# Introductory graphics for very diverse audiences

M. Fairén and N. Pelechano

ViRVIG–Moving Group. Universitat Politècnica de Catalunya (UPC), Spain

**Abstract**

*This paper presents our first experience teaching WebGL in a master's degree for a class of students with very different backgrounds. The main challenge was to prepare a course that would be engaging for students with computer graphics experience, and yet interesting and non-frustrating for those students unfamiliar with OpenGL. In this paper we explain how we prepared this course, and the project assignment to achieve our goal. The results achieved by the students show that the course succeeded in keeping different kinds of students engaged and excited with the implementation of their final project.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and techniques—

## 1. Introduction

Interactive Graphics System is a mandatory course in the Masters in Computer Science. One of the goals of this course is that the students learn how to create multimedia applications including 3D graphics and interaction. The main challenge of the course was to make it interesting for students with and without a background in computer graphics. Therefore we needed to prepare a course that could provide new knowledge to those students familiar with the OpenGL fixed pipeline and GPU programing, as well as engaging students who had no previous experience in 3D graphics. We decided to teach WebGL, since it is not included in any course of the undergraduate degree, and thus could provide interesting new knowledge for those students that had taken previous courses on computer graphics.

WebGL (Web Graphics Library) [khr] is a cross-platform web standard for a 3D graphics API based on OpenGL ES 2.0 [MGS08]. It is a JavaScript shader-based API using GLSL [GLS] that is executed on a computer's Graphics Processing Unit (GPU), and can be included in a web page with the HTML5 Canvas element, without the need of plug-ins.

WebGL offers a very exciting way of getting started with 3D graphics, including easy integration in web pages. Using HTML5 we can get the students to implement a project where they incorporate the basic concepts of this course, such as: 3D graphics, multimedia and interaction.

## 2. Outline

This paper details the contents of half a semester (7 weeks of classes), which is the part of the course dealing with interactive 3D graphics and multimedia. The course consists of theory classes, as well as lab sessions.

The course starts with an *Introduction to Computer Graphics* that includes the fundamentals of OpenGL, technical details about the pipeline, the concept of camera with model-view and projection transformations and 3D data structures to render different models. We continue with an introduction to the *Programmable Pipeline*, the GLSL language to program Vertex and Fragment shaders (VS and FS), and how to initialize and compile shaders. Next we explain how 3D models are sent to the pipeline to be rendered, which in WebGL is done using *Vertex Buffer Objects (VBO)*. We introduce the *VBOs* and compare them to the immediate rendering mode (OpenGL 2.0 and Vertex Arrays).

We also explain in depth *Transformation Matrices* for camera and 3D models. We show how to set the camera position, orientation and projection, as well as all the matrix transformations needed to build a scene by placing objects given in model coordinates. In WebGL there are no built-in uniform variables containing these matrices as in OpenGL compatibility mode, so we offer the students some existing libraries that they can use to facilitate their work with matrices and camera [Thr, Sce]. In order to render more interesting models, we also study *textures* and concepts like

DOI: 10.2312/conf/EG2013/education/009-010

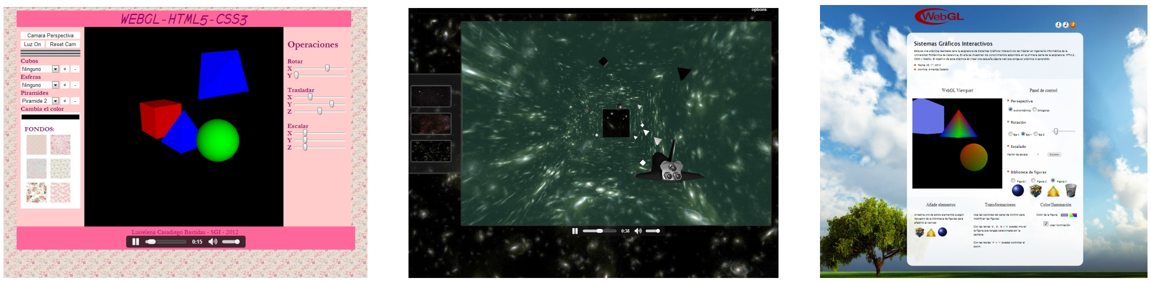

**Figure 1:** *Examples of WebGL project assignments. Two projects to showsimple 3D objects (left and right), with HTML5 audio and Form elements to interactively modify camera, light, add new objects, and apply transformations, with drag and drop used to change backgrounds or add objects. In the center, a game application where the user has to drive a rocket without colliding with simple 3D objects. This includes drag and drop to change the background and different shaders used to render crashes.*

color mapping, bump mapping, opacity mapping and displacement mapping.

In parallel to these theory classes, we carry out lab sessions where students learn about HTML5, JavaScript and CSS3. Each of the introductory classes consisted of one hour of explanation by the Professors, followed by one hour of work in the laboratories with the Professor present to help them out. During this hour the students had to solve a list of exercises. After these three sessions, they had another one on WebGL. The focus of this session was to get the students up to date with the basic OpenGL concepts, provide them with JavaScript libraries to perform the most typical operations (such as: camera and modelview transformations), and introduce shaders programming. During these sessions the students were given a list of exercises to practice implementing simple VS and FS, and learn which variables are needed to pass information between the VF and FS, and between JavaScript, and the shaders.

## 3. Project assignment

The project consisted of developing a program for displaying 3D models, including modelview and projection transformations, interaction through mouse, keyboard and HTML5 form elements, simple VS and FS implementation, and multimedia (audio and video). As additional concepts, they could include more complex shaders (such us: phong shading), or load complex 3D objects using external libraries [OBJ]. The assignment description was kept vague on purpose to give the student the freedom to be creative. We find that students get more interested in their projects if there is not an strict sequence of rules to follow.

Starting basically from the simple examples practiced during the lab sessions, the students could incrementally build a final project without getting stuck in front of an apparently very complex project. We wanted to make sure that even those students without computer graphics background could easily get started with their projects, as long as they had attended the lab sessions.

## 4. Conclusions

We have presented a computer graphics course using WebGL. The main challenge that the Professors faced in this course, was the big differences in the level of computer graphics knowledge of the class. Therefore, we needed to orient the project assignments in a way that students that had previously taken advanced computer graphics classes, could still find the project challenging, and at the same time make sure that students with a lack of computer graphics background could have access to enough information to successfully complete their assignment without getting frustrated and giving up.

From the results achieved by the students, we can conclude that this course succeeded to get students of all different backgrounds excited with their final projects. Even students that had never worked with OpenGL, managed to learn WebGL relatively fast by following our tutorials, other online tutorials [Lea], and completing the basic exercises given in class. We can see in figure 1 the result of several student's projects. It is important to mention that we did not observe a major correlation between their previous computer graphics experience and their final grade. This was achieved by giving higher importance to WebGL concepts which were new for all students rather than implementation of fancy shaders for which experienced students would have had an advantage.

## References

[GLS] http://www.opengl.org/documentation/glsl. 1

[khr] http://www.khronos.org/webgl. 1

[Lea] http://learningwebgl.com. 2

[MGS08] Munshi A., Ginsburg D., Shreiner D.: *OpenGL ES 2.0 Programming Guide*, 1 ed. Addison-Wesley Educational Publishers Inc., July 2008. 1

[OBJ] https://github.com/kod3000/webGL-Examples/blob/master/js/J3DI.js. 2

[Sce] http://scenejs.org. 1

[Thr] http://github.com/mrdoob/three.js. 1