Elements and APIs useful for writing games

Several elements have already been studied in the HTML5 Part 1 course (canvas, drawing, animation). These will be revisited briefly in this course, but we'll go further. We do, however, recommend going back to the HTML5 Part 1 course, and specifically weeks 3 and 4 that covered drawing and animating, to refresh your memory about the HTML5 canvas.

Here we present some elements that are useful for writing games.

DRAWING: THE `<CANVAS>` ELEMENT

The `<canvas>` is a new HTML element described as "*a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly*." It's a rectangle included in your page where you can draw using scripting with JavaScript. It can, for instance, be used to draw graphs, make photo compositions or do animations. This element comprises a drawable region defined in HTML code with `height` and `width` attributes.

You can have multiple canvas elements on one page, even stacked one on top of another, like transparent layers. Each will be visible in the DOM tree and has it's own state independent of the others. It behaves like a regular DOM element.

The canvas has a rich JavaScript API for drawing all kinds of shapes; we can draw wireframe of filled shapes and set several properties such as color, line width, patterns, gradients, etc. It also supports transparency and pixel level manipulations. It is supported by all browsers, on desktop or mobile phones, and on most devices it will take advantage of hardware acceleration.

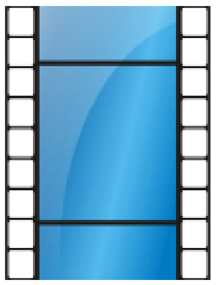
It is undoubtedly the most important element in the HTML5 specification from a game developer's point of view, so we will discuss it in greater detail later within the course.

Latest news: on 19 Novembre 2015, the HTML Working Grouppublished HTML Canvas 2D Context as W3C Recommendation (i.e., Web standard status).

ANIMATING AT 60 FRAMES/S: THE REQUESTANIMATIONFRAME API

The requestAnimationFrame API targets 60 frames/s animation in canvases. This API is quite simple and also comes with a high resolution timer. Animation at 60 frames/s is often easy to obtain with simple 2D games on major desktop computers. This is the preferred way to perform animation, as the browser will ensure that animation is not performed when the canvas is not visible, thus saving CPU resources.

VIDEOS AND ANIMATED TEXTURES: THE `<VIDEO>`ELEMENT



The HTML5 `<video>` element has been introduced in the HTML5 specification for the purpose of playing *streamed* videos or movies, partially replacing the object element. The JavaScript API is nearly the same as the one of the `<audio>` element and enables full control from JavaScript.

By combining the capabilities of the `<video>` element with a`<canvas>`, it's possible to manipulate video data in real time to incorporate a variety of visual effects into the video being displayed, and on the contrary, to use images from videos as "animated textures" on graphic objects.

AUDIO (STREAMED AUDIO AND REAL TIME SOUND EFFECTS): THE `<AUDIO>` ELEMENT AND THE WEB AUDIO API

The `<audio>` element

`<audio>` is an HTML element that was introduced to give a consistent API for playing *streamed* sounds in browsers. File format support differs from one browser to another, but MP3 works on nearly all browsers today. Unfortunately the `<audio>` element is only for streaming compressed audio, so it consumes CPU resources, and is not adapted for sound effects where you would like to change the playing speed or add real time effects such as reverberation or doppler. For this, the Web Audio API is preferable.

The Web Audio API

This is a 100% JavaScript API designed for working in real time with uncompressed sound samples or for generating procedural music. Sound samples will need to be loaded in memory and decompressed prior to being used. Up to 12 sound effects are provided natively by browsers that support the API (all major browsers except IE, but Microsoft Edge supports it).

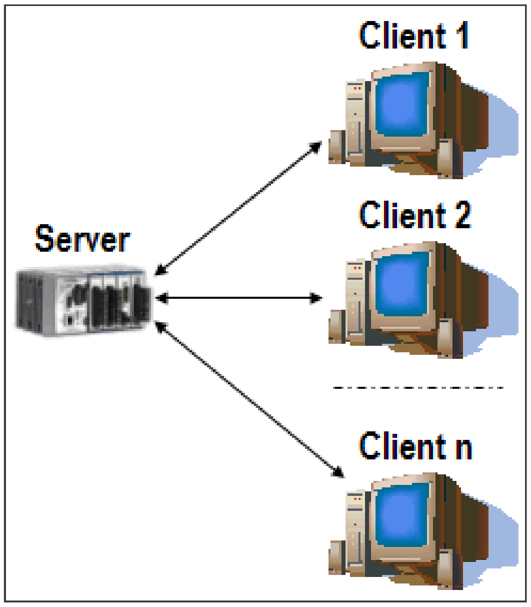
INTERACTING: DEALING WITH KEYBOARD AND MOUSE EVENTS, THE GAMEPAD API

User inputs will rely on several APIs, some of which are well established, such as the DOM API that is used for keyboard, touch or mouse inputs. There is also aGamepad API (in W3C Working Draft status) that is already implemented by some browsers, which we will also cover in this course. The Gamepad specification defines a low-level interface that represents gamepad devices.

MULTI PARTICIPANT FEATURES: WEBSOCKETS

IMPORTANT INFORMATION: NOT COVERED IN THIS COURSE

Using the WebSockets technology (that is not in the HTML5 specification but comes from the W3C WebRTC specification - "Real-time Communication Between Browsers"), you can create two-way communication sessions between several browsers and a server. The WebSocket API, and useful libraries built on top of it such assocket.io, provides the means for sending messages to a server and receiving event-driven responses without having to poll the server for a reply.


Client 1
Server
Client 2
Client n