

Master's Thesis



Czech  
Technical  
University  
in Prague

**F3**

Faculty of Electrical Engineering  
Department of Cybernetics

# 3D Atlas of Human Bones

Bc. Michal Šmrha

Leden 2015



## / Declaration

TODO

Abstrakt / Abstract

TODO

TODO



# Contents /

<b>1 Introduction</b> .....	1
<b>2 Problem Analysis</b> .....	2
2.1 Requirements .....	2
2.1.1 Functional Requirements - User .....	2
2.1.2 Functional Requirements ■ Editor .....	2
2.1.3 Functional Requirements ■ Administrator ....	2
2.1.4 Non-functional Requirements .....	3
2.2 Target Group .....	3
2.3 Existing Solutions and Alternatives .....	3
2.4 Analysis of the Current Version of the Atlas .....	3
2.4.1 Bone Selector .....	4
2.4.2 3D Viewer .....	4
2.4.3 PHP Parts .....	4
2.4.4 Editing .....	4
2.4.5 Identified Flaws .....	5
2.4.6 Conclusion .....	5
2.5 Project's Contributions .....	5
<b>3 Relevant Technology</b> .....	6
3.1 3D Technology for the Web ....	6
3.1.1 CSS 3D Transforms .....	6
3.1.2 WebGL .....	6
3.1.3 Flash .....	7
3.1.4 Other Plug-in Technologies .....	8
3.1.5 Performance Test 1: Cubes (WebGL and Flash) .....	8
3.1.6 Performance Test 2: Aquarium (WebGL) .....	9
3.1.7 Frameworks (WebGL) ....	9
3.1.8 Conclusion .....	9
3.2 Server-Side Language .....	10
3.2.1 IEEE Spectrum Ranking .....	10
3.2.2 Personal Experience ....	10
3.2.3 Conclusion .....	11
<b>4 Application Design</b> .....	12
<b>5 Implementation</b> .....	13
<b>6 Testing</b> .....	14
<b>7 Conclusion</b> .....	15
<b>References</b> .....	16

## Tables / Figures

<b>2.1.</b> Identified Flaws .....5	<b>3.1.</b> WebGL vs. Flash rendering frequency .....8
	<b>3.2.</b> IEEE Spectrum - Top Web Development Languages ..... 10



# Chapter 1

## Introduction

TODO

# Chapter 2

## Problem Analysis

The task is to develop a new version of the Atlas of Human Bones, an online database of bones including 3D models with labels and web pages with additional information. The Atlas will serve as a source of information and a study material, primarily for students of the Third Faculty of Medicine at Charles University in Prague. The existing version is insufficient and should be replaced by a new system while reusing the partially pre-filled data content.

### 2.1 Requirements

#### 2.1.1 Functional Requirements - User

Any visitor of the website is considered a user and should have access to following features:

##### **3D Viewer with following features:**

- Display 3D models of bones
- Display labels for certain parts
- Display detailed descriptions of parts when selected
- 3 display modes for labels: Full labels / Labels with hidden names / No labels
- Adjustable view (move, rotate, zoom)

##### **Website with following features:**

- Navigation to specific bones
- Web page for each bone with a 3D model, relevant images, text and references

#### 2.1.2 Functional Requirements – Editor

Editors are authenticated users with access to following extra features:

- Login into editorial system
- Add, edit and delete pages (bones)
- Upload 3D models
- Add, edit and delete labels and descriptions of the models
- Add, edit and delete textual information related to each bone
- Add and delete images related to each bone

#### 2.1.3 Functional Requirements – Administrator

Administrators are authenticated users with access to following extra features:

- Login into account management system
- Create, edit or delete editor accounts



### ■ 2.1.4 Non-functional Requirements

- Browser-based online application
- Multiplatform
- Support for bilingual content (Czech, English)

## ■ 2.2 Target Group

The application is targeted at students of the Third Faculty of Medicine at Charles University in Prague. Other users are to be expected, but the focus should remain on medical students.

TODO: Add details if there's a survey

## ■ 2.3 Existing Solutions and Alternatives

The following list is by no means complete. These are just several examples of the most relevant solutions and alternatives found online.

- Skelet 3D (available at [1]) is the previous version of our project, further discussed in the next section. Our intention is to rework and upgrade this solution.
- Skeletopedia (available at [2]) seems to be the closest independent solution available. It provides interactive 3D models of bones in browser environment as well as simple labeling of certain parts, unfortunately not in great detail. The contribution of our project in comparison to this solution should be more detailed and far more numerous labeling of the models as well as further information about each label and each bone in general.
- Zygote Body, Anatronica (available at [3–4]) and many other systems include 3D models of the human skeleton. However, they provide little to no additional information besides the name of each bone.
- 3D Science provides extremely detailed 3D models (available at [5]). However, these are not free, most likely not suitable for online display and most importantly provide no additional information.
- Palacký University in Olomouc provides information on most bones in their online human anatomy as well as photography of human bones, sometimes including labels with short descriptions (available at [6]). The information, however, seems incomplete. Moreover, there is no 3D material available, which is the trademark of our project.
- Wikipedia provides rather detailed information on human bones (available at [7]). English version includes simple 3D animations of most bones, but it doesn't provide any connection of 3D models with labels and descriptions, which is the intended contribution of our project.

## ■ 2.4 Analysis of the Current Version of the Atlas

The Atlas project is not beginning with this thesis. It was initiated in 2012 and most of the development of the existing version happened in 2013.

The application was developed on the fly and went through a series of overhauls without proper planning. Although it works, it suffers from a variety of flaws because

of this. From a developer's point of view, the internal structure is unorganized and undocumented. While it might be sufficient for now, the lack of order and proper organization makes sustainability and further development an issue.

The current version is a publicly available website running on the servers of the Third Faculty of Medicine, Charles University [1]. It consists of a list of bones sorted in a hierarchy of groups based on parts of the human body, a 3D viewer of models of bones and a very simple editing page which isn't public. An as-of-yet unreleased bilingual version allows editing in Czech and English (as opposed to Czech only).

### ■ 2.4.1 Bone Selector

The list of available bones is a Flash application written in ActionScript 3 that allows selection of a model and filtering using a hierarchy of predefined groups. It does what it's supposed to do, although the layout might not be the most intuitive and there are no additional features such as "search". The loading time is longer than expected due to inefficient use of XML files that define groups and bones.

### ■ 2.4.2 3D Viewer

The model viewer is another Flash (ActionScript 3) application using a simple 3D engine called Sandy with two basic modes: view and edit. It allows the user to manipulate view freely, although not in the most intuitive way, utilizing only one mouse button. It allows users to display or hide labels and pins as it should.

However, it suffers from graphical errors, most notably seeing labels through a bone while they should be behind it and vice versa. This is caused by Z-sorting algorithms used in Sandy 3D engine, which are too simple for the task.

It is also missing some minor tweaks such as adjusting the label width to fit the length of the text.

The edit mode is working, although not pretty to look at.

The rendering speed of the application is subpar, greatly limiting the use on mobile devices and the quality of models. Some of the more complex models are reaching low FPS even on average desktop computers, while being virtually unusable on older machines and mobile devices. A tradeoff between model complexity and rendering speed is to be expected, but current application's performance is nevertheless underwhelming. The technology used doesn't provide easy solutions to some of the aforementioned flaws.

### ■ 2.4.3 PHP Parts

The Flash application is nested in a simple PHP page. The web page which is used to upload new models and enter the editing mode is implemented as a series of simple PHP scripts. Other PHP scripts are called by the Flash application to handle XML files on the server.

### ■ 2.4.4 Editing

There are two versions of the application, a public version with editing disabled and a separate version for editing only, whose location is not known to public, but which is not protected in any other way. The live version is updated manually by transferring data files from the editable version. Introduction of a login system would allow better sustainability through direct editing by privileged users as opposed to transfers by the website administrator.

## 2.4.5 Identified Flaws

Identified Flaw	Affects	Importance	Suggested Solution	Difficulty
No images / text	Content value	High	Add editorial system	High
No authentication	Safety, sustainability	High	Add login system	Medium
Hardware demands	Availability, details	High	Change of 3D engine	High
Graphical errors	User experience	Medium	Change of 3D engine	High
Imperfect GUI	User experience	Medium	Rework GUI	Medium
Bad internal design	Future development	Medium	Proper design	Medium
No documentation	Future development	Medium	Add documentation	Low
No search	User experience	Low	Implement search	Low
Slow loading of lists	User experience	Low	Use DB over XML	Low

**Table 2.1.** Identified flaws of the current version of the Atlas. Author's subjective evaluation of their impact and solutions.

## 2.4.6 Conclusion

Overall, the technologies used seem to be inefficient and partly outdated. A complete reworking of the system seems to be the best solution considering the extent of individual improvements, especially the change of 3D engine and introduction of an editorial system.

The models and labels entered by medical students into the existing version should be valid and reusable in the new implementation.

## 2.5 Project's Contributions

The main goal is to help medical students acquire knowledge of human bones. There are numerous sources available, ranging from lectures and printed textbooks to aforementioned online solutions. Our application cannot compete with the experience and insights of lecturers and it is not likely to replace textbooks because of the sheer volume of information included in those. But it can be a great study material, available anytime, hopefully better than other online sources.

None of the known sources (other than the previous iteration of our Atlas) provide what we hope to achieve in this project: Exhaustive information linked to illustrative 3D models. There are 3D models and there is information. This project's contribution is linking the two in an interactive manner, allowing students easy navigation and a mental link between spatial and textual information.

This was attempted in the current version of the Atlas in a clumsy and imperfect manner. This version strives to be a more professional solution of the problem without all the flaws of the last version.

The aim is to develop the application in an organized, orderly and documented way, using the best available technologies and methods chosen after careful consideration. We need to avoid the glitches and imperfections of the previous iteration, to make the display module more efficient, the interface more user-friendly. We need sustainability, most notably a proper editorial system with authentication to allow the application to grow and fill up with useful information.

Moreover, there will be the new feature of web pages with additional information. They will contain any information seen fit by the editors, giving this project the potential to be a truly exhaustive source of knowledge regarding human bones, ranging from anatomy to pathology. Other new features might be added as well, given a time reserve.

## Chapter 3

### Relevant Technology

#### 3.1 3D Technology for the Web

There are two main directions in development of hardware accelerated 3D for the web: Rendering directly in the browser with HTML5 and against it the traditional plug-in based approach.

Certain technologies allow the application to be a built-in part of an HTML5 web. The main advantage is that users do not need to install any additional software. The application is fully integrated into the web page and its execution is managed by the browser, which might result in slight variations across browsers. In case of 3D technologies, even the use of underlying graphics API depends on the environment, such as browsers using ANGLE with WebGL to utilize DirectX over OpenGL on Windows.

The advantage of plug-ins is that the application will look the same on every device and doesn't need to be tweaked for different browsers. The reason behind that is that the application is executed by the plug-in, not the browser. The disadvantage is that users have to install a plug-in, which is usually an inconvenience and sometimes a real problem, especially on mobile devices. Also, the application is not fully integrated in the web page, which might result in behavior inconsistent with the rest of the web page.

##### 3.1.1 CSS 3D Transforms

CSS3 can apply 3D transforms to elements directly without using HTML5 Canvas. This can be used to create 3D objects and animations.

All geometry is created by transforming rectangular elements, which is very restrictive compared to triangular faces created between vertices. This obstacle can be bypassed by using alpha textures and 3D transforms to create arbitrary shapes. Although it is possible to get creative and assemble complex 3D scenes this way, I feel like this was not the intended purpose and there are tools better suited for the task.

I was not able to find examples demonstrating the use of CSS 3D transforms for displaying complex models comparable to those in our project. Even much simpler scenes often contained graphical errors and did not run smoothly in either Internet Explorer 11 or Firefox 32. The only advantage is superior availability.

##### 3.1.2 WebGL

WebGL (Web Graphics Library) is a JavaScript API for development of interactive 3D scenes for the web. It uses underlying low-level graphics API, typically OpenGL ES 2.0, to make full use of GPU acceleration. It is a royalty-free, cross platform standard maintained by the non-profit Khronos Group. There are frameworks to simplify the use of WebGL, which itself can be considered a low-level API.

WebGL uses HTML5 Canvas to display its content and does not require the use of a plug-in. It will work on any browser with WebGL support without additional software (although it might require appropriate GPU drivers).

The required features such as advanced mouse controls and 3D acceleration are all present and comparable to those of Flash with possible minor benefits (such as consistency of user controls throughout the page).

WebGL is a new technology compared to Flash and other plug-ins. WebGL 1.0 standard was issued in 2011 and in the following years, support by vendors and developers has been growing. Firefox, Google Chrome and Safari have supported WebGL for some time as shown at [8]. Microsoft support starts with Internet Explorer 11 and according to [9] “WebGL is available on all IE11 devices”. Apple announced support on their new iOS 8 (related article at [10]). Default Android browser doesn’t support it, but Firefox and Chrome for Android do.

### ■ 3.1.3 Flash

Adobe Flash is a multimedia and software platform widely used throughout Internet.

Flash uses vector graphics, static or animated, supports streaming of videos, and offers other multimedia related features. The language typically associated with online Flash is ActionScript, although Haxe can be compiled into Flash applications as well. Flash is a proprietary platform of Adobe. Freeware editing software exists, but arguably inferior to licensed Adobe Flash Professional.

Flash requires a plug-in (Flash Player) to run in web browsers, which is enough of a reason to reject many competing technologies. However, the Flash Player is so prevalent today that most users have it installed regardless of our application. Adobe claimed to have 99% penetration on desktop computers in 2011 [11]. That makes the necessity of a plug-in a much smaller issue for our project.

Since version 11.2 (2012), Flash Player allows advanced mouse control such as scrolling or right click [12], which was one of the main features missing in previous versions. Nevertheless, user input might be clashing with the rest of the HTML page, resulting in poor user experience.

Flash Player 11 also introduced Scene3D, an API for development of hardware accelerated 3D content. Scene3D is one of the major candidates for our project’s 3D API. Previous Flash technologies relied on CPU rendering and were significantly slower than GPU accelerated alternatives. Such slower technologies were used in previous implementations of the Atlas.

For several years, numerous writers and developers have claimed that Flash is insecure, a dying technology and a developmental dead end (examples at [13–14]). Despite these voices, Flash is still alive in 2014 and here to stay for some time. Although still widely used, the future of Flash is not all bright. According to statistics at builtwith.com [15], the usage of Flash on major websites has been on the decline lately.

According to [16] Adobe Flash is currently available on most desktop operating systems (Windows, OS X, Linux, Solaris), although the development for Linux and Solaris has been discontinued since Flash Player 11.2 (outside of Google Chrome).

Flash Player for mobile browsers availability is controversial. It is not available on iOS devices (iPhone, iPad...), but it was officially available on Android 2.2-4.0 and BlackBerry (Tablet OS, BB10). However, in 2011 Adobe announced at [17]: “We will no longer continue to develop Flash Player in the browser to work with new mobile device configurations” and admitted that growing support makes “HTML5 the best solution for creating and deploying content in the browser across mobile platforms”. A blog post at [18] confirms that Flash will not be installed on new Android devices and Flash Player for Android will no longer be updated. Apple’s (and formerly Steve Job’s) attitude and lack of support (more information at [19]) also speaks against mobile Flash.

### 3.1.4 Other Plug-in Technologies

There are several other plug-in based systems worth mentioning. All following examples suffer from the necessity of installing a plug-in which is not likely to be pre-installed.

Unity [20] and ShiVa3D [21] are fine examples of multiplatform 3D engines that allow development for desktops, mobile devices and browsers. However, the browser version is not supported on mobile devices, forcing the development of native applications for each mobile operating system (often at a substantial price). That is beyond the scope of this project and browser-based solutions for all platforms are preferred.

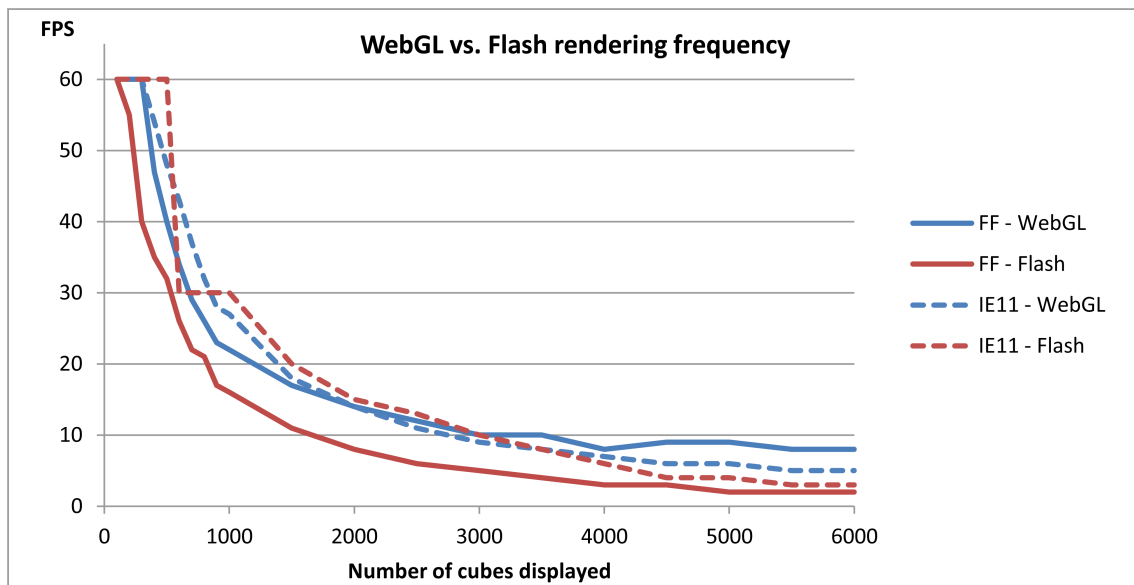
Markup languages such as X3D (and its predecessor, VRML) used to be strictly plug-in based, but lately benefit from GPU acceleration in the form of WebGL through integration models such as X3DOM. When using X3D now, there is little reason to choose a plug-in viewer over WebGL rendering. The support for WebGL (see below) is comparable to that of plug-ins and there is no need of additional installation.

### 3.1.5 Performance Test 1: Cubes (WebGL and Flash)

I carried out a simple performance test to compare the rendering speed of equivalent Flash and WebGL applications.

A demo application (available at [22]) displaying a number of semi-transparent cubes was used, implemented once using Flash Scene3D and once using WebGL. The experiment was run on a Dell Vostro 3460 machine.

The number of cubes was gradually increased while noting FPS values for current number of cubes. There were minor fluctuations in the FPS and recorded values represent estimated medians over a period of several seconds.



**Figure 3.1.** Performance of WebGL (blue) and Flash (red) when rendering a demo application in Firefox 32 (solid) and Internet Explorer 11 (dashed). Frames per second on vertical axis, number of displayed translucent elements on horizontal axis.

It is apparent from the results shown in figure 3.1 that the performance is greatly dependent on the browser.

WebGL in Firefox is by far the fastest combination in very complex scenes. Flash in Firefox is always slower than any other tested combination. In Internet Explorer,



both Flash and WebGL show comparable performance with WebGL gaining in more complex scenes.

Overall, WebGL did better in this performance test, especially with a large number of displayed elements.

It is important to mention that this is an illustrative example rather than a rigorous experiment. The implementation used two different frameworks (Away 3D 4 for Flash, Three.js for Web GL) which might or might not be a source of bias. The performance might also be affected by hardware usage fluctuations in the testing machine.

### ■ 3.1.6 Performance Test 2: Aquarium (WebGL)

To further verify that WebGL can handle complex geometry at reasonable frame rates, a public WebGL demo “Aquarium” (available at [23]) was run at maximum complexity. It displays 4000 models of fish using normal maps, reflections and other effects. The scene geometry easily exceeds 100,000 vertices. Testing on Dell Vostro 3460, the scene was sufficiently smooth, displaying at 12 FPS in Internet Explorer 11 and 14 FPS in Firefox 32. For comparison, current implementation of the Atlas shows significantly worse performance when displaying models of 10,000 vertices.

### ■ 3.1.7 Frameworks (WebGL)

WebGL is a low-level graphics API. Though it is possible to work with it directly, it might be more convenient to use a higher level framework. A list of many available frameworks has been assembled at [24], although not all of them are relevant for our project.

Upon brief research, I came to the conclusion that many of them were interesting solutions, but very few had a large and active community. In fact, out of those frameworks relevant to our project only *three.js* (available at [25]) showed actual signs of active use throughout Internet. While several of the frameworks have tutorials on their websites, only *three.js* can boast a number of external tutorials as well. To verify my assumption, I searched for the frameworks on *StackOverflow* [26], arguably the biggest question & answer site for programmers. The fact that *three.js* was tagged in 4775 questions while no other framework was tagged more than 60 times confirms that *three.js* is likely the most prevalent.

The API of *three.js* seems suitable for our cause. It provides easy access to scenes, objects, cameras, materials, lightning, shaders and more, making the use of WebGL a lot less complex for the developer. The API is documented online.

Considering all this, *three.js* seems to be the perfect framework for our project and there is no need to seek for a novelty solution.

### ■ 3.1.8 Conclusion

CSS3 is an interesting approach with great availability, which is however not very well suited for a complex 3D project. The technology clearly wasn’t meant to render detailed 3D models.

The other technologies offer us what we need: GPU acceleration, sufficient interactivity, quality of user controls and an API suitable for displaying 3D scenes (sometimes using additional frameworks). All options seem to be perfectly viable for our purpose in terms of available features, so the main points to consider are performance and availability.

Availability is probably the biggest concern. Most plug-in based solutions require an inconvenient installation and do not run in mobile browsers. That is a major drawback

that hurts availability and a reason not to choose such solutions. The exception is Flash which is often pre-installed and has partial support in mobile browsers. The other suitable candidate left is WebGL.

Both Flash and WebGL are available on most if not all up-to-date desktop browsers. Neither is available on every mobile device. The trend seems to be obvious though, the use of Flash is on a decline and Flash development shouldn't focus on mobile browsing. On the other hand, more and more companies (including Microsoft and Apple) declare and implement support for WebGL. That is the main reason why WebGL seems to be the better candidate: Its future seems bright, which cannot be said about Flash.

As for performance, simple tests on a laptop favored WebGL over Flash, although the results depend greatly on chosen browser. WebGL also proved its ability to render very complex scenes at reasonable frame rates in an "Aquarium" demo.

After careful consideration, WebGL was chosen as the preferred technology for the 3D model viewer, even though it is not universally available. To simplify the implementation process, *three.js* framework will be used as a higher level API.

## 3.2 Server-Side Language

There are many languages to choose from when developing the server side of a web application.

### 3.2.1 IEEE Spectrum Ranking

To help evaluate the quality of the many languages, I consulted the results of a recent IEEE Spectrum ranking (available at [27]). It takes into consideration *Google* search results, *Google Trends* data, *StackOverflow* questions, demand for jobs on several job sites, *IEEE Xplore* journal articles and more. The top 10 results for web development are shown in figure 3.2 with Java, Python and C# taking the top (in this order).

Language Rank	Types	Spectrum Ranking
1. Java	🌐 📱 🖥️	100.0
2. Python	🌐 🖥️	93.4
3. C#	🌐 📱 🖥️	92.3
4. PHP	🌐	84.7
5. Javascript	🌐 📱	84.4
6. Ruby	🌐	78.8
7. PERL	🌐 🖥️	70.3
8. HTML	🌐	65.3
9. Scala	🌐 📱	63.0
10. Go	🌐 🖥️	60.5

**Figure 3.2.** Top 10 web development languages, ranked by IEEE Spectrum in 2014 (taken from [27]).

### 3.2.2 Personal Experience

An important part of choosing the right language is my prior experience: I was introduced to PHP several years ago and have not used it much since. Later I worked on a desktop project in .NET C#. Throughout university, a lot of my education involved Java, although I never used it in non-academic projects or web applications. Judging



by my brief experience, I can say that I dislike PHP. My opinion is that it seems inconsistent, full of small surprises and traps for an inexperienced user. A more elaborate criticism by a more experienced developer is available at [28].

On the other hand, Java and C# felt like professional tools and I did not mind working with either, although I preferred Java for subjective reasons. An objective reason would be that it is multi-platform and open-source with several free IDEs to choose from.

However, I have not developed a web application in either, so I looked for educated opinions online. Most comparisons and reviews agree that Java and .NET for web are comparable and the choice is mostly preferential (examples at [29–30]).

### ■ 3.2.3 Conclusion

Considering that Java was ranked the best language for web development by IEEE Spectrum and at the same time is my preferred language from personal experience, the site will be developed in Java.



## Chapter 4

### Application Design


TODO



## Chapter 5

### Implementation


TODO



## Chapter 6

### Testing

TODO



## Chapter 7

### Conclusion

TODO

## References

- [1] *Skelet 3D*.  
<http://skelet3d.lf3.cuni.cz/>.
- [2] *Skeletopedia*.  
<http://skeletopedia.sk/>.
- [3] *Zygote Body*.  
<http://www.zygotebody.com/>.
- [4] *Anatronica*.  
<http://www.anatronica.com/>.
- [5] *3D Science - Skeletal Models*.  
[http://www.3dscience.com/3D\\_Models/Human\\_Anatomy/Skeletal/](http://www.3dscience.com/3D_Models/Human_Anatomy/Skeletal/).
- [6] *UPOL - Atlas člověka*.  
<http://www.atlascloveka.upol.cz/cs/cs02/cs0201/cs020100.html/>.
- [7] *Wikipedia - List of Human Bones*.  
[http://en.wikipedia.org/wiki/List\\_of\\_bones\\_of\\_the\\_human\\_skeleton](http://en.wikipedia.org/wiki/List_of_bones_of_the_human_skeleton).
- [8] *Can I Use - WebGL*.  
<http://caniuse.com/#feat=webgl>.
- [9] *Internet Explorer Dev Center - WebGL*.  
<http://msdn.microsoft.com/en-us/library/ie/bg182648%28v=vs.85%29.aspx>.
- [10] Stephen Shankland. *CNET - iOS 8 brings big boost for Web programmers*.  
<http://www.cnet.com/news/ios-8-brings-big-boost-for-web-programmers/>.
- [11] *Adobe - Flash Penetration Statistics*.  
<http://www.adobe.com/cz/products/flashplatformruntimes/statistics.html>.
- [12] Tom Krcha. *Using MouseLock, Right Click, Middle Click features in Flash Player 11.2*.  
<http://tomkrcha.com/?p=2621>.
- [13] Marcus Kruger. *Wired - Flash Is Dead*.  
<http://archive.wired.com/insights/2014/05/flash-dead-long-live-webgl/>.
- [14] Serdar Yegulalp. *Infoworld - Adobe Flash: Insecure, outdated, and here to stay*.  
<http://www.infoworld.com/article/2610420/adobe-flash/adobe-flash--insecure--outdated--and-he.html>.
- [15] *BuiltWith - Shockwave Flash Embed Usage Statistics*.  
<http://trends.builtwith.com/framework/Shockwave-Flash-Embed>.
- [16] *Adobe - About Flash Player*.  
<https://www.adobe.com/software/flash/about/>.
- [17] Danny Winokur. *Adobe News - Flash to Focus on PC Browsing and Mobile Apps*.  
<http://blogs.adobe.com/conversations/2011/11/flash-focus.html>.

- [18] Tareq Aljaber. *Adobe Blogs - An Update on Flash Player and Android*.  
<http://blogs.adobe.com/flashplayer/2012/06/flash-player-and-android-update.html>.
- [19] *Wikipedia - Apple and Adobe Flash controversy*.  
[http://en.wikipedia.org/wiki/Apple\\_and\\_Adobe\\_Flash\\_controversy](http://en.wikipedia.org/wiki/Apple_and_Adobe_Flash_controversy).
- [20] *Unity 3D*.  
<http://unity3d.com/>.
- [21] *ShiVa3D*.  
<http://www.stonetrip.com/>.
- [22] Felix Turner. *Airtight Interactive - Stage3D vs WebGL Performance*.  
<http://www.airtightinteractive.com/2011/10/stage3d-vs-webgl-performance/>.
- [23] *WebGL Samples - Aquarium*.  
<http://webgl.samples.googlecode.com/hg/aquarium/aquarium.html>.
- [24] *WebGL User Contributions*.  
[https://www.khronos.org/webgl/wiki/User\\_Contributions](https://www.khronos.org/webgl/wiki/User_Contributions).
- [25] *three.js*.  
<http://threejs.org/>.
- [26] *Stack Overflow*.  
<http://stackoverflow.com/>.
- [27] *IEEE Spectrum - The Top Programming Languages*.  
<http://spectrum.ieee.org/static/interactive-the-top-programming-languages>.
- [28] *PHP: a fractal of bad design*.  
<http://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/>.
- [29] Walker Rowe. *Southern Pacific Review - Choosing Java Vs .Net*.  
<http://southernpacificreview.com/2013/08/08/choosing-java-vs-net-for-web-development/>.
- [30] Srinath Davu. *Segue Technologies - .NET vs Java*.  
<http://www.seguetech.com/blog/2013/06/03/dotnet-vs-java-how-to-pick>.