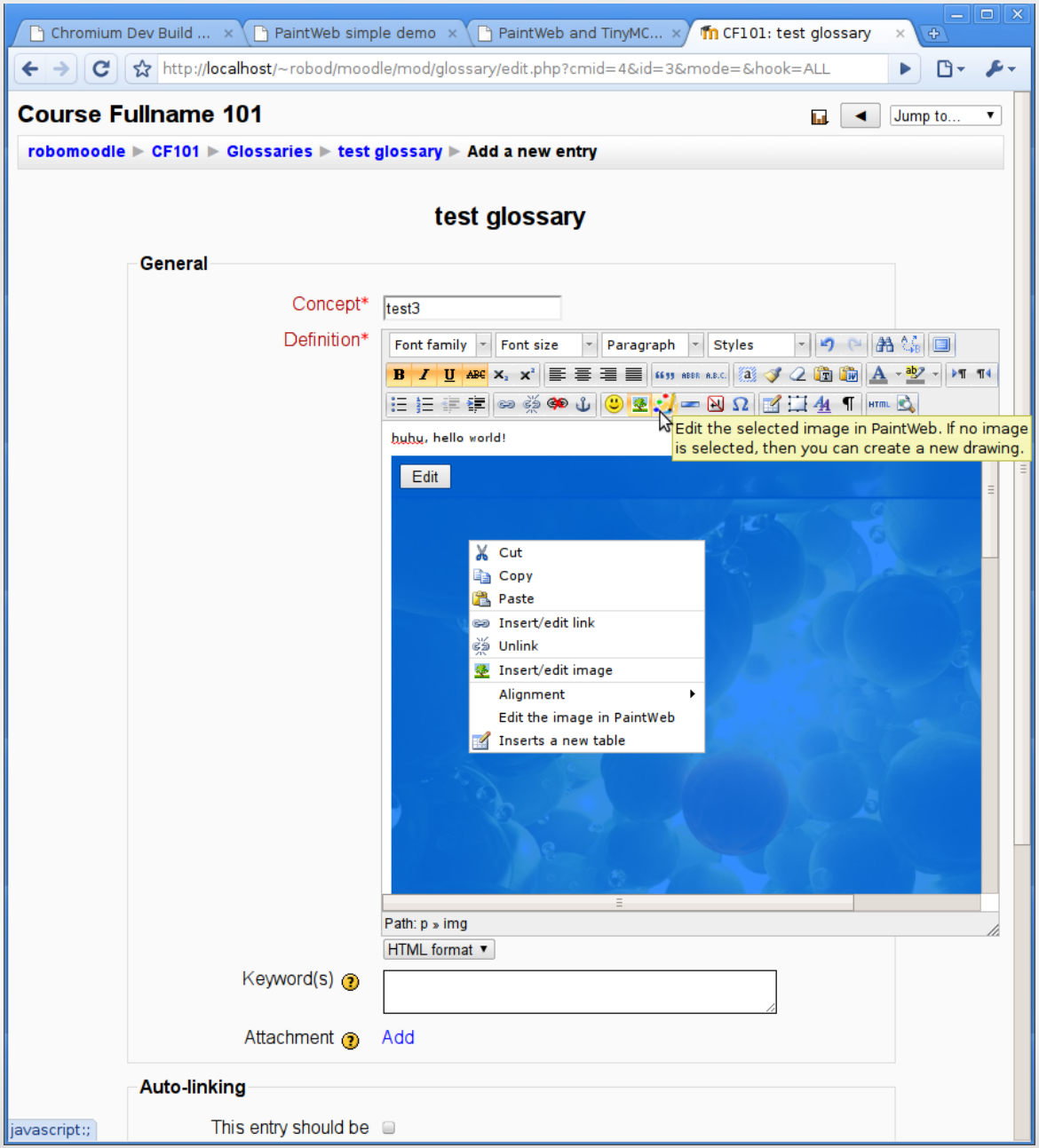
PaintWeb în TinyMCE



PaintWeb – Integrare în Moodle 2.0

- Moodle 2.0 este în curs de dezvoltare.
- Multe îmbunătățiri, printre care un File API nou.
- Foloseşte TinyMCE 3 implicit.
- PaintWeb este integrat în TinyMCE şi poate salva imaginile pe server cu noul File API.
- PaintWeb are o extensie proprie care se ocupă de salvarea imaginilor în Moodle.
- Orice ataşament la un `<textarea>` intră în user_draft file area. La salvare, ataşamentele sunt mutate într-un file area non-temporar.

PaintWeb – TinyMCE 3 în Moodle 2



PaintWeb – PaintWeb în Moodle 2

PaintWeb – Integrare în Moodle 1.9

- Moodle 1.9 este versiunea stabilă. Sunt acceptate doar soluții pentru erori existente.
- Nu are un File API suficient de evoluat.
- Foloseşte HTMLArea implicit, dar are şi TinyMCE 2 - foarte vechi ambele.
- Martin Langhoff a publicat patchuri pentru integrarea lui TinyMCE 3. Acestea le-am actualizat la ultima versiune.
  - PaintWeb este funcțional în TinyMCE 3.
- Salvarea de imagini nu este finalizată. Mai trebuie şterse imaginile neutilizate, şi trebuie backups/restore la cursuri.
- Disponibilitatea proiectului va fi sub forma de contrib-patches pentru oricine.
- Principalul beneficiar este OLPC school server.

PaintWeb – Documentație

- Referințe API la tot codul, inclusiv pentru unelte de desenat, extensii, comenzi, interfață şi evenimentele aplicației PaintWeb.
- Utilizarea în TinyMCE.
- Împachetarea lui PaintWeb.
- Cum se poate extinde PaintWeb.
- Câteva tutoriale scrise pentru Opera:
  - Introducere în HTML 5 Canvas.
  - Cum s-a început dezvoltarea lui PaintWeb.
  - Accesibilitatea din tastatură a aplicațiilor Web.

PaintWeb – Compatibilitate

- Navigatoare Web: Opera, Konqueror, Google Chrome, Firefox şi Safari.
- Motoare de navigatoare Web: Presto, KHTML, Webkit şi Gecko.
- Microsoft Internet Explorer nu are Canvas.
  - Probabil vom folosi excanvas pentru compatibilitate.
- Unealta de text nu merge pe Opera.
  - Încercarea de a folosi SVG:Text a eşuat.
- Afişarea de umbre nu merge pe Opera.
  - Nici pe Chrome, dar funcționează pe Safari şi Firefox.
- Multe erori mici şi mari în diferite navigatoare Web.

PaintWeb – Concluzii

- Comunitatea Moodle este prietenoasă şi primitoare.
  - Mulțumiri lui Martin Langhoff (mentor), Helen Foster (GSOC admin la Moodle), Robert O'Callahan (Mozilla) şi Olli Savolainen (student, participant GSOC la Moodle).
- GSOC a fost o experiență din care am învățat multe lucruri noi şi am cunoscut multe persoane.
- GSOC poate fi o rampă de lansare pentru proiecte noi, nu doar un „job” de vară.
  - Vom continua colaborarea cu Moodle.
- Această vară ar trebui să fie doar începutul.
  - … pentru un proiect PaintWeb în dezvoltare constantă.

PaintWeb – Ce urmează

- Finisare a codului şi funcțiilor PaintWeb.
  - Avem un TODO suficient de mare pentru acest scop.
- Finisare integrare în Moodle 2.0.
  - Finalizare integrare în Moodle 1.9.
- Îmbunătățiri de interfață.
  - O variantă mai simplă pentru copii, pentru OLPC.
- Lansare versiune finală şi stabilă PaintWeb.
- Următoarea versiune PaintWeb:
  - Filtre, layers, poate SVG, etc.

PaintWeb – Planurile din 2008

- Funcționalități mai importante:
  - Filtre, layers, degradeuri, patterns, „smart objects” şi chiar SVG.
- Interfață animată: CSS Animations.
- Integrare în aplicații Web (mult) mai mari.
- Mai mult HTML5:
  - Aplicație Web offline (Opera Unite, Adobe Air);
  - Client-side storage şi database storage;
  - Drag and drop;
  - Server-sent events.

PaintWeb – Merci

- Vă mulțumim pentru timpul acordat.
- Pentru a testa aplicația intrați pe:
  - http://code.google.com/p/paintweb

www.robodesign.ro