CANV.3D DEVELOPER REFERENCE

THIS DOCUMENT IS VERY LIMITED DUE TO TIME CONSTRAINS AND IS INTENDED TO GIVE A FAST GUIDE TO USING THE DRAWING TOOLKIT. REFER TO THE CODE FOR THE FULL METHODS AVAILABLE FOR DRAWING IN THE MEANTIME.

CONTENTS

CANV.3D is a stereoscopic drawing library based in HTML5 canvas. It allows the creation of stereo content for web browsers and takes advantage of the '2d' drawing context of the canvas DOM element

developed by diekus & beru in sunny barcelona

https://github.com/diekus/canv.3D

to create side by side stimuli that can be viewed in 3D displays. This document serves as a starting point and reference to develop content using this library.

REQUIREMENTS

The library requires a modern browser (see compatibility section) capable of parsing HTML5 content. To display the created content, a stereo display or projector is required. If the hardware does not automatically recognize the stereo content, manually switch to side by side format.

COMPATIBILITY

	Microsoft Edge	Runs
	Microsoft Internet Explorer	Runs
	Google Chrome	Runs
	Mozilla Firefox	Runs
	Apple Safari (OSX)	Runs

This section needs updating. It will come, but if you manage to test it and want to collaborate, send me an email with the tested browser, version, device and outcomes and I'll be sure to include the information.

COMPOSITION/APPLICATION STRUCTURE

The structure of a composition/application varies based on the static/interactive nature of the project. The most basic structure of a composition consists of a HTML page containing the composition in a canvas. The HTML page has a reference to the canv.3d.js library and a reference to a JavaScript file that has the code of the drawing or composition. While including the JavaScript files to the HTML page,

remember that the references to the 3D drawing functions must be set BEFORE the actual composition or methods that use these functions.
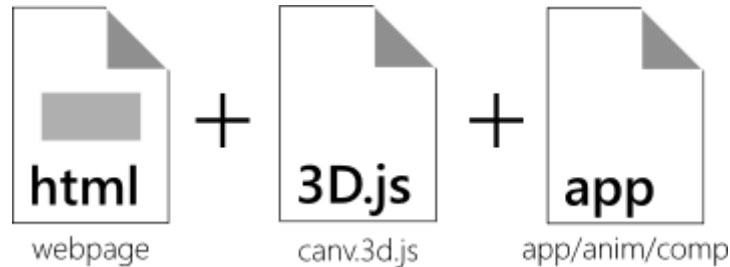
This basic structure can vary by including an animation loop (also included in the basic template). From here the complexity is only limited by the developer's creativity AND the mix with other JavaScript libraries that might implement their own loops.

The recommended implementation for a game/animation loop is built using the browser's RequestAnimationFrame.

Also, when drawing, if the offset is set to a negative value, it is in user space, if its set to positive, it is window space.

The BASIC TEMPLATE

The downloaded package includes a basic template that contains all necessary files to get running in no time. This files are:

- Index.html: web page that will host the composition.
- j/example.js: actual composition. This file is important since it contains the "main" entrance function (`canv3d_main()`).
- j/canv3d.js: stereo 3D drawing library.
- s/canv3c.css: stylesheet applied to the html web page.

FUNCTIONS

Of all the functions specified below work on the current active canvas. To know which the current active canvas is, we can inquire the "`activeDrawingCanvas`" variable on the browser console. To change it, we should use the function `changeActiveCtx`. These functions represent a mapping to the normal '2d' context functions. They are adapted to work over a layer. Most of their usage is self-explanatory, but for more information you can refer to the JavaScript documentation of the '2d' drawing context.

Context

- `ctx.restore ()` and `ctx.save()`: saves and restores the active layer's context.
- ctx.clip():
- ctx.translate (pX, pY):
- ctx.scale (pX, pY):

- ctx.fill():
- ctx.stroke ():
- ctx.clearRectColor (px, py, cWidth, cHeight, fillStyle):
- ctx.clearRect (px, py, cWidth, cHeight):

Drawing

All these functions work 'simultaneously' on each canvas.

- ctx.drawImageDefaultValues (pImg, pPosX, pPosY, pHorOffset):
- ctx.rect (pPosX, pPosY, pAncho, pAlto, pHorOffset): draws a rectangle.
- ctx.circle (pPosX, pPosY, pRadius, pHorOffset): draws a circle.
- ctx.beginPath (): begins a path in the stereo layer.
- ctx.closePath(): closes a path in the stereo layer.
- ctx.lineTo (pPosX, pPosY): specifies a line to the specified location.
- ctx.moveTo (pPosX, pPosY): moves to the specified location.
- ctx.stroke(): draws the defined path.
- ctx.fillRect(): fills the defined path.
- ctx.arc(pPosX, pPosY, pRadius, pStartAngle, pEndAngle, pDirection, pHorOffset): creates an arc.
- ctx.fillText(pText, pPosX, pPosY, pHorOffset): draws the text. It can be filled or just stroked, and the textalign and baselines properties follow the same values for the 2d context.
- ctx.bezierCurveTo (pCP1X, pCP1Y, pCP2X, pCP2Y, pX, pY): creates a bezier curve to the specified position.
- ctx.quadraticCurveTo (pCPX, pCPY, pX, pY): moves the path in a quadratic curve to the specified position.

CREATED CONTENT

The created content is in side by side format. The content adapts to the initial size of the viewport once the window loads. Note that resizing the canvas will blur the image and distort the final composition. The developer might overcome this issue by repainting the scene if she/he sees fit. The library takes care of the coordinate mappings between the coordinates specified and the final rendering. Keep in mind
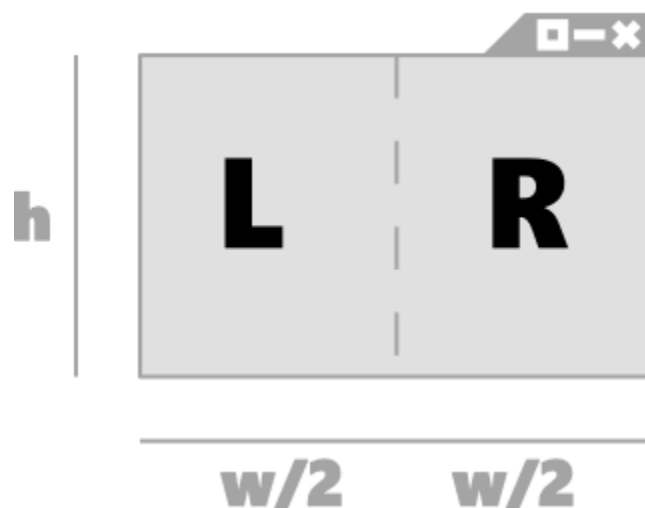
that in order to make the composition stereoscopic the frame is scaled horizontally. The developer should use the coordinates in the original intended position, as if the whole resolution of the screen was available.

CANV3D CODE FLOW (ADVANCED)

The way the canv3D library works is the following. The containing HTML webpage should have two canvases: one for the left view and one for the right view. The following code snippet shows both canvases on the webpage. Their names must always be <name> and sxs3d_<name>. This is required for the script to identify and set the drawing context of all canvases.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <link href="s/3dcnv.css" rel="stylesheet" />
  <script src="j/canv.3d.js"></script>
  <script src="j/example.js"></script>
</head>
<body>
  <div id="dobleCnv">
  </div>
  <div id="output"></div>
</body>
</html>
```

The example.js file, this is, the file where the composition or application resides, must comply with the following:

- On the window.onload function the canvasNames[] must be populated with the name of each canvas on the HTML page. These canvas names do not include the canvas clones (the ones that start with the "sxs3d_" prefix).
    o If there are several pairs of canvases, then each original canvas name must be specified into the canvasNames array.
- The window.onload function must have as its last instruction a call to the method startDuoCanvas().
- The function sxs3dcnv_main() is the starting point of the application. Here is where the composition or application should start.
- If an animation is desired, then call the renderFrame() function.
    o The renderFrame function must call the requestAnimationFrame(renderFrame) to create the loop.

To recap, to use the library, what you need to do is (i) populate the canvasNames Array with the names of the original canvases and (ii) put your drawing instructions in the canv3d_main or renderFrame method. *It's as easy as that*.

DRAWING CONTEXTS

The canv.3d library features the ability to draw in different layers. Each layer represents an independent pair of canvases. Let's see. One basic composition is conformed of two canvases-- the original and the ("sxs3d_") clone. So two canvases represent a stereoscopic layer. But for some experiences and applications, it is desired to have different layers of content (think menus, cursor, and backgrounds in scroller games). To achieve this, the library stacks pairs of canvases over each other. Each pair of canvases has its own drawing context. Each drawing context has its own current paths, colors, and "environmental" settings based on its last code executions. This is, they are completely independent, so if my current fill style in a pair of canvases is set to red, it can be set to cornflowerblue in another pair. The library has been tested with up to 5 layers (10 canvases as a whole).

While the benefits of this might not be immediately visible, the fact that it can avoid repainting whole canvases each frames can be a big performance win. The developer or artist must do everything in his power to make good use of these resources.

If more than 1 layer exists, then the HTML file must have the adequate number of canvases. If we want to develop a 2D "Mario Bros"-like scroller game, we can think of having a layer for the background, a layer for the level and character and a layer for the lives and other UI elements. Each layer can have its own different depths. Again, the developer needs to make sure that his compositions have correct depths in the whole layer view. Putting elements in different depths without assessing if they comply with a logical depth bracket might cause eyestrain and headaches to the user.

```
canvasNames[0] = 'background';
canvasNames[1] = 'cnv';
canvasNames[2] = 'ui';

<div id="dobleCnv">
<!—these pairs of layers will be created by the script-->
    <canvas id="background"></canvas>
    <canvas id="sxs3d_background"></canvas>
    <canvas id="cnv"></canvas>
    <canvas id="sxs3d_cnv"></canvas>
    <canvas id="ui"></canvas>
    <canvas id="sxs3d_ui"></canvas>
  </div>
```

To change between the available drawing contexts, use the instruction `changeActiveCtx(n)`. "n" represents the desired context. In the Mario example, 0 would be the background, 1 the level and 2 the UI elements.

KNOWN ISSUES

PLUG-INS

A charting plug-in and a wrap item list are under develop. Will announce something soon.

Not necessarily a plug-in, but we have managed to create a drawing in a vector drawing application, export it as SVG, converted it to canvas using an online conversion like this one, and then changing the

'2d' functions for their stereo mappings. We created an application that allows the automatic conversion to canv.3d script. Nonetheless, the application needs to be updated to the latest version of the 3D script and migrated online. It currently is a C# desktop app.

EXAMPLES

Refer to http://diekus.net/3dcnv.html. More coming soon.

LICENSE

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/4.0/
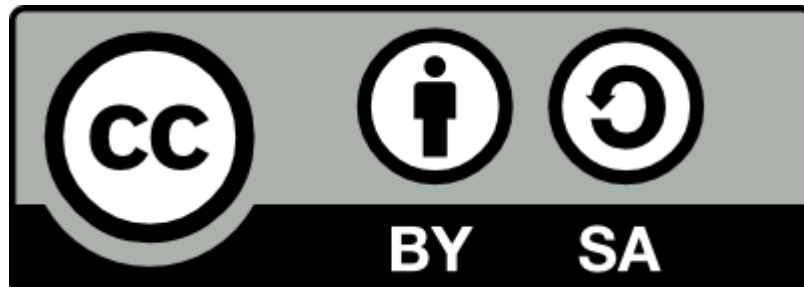
CONTACT

For any questions, suggestions, comments, rants, bugs please contact us via the github CANV.3D page at https://github.com/diekus/canv.3D

Canv.3d was previously called SXS3DCNV.