Elearning and mobile devices – technical problems and possible solutions

Robin Horák, Jiří Hrbáček
Department of Technical Education and Information Science, Faculty of Education, Masaryk University, Brno, Czech Republic
robin.horak@mail.muni.cz, hrbacek@mail.muni.cz

***Abstract*—The rapid expansion in the use of tablets and other mobile devices pushes away personal computers and laptops. This statement is especially true in the case of regular users, who use these devices primarily for entertainment and occasional communication via email, for example. More advanced users, among which we can place also students, need personal computers for work, writing essays, or for using specialised programs, depending on the field of study. Even so, many students own both the computer and a mobile device, such as a tablet. It is therefore not unusual that a student chooses to use the tablet for elearning because of more comfortable reading or the possibility to study anywhere and anytime. Unfortunately, most mobile devices don't support technologies needed for displaying interactive applications or complicates playing videos in an internet browser. The aim of this paper is to present possible solutions that can make the elearning content available for the vast majority of users regardless of the device used.**

I. INTRODUCTION

One of the most frequently mentioned benefits of elearning is interactivity, which means that a student is being activated during his or her elearning studies [1]. The student can be activated, for example, by implementing various multimedia elements. The use of multimedia element allows perception using more senses and helps to remember information [2].

Nowadays, thanks to the spread of mobile devices, the situation regarding the use of multimedia is complicated to some extent. While until recently the majority of users used mainly personal computers, where there were not many complications when using multimedia, a significant number of elearning users now use tablets or other mobile device to access elearning. The problem is that mobile devices very often don't allow displaying multimedia content in browser in the same way as in the case of personal computers.

A. Advantages of mobile devices in elearning

Elearning study using mobile devices has its advantages. For example, reading from tablets can be more comfortable than reading from the computer screen. A research focused on the problem of reading from the computer screen showed that two thirds of students printed out a long text before even reading it. Moreover, most of the remaining third of students printed out the text after the first reading [3]. Elearning study using mobile devices also supports another of the fundamental ideas of elearning, which is an unlimited access to information, thus not limited by time and place [4]. We should support the elearning study using mobile devices not only because of its benefits, but also because we can't afford to ignore these users anymore.

As it was already mentioned, the possibility of elearning study using mobile devices brings considerable technical difficulties to creators of multimedia content. Traditionally, various educational applications for elearning have been developed as Java applets or Flash applications, and video playback has been ensured by various plugins (such as Windows Media Player Plugin). Later, Adobe Flash Player Plugin (with various video players created in Adobe Flash) has been widely used (FLV, F4V, and MP4 video formats). Unfortunately, the vast majority of mobile devices do not support any plugins, which is as a trend introduced by Steve Jobs, co-founder and former CEO of Apple. Steve Jobs was known to be a great opponent of the Flash Platform, which he sharply criticized in an open letter "Thoughts on Flash", written in 2010. He also criticised Java: "Java's not worth building in. Nobody uses Java anymore. It's this big heavyweight ball and chain."

II. POSSIBILITIES OF CREATION AND USE OF MULTIMEDIA APPLICATIONS

What are our possibilities, if we want to deliver our elearning content to users with various devices?

A. HTML5

HTML5 has been around for quite a long time and gained a lot of popularity especially among those without a deeper insight to this technology. The truth is that if we want to create multimedia applications, we have to use JavaScript together with HTML5. Using JavaScript, we are able to draw graphics on HTML5 Canvas, which is one of the most important elements that came with HTML5.

HTML5 together with JavaScript and CSS3 is a good choice when we want to create web pages with interesting effects and animations, but the creation of more complex interactive application is a little bit complicated. The problem is that there are still not suitable tools for HTML5 that could replace Adobe Flash in the field of animation, for example. Even though Adobe has been working for some time on Adobe Edge Animate tool, which allows visual creation of animations in HTML5 that can be scripted using JavaScript, its possibilities are comparable with early versions of Adobe Flash. There is also an extension for Adobe Flash called Tool for CreateJS, which allows converting Flash animations to HTML5 and JavaScript. However, the export possibilities are (and will probably be for quite a long time) very

978-1-4799-2162-1/13/$31.00 ©2013 IEEE

limited. We are definitely not able to convert a complex application coded in ActionScript (which is the programming language of Flash) into JavaScript and HTML5. On the other hand, we are able to convert animations created in Adobe Flash and add JavaScript, which is still much easier than animate by writing JavaScript code.

The great advantage of HTML5 technology is that it can be used almost in the same way both at personal computers and mobile devices, if it is supported. However, we have to expect quite different results depending on internet browsers.

If we decide not to use HTML5 for creating multimedia applications, either because of more demanding creation, or lower support especially from older internet browsers, we can create several versions of our multimedia application for other platforms. That means, for example, to create a Flash application or Java applet, and another versions for iOS (Objective-C) and Android (Java) in their native development environments. Obviously, this brings enormous extra work and no one would probably take this route.

B. *Adobe Flash*

It is not very widely known that Adobe Flash (since CS 5.5 version) allows creating applications both for Android and iOS (see Fig 1).

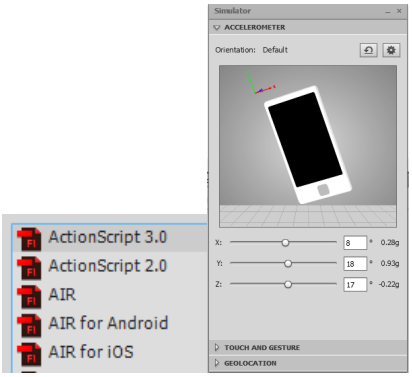

Figure 1. Possibilities of creation for iOS and Android

It is important to mention that we are not talking about applications for web browsers using Adobe Flash Player plugin, because plugins in general are often not supported on mobile devices. Adobe Flash allows exporting applications, that could be downloaded and installed from App Store, in the case of iOS, and from Google Play (formerly Android Market), in the case of Android. This could save us a lot of effort, because we could create our multimedia application in one development environment and just modify it to fit the other platforms. Nevertheless, we should count with possible limitations depending on the target platform and it is advisable to create the application universally so we could avoid possible demanding modifications.

If we want to create an application in Flash for iOS, for instance, we are not able to use externally loaded SWF files. Even though we can export SWF files as SWC libraries and include these libraries in our application, it seems to be more comfortable just to create the whole application as one SWF file, if it is possible.

In most cases, we will also have to modify the user interface to fit the target devices. We should also support gesturers for easier and more comfortable control on mobile devices, such as pan, rotate, swipe, zoom, tap, two finger tap, press and tap, etc. However, If we won't use gestures for mobile devices, the mouse events we used in our application will be available – the finger tap will behave in the same way as the mouse click, the mouse drag will be the same as if we would drag using a finger, and so on.

We can then embed the Flash Application in our HTML webpage, either using some tool in our LMS, or just use the HTML code published from Flash. After the block of code for embedding the application, we can insert some script, for example in JavaScript, that will detect the system of the user and provide a link either to App Store (iOS), or Google Play (Android), where the application can be downloaded (see Fig 2).

```
<script type="text/javascript">
isMobile = {
    Android: function() {
        return navigator.userAgent.match(/Android/i);
    },
    iOS: function() {
        return navigator.userAgent.match(/iPhone|iPad|iPod/i);
    },
}

if(isMobile.iOS()){
    document.write('iOS - link to the app in App Store');
}
else if(isMobile.Android()){
    document.write('Android - link to the app in Google Play')
}
</script>
```

Figure 2. Detection of mobile operating system in JavaScript

C. *Unity*

Another option that we can use for creation of multimedia applications is Unity. This development environment is more suitable for creation of 3D applications, and apart from other important aspects, it contains a very good physical engine, which can make our work easier in some cases, because neither Flash nor HTML5 has a built-in physical engine. Even though that some external physical engine can be used (such as Box2D), everything has to be done via scripting and not by using UI as in the case of Unity. The same is for 3D in the case of HTML5 and Flash – there is no built-in truly 3D engine. We can use engines such as Papervision3D, Alternativa3D, or Yogurt3D for Flash, but we won't see anything in 3D until we compile the application. In the case of Unity, we can easily import 3D models created in Cinema 4D or Blender, for instance, and we can directly manipulate with these models in Unity environment. We can then add functionality using JavaScript or C#.

We can use the basic version of Unity for free. Even though the free version of Unity has some limitations comparing with the Pro version, we wouldn't probably use these additional features if we were not creating a 3D game.

As in the case of previously mentioned Adobe Flash, we can export our application for web, iOS and Android platforms. There are also publishing possibilities for other platforms, as the figure shows (see Fig 3).

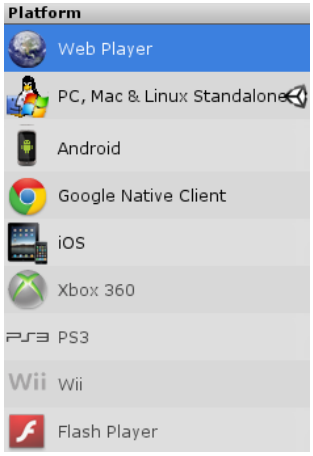

Figure 3. Export possibilities of Unity

As we can see, there is also an export option for Flash Player, which could be interesting for us. Unfortunately, this option is not available anymore because Unity decided to stop supporting it (they only provide bug fixes for those who bought the licence for this export possibility).

III. VIDEO PLAYBACK POSSIBILITIES

A. *Video formats*

If we want to use videos that can be played on various devices, we will not avoid using more video formats. Even though new versions of all major internet browsers support HTML5 video, there are some disputes over video formats and we have to use MP4 video format using H.264 video codec and ACC or MP3 audio codec, and we also have to use either WebM format (VP8 video codec, Vorbis audio codec) or Ogg format (Theora video codec, Vorbis audio codec) to cover all the major internet browsers. Not long ago, even Mozilla Firefox didn't support MP4 because of hopes that WebM will be widely used. At one point, Google announced removing it from Chrome browser because of the patent-encumbered H.264 codec, and, at the same time, Microsoft announced adding it into Internet Explorer. The current situation is that all the latest versions of the major internet browsers, apart from Opera browser, support H.264. Firefox added H.264 support in version 22, and Internet Explorer in version 9. Unfortunately, not every user uses the most recent browser versions, so there will be a lot of outdated browsers that don't support H.264, or even HTML5 video at all.

It is claimed that HTML5 video can be played on 85 % of devices. For the rest 15 %, we have to provide an alternative, which could be Adobe Flash. Flash video players use Adobe Flash Player plugin, which is supported on 99 % personal computers, but often not supported on mobile devices. Flash video player can use FLV, F4V, or MP4 video formats, where F4V an MP4 can be used since Plugin version 9.0.115 (current version is 11).

Both F4V and MP4 (both MPEG-4 standard) use H.264 video codec and MP3 or AAC audio codecs. FLV uses On2 VP6 video codec and usually MP3 audio codec. Generally, F4V and MP4 videos can achieve a better quality than FLV when the video size remains the same.

B. *H.264/MP4 in detail*

The advantage of using MP4 is that we can use it both for HTML5 and Flash video. Nevertheless MP4 playback can be problematic because the video metadata are often placed at the end of the file after export. This means, that we have to wait until the whole video is loaded before we can start playing it. If we look more closely at this issue, each video in MP4 format contains a moov atom (movie atom; container for all the metadata). This movie atom contains an mvhd atom (movie header atom), that defines video duration, video dimensions, etc. [5]. The mentioned moov atom is basically an index of the video file, which must be loaded before we can even start playing the video. Unfortunately, this moov atom is often placed at the end of the video file, and a user has to wait until the whole video is loaded, which is very inconvenient especially for users with slower internet, and when the video size is large.

If we don't have problems with streaming after export from our favourite video editing or converting software, we can use MP4. If we have problems, we can try finding options such as "progressive download," "fast start" or "use streaming mode" in our software, which places moov atom at the beginning of the video file [6].

C. *Use of popular video servers*

We can try saving ourselves some work by using services such as from YouTube or Vimeo, where we can, after registration, upload our videos. We don't have to worry about video formats because many video formats are supported by these servers and the uploaded videos are automatically converted to suitable formats for web. After that, we can just embed the generated HTML/JavaScript code in our web page. This code ensures that our videos will be playable at large spectrum of devices. Primarily, a Flash video player is used, if it is possible. If not, HTML5 video player is used instead. It is noteworthy, that Vimeo uses only H.264 video, so we will not be able to play the video in some cases, where Flash is not available and the internet browser doesn't support H.264.

If we want to use these services, we have to count with several limitations, such as with a limited space for our videos, limited maximal size of a video, limited video upload per day, etc. If we decide not to use the services of YouTube or Vimeo, for example because we want to have our videos uploaded on our server, we can use some of the widely used video players, such as JW Player, flowplayer, etc. These video players also ensure that in the case of lack of support of Flash, or HTML5, the other technology will be used. Even though the basic editions of these players are free, we are limited in certain aspects, such as we must have a watermark in the corner, we are not able to customize UI, etc. In order to unlock additional features or remove the watermark, we have to buy a licence.

D. *Custom Flash/HTML5 playback*

More advanced users can use their own Flash video player with a HTML5 alternative. If we are able to embed Flash SWF files, which is also the case of a video player created in Flash, we can just replace the default alternative content (for the case that the required version of Flash plugin is not available), with the HTML5 video player (the default alternative content is usually a link to download the latest version of Flash Player). In this case,

if any user doesn't have the required version of the Flash Player Plugin installed, the HTML5 video should start playing. But if the browser of the user doesn't support HTML5 either, no content would be displayed. For this reason, we place a link to download the latest version of Adobe Flash Player at the end of the video tag (see Fig 4).

```
<video width="640" height="360" controls>
    <source src="video.mp4" type="video/mp4">
    <source src="video.ogg" type="video/ogg">
    <a href="http://www.adobe.com/go/getflash">
    Please download the latest Adobe Flash Player.
    </a>
</video>
```

Figure 4. Alternative content in HTML5 video tag

We can also use primarily HTML5 player that would be replaced by Flash player, if HTML5 video could not be used. In this case, we would place the code for embedding SWF files at the end of the HTML5 video tag. The default code for embedding the SWF file would also provide the alternative content which is the link to download the latest version of Flash.

Nevertheless, the first mentioned way is still more suitable, because it is claimed that Adobe Flash Player is still more spread than HTML5 among personal computers, but this may change soon.

IV. CONLUSION

As the paper demonstrated, the use of multimedia can be problematic nowadays if we want to target the largest possible spectrum of devices, including mobile devices. We can't afford to ignore these users of mobile devices, such as tablets, anymore, because the number of these users will be only increasing, while the use of personal computers may stagnate or even decrease. The future may bring a simple and an unified solution, but now we are at some breaking point among several competing platforms and technologies, as we could have seen in the case of disputes over patent-encumbered H.264 video codec and open VP8 video codec (and major browsers adding one, removing the other, and vice versa). Unfortunately, not always the better technology wins, but we will have to adjust either way.

REFERENCES

[1] ZLÁMALOVÁ, Helena. Distanční vzdělávání a eLearning: učební text pro distanční studium. Vyd. 1. Praha: Univerzita Jana Amose Komenského Praha, 2008, 144 s. ISBN 978-808-6723-563.

[2] KOPECKÝ, Kamil. E-learning (nejen) pro pedagogy. Vyd. 1. Olomouc: Hanex, 2006. 130 s. ISBN 80–85783–50–9.

[3] MOORE, Michael G. Handbook of Distance Education. Vyd. 2. United States: Lawrence Erlbaum Associates, 2007, 690 s. ISBN 0-8058-5847-4.

[4] WHITE, Cynthia. Language Learning in Distance Education. Cambridge : Cambridge University Press , 2003. ISBN 9780521815413.

[5] ISO/IEC 14496-12:2005(E). Coding of audio-visual objects: ISO base media file format. Geneva: ISO/IEC, 2005.

[6] Understanding the MPEG-4 movie atom. LEVKOV, Maxim. Adobe Developer Connection [online]. 2010 [cit. 2013-09-20]. Available from : http://www.adobe.com/devnet/video/articles/mp4_movie_atom.html