Towards Framework for Choosing 360-degree Video SDK

Antti Luoto
Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 10, FI-33101 Tampere, Finland
antti.l.luoto@tut.fi

Keywords: 360-degree video, SDK, Software Development Kit

Abstract: 360-degree videos are gaining popularity among consumers. Still, software developers are early adopters of technology so it is important to map their needs for 360-degree video development. They use software development kits that help creating software on the 360-degree video software domain. We want to find out which factors developers need to take into account when choosing these software development kits. In this position paper we describe a preliminary 360-degree video SDK choosing criteria, based on literature and our own experiences, which we plan to evaluate with a survey.

1 INTRODUCTION

360-degree videos, also known as spherical videos, are getting more popular (Alface et al., 2011). Applications for 360-degree videos can be found for example from entertainment, industry, surveillance, and robotics. One of the reasons for popularity is that head mounted displays (HMD) supporting 360-degree video and virtual reality (VR) have become easily available for consumers. HMD is a wearable display device which consists of an optical system in a helmet with displays located in front of user's eyes creating an illusion of depth (Shibata, 2002). They are applicable for presenting interactive spatial information such as 360-degree videos or VR worlds. HMDs include dedicated display devices like Oculus Rift and HTC Vive in addition to mobile phone devices attached to headsets like Google Cardboard/Daydream and Gear VR. 360-degree videos are also getting more popular in web applications. For example both Youtube and Facebook support 360-degree videos.

360-degree video domain often requires reading the sensors of HMDs and calculating sphere mathematics. Luckily, there are Software Development Kits (SDK) that help developers with such tasks. We define SDK by using the definition by (Palme et al., 2010): "The SDK is as a set of development tools that allows a developer to create applications for a certain software package and hardware platform. SDKs differ in terms of their programming code languages, their libraries and API support." Thus, 360-degree video SDK is an SDK that allows developing 360-degree video applications. A 360-degree video SDK can, for example, offer an API that helps creating a video player which can detect the head movements in spherical videos both in monoscopic or stereoscopic mode. Examples of 360-degree video SDKs include Google VR SDK, OZO player SDK and KRPano. Comparing those SDKs and finding the best one for development work can require a considerable amount of time.

360-degree video SDKs offer many useful features but naturally they have different specifications and features, and they cannot offer everything for everybody. For example, our experience is that some 360-degree video SDKs for mobile devices do not allow easy development of user interface (UI) elements such as user interaction points. However, some of those 360-degree video SDKs can be integrated with a popular game engine called Unity 3D. With the help of Unity 3D or similar game engines, UI elements can be added more easily on top of 360-degree videos. Still, using complex tools such as Unity 3D can require more resources. For example, (Linowes and Schoen, 2016) state that an empty Unity 3D scene for Android requires a much bigger application package than a simple native code application, which has an increasing effect on memory and battery consumption.

The motivation for the work is that we have use cases for 360-degree videos so we need to choose a convenient SDK to work with. Further, we noticed that there seems to be relatively few scientific publications about choosing 360-degree video SDKs. The study aims to gain knowledge about the growing field, and the topic is significant for developers who need

a scientifically structured criteria for choosing 360-degree video SDKs, for example for inspecting development tools.

Our research questions are:

- Which criteria is important when software developers choose SDKs for 360-degree video application development?
- What features software developers hope for 360-degree video SDKs to have?

To answer these questions we present a preliminary criteria for choosing an SDK from the developer's viewpoint. Our study can be seen as a contribution to developer experience studies.

The structure of the paper is as follows. Section 2 presents the scientific background for our work. Section 3 describes the identified criteria for choosing a 360-degree video SDK. Section 4 analyses the criteria and planned survey from a critical point of view. Section 5 concludes the paper and describes our on-going and planned research.

2 BACKGROUND

Our work can be seen as a part of developer experience studies. According to (Fagerholm and Münch, 2012) developer experience "consists of experiences relating to all kinds of artifacts and activities that a developer may encounter as part of their involvement in software development." They define three developer experience categories: development infrastructure, feelings about work and the value of one's own contribution. Our study relates mostly to the first one since SDKs are a part of development infrastructure.

Since 360-degree video software development has similarities to VR software development, (Bierbaum and Just, 1998) offers the most related background for our work. According to them, the primary requirements for VR development environment include performance, flexibility, and ease of use. On a more detailed level they list required capabilities and factors such as cross-platform development, support for VR hardware, high-level and low-level interfaces, programming languages, user interaction, minimal limitations, and choosing between commercial and open solutions, all of which we also included in our criteria. They discuss about the whole development environment where as we concentrate more on choosing only the SDK. The following background describes research about how developers choose their SDKs on other domains (than 360-degree video domain) and what they need to consider when doing that.

(Palme et al., 2010) propose a six-dimension benchmark (security, individual and organization buyer choice, market growth, ease of implementation and net revenue) for choosing a smart phone operating system (OS) for mobile application development. The dimensions are based on their own opinion about the most critical ones. They take SDK related viewpoints into account for example by saying that the OS SDK for Android is related to ease of development and market availability. They also note that the license of an SDK can have an effect on the usage decision. We did not include security in our criteria because we think that security is not among of the most essential characteristics for a 360-degree video SDK. If there are security related requirements, the other software components should offer the solution for most of them.

(Nykaza et al., 2002) interviewed developers about their needs for SDK documentation. They studied one particular documentation and give a detailed analysis of useful documentation features such as necessary content, taking into account the target audience and prerequisite knowledge, etc. Though documentation is an important part of using an SDK, we are not interested only about documentations. Documentation is however related to "ease of implementation" found in our criteria.

(Dalmasso et al., 2013) acknowledge the problem of the variety of different platforms and SDKs. They formed a classification and a comparison for different cross platform development tools for mobile platforms. The classification is based on general desirable requirements identified by themselves. The criteria they use has some similar aspects to our work. For example, they included the SDK's ability to be used for multiple platform development (also mentioned in our interview) as a part of their criteria. Additionally, they discuss performance in terms of CPU, memory and power usage, which we approach from the low resource consumption point of view.

(Argyriou et al., 2016) discuss about the challenges of designing UI in mixed 360-video and game environment using Google VR SDK and Unity 3D. Though they do not concentrate on creating choosing criteria of different SDKs, they are interested about user interaction which is also taken into account in our criteria.

3 CRITERIA FOR CHOOSING 360-DEGREE VIDEO SDK

We present a criteria about the factors software developers need to concider when choosing a 360-degree

video SDK. The criteria is based on scientific literature, an interview with a 360-degree video application developer from industry (with seven years of 360-degree video application development experience), our own experiences (six months with Google VR SDK for Android, Nokia OZO SDK and Unity 3D). The interview with an expert was helpful especially from the web development perspective. Additional background for the criteria comes from our discussions with industry partners that have made us interested in some particular use cases such as user logging and user interaction.

3.1 Platforms, Domains and SDKs

As Table 1 presents, there are multiple platform alternatives for 360-degree video applications. Every SDK can not support every platform but for example web browser applications can be run in different kind of environments and Cardboard/Daydream applications can be run on different mobile devices (Android and iOS). There are also many fields for 360-degree video applications such as education, entertainment, industry, and research, which can have an effect on the desired characteristics of an SDK. Further, Table 2 lists different 360-degree video SDKs. We might be missing some SDKs but the list works as an example of different alternatives and it reflects the difficulty of choosing the best one for development work from many options.

Table 1: Example platforms for 360-video applications

Platform
Web browser
Cardboard/Daydream
Windows
Linux
macOS
HTC Vive
Gear VR
Oculus Rift

3.2 Features and Characteristics of SDKs

The combination of different features and characteristics of SDK can be the most important reason for choosing one. Table 3 lists the features and characteristic that are based on our own development experience, knowledge gained from an interview with an expert from industry and aspects found from the research literature. There are naturally other features and characteristics as well but either we have experience or we have found scientific background for the chosen ones. The features and characteristics in the list are elaborated in the following paragraphs.

User interaction. For immersive experience, a convenient way to implement user interaction, for example by adding an embedded UI on top of 360-degree video, can be an important factor. Probably the development of UI requires some kind of graphical API. However, not every SDK offers a possibility to add user interaction points easily.

Minimal limitations. The usage of an SDK should not be restricted only to the ready-made features. A skillful programmer should be able to extend the functionality if needed.

Performance. While even modern mobile devices are powerful enough for showing 360-degree videos, the performance requirements can increase for high resolution videos with a high refresh rate including other computation. An SDK should provide sufficient performance for comfortable immersive experience. Some SDKs include performance monitors but it is probably not the most essential feature for a 360-degree video SDK if the development environment otherwise includes a performance monitor.

VR hardware support. HMDs can be integrated to different kinds of devices that help for example with navigation and user interaction. For example Google Daydream supports a decicated controller that can be used for pointing and clicking.

Low-level API. In addition to high-level interface, an SDK can offer a low-level interface. With low-level source code the developers can make applications that perform faster or use less resources. For example, Google VR SDK for Android comes with a native developer kit (NDK) that is less restricted than the Java SDK but requires knowledge about C and C++. On the other hand, development work's abstraction level can be even increased for example with Unity 3D integration.

Programming language. The programming language of an SDK can have an effect on the usage decision. Some programmers are more familliar with some languages or the platform can require a certain language. For example a high-level language can support cross-platform development better or Javascript can be needed for web development.

Multiple platforms. Often creating a application for a single platform is not enough but it the implementation is needed for other platforms as well. An example of multiple platform support is Google VR environment that is provided for Android and iOS in addition to integration with Unity 3D and Unreal game engines and support for web applications.

Table 2: Example 360-degree video SDKs

SDK	Developer	Platform
Google VR SDK	Google	Android, iOS, Unity 3D, Unreal, Web
OZO player SDK	Nokia	Android, iOS, Oculus Mobile, SteamVR
OpenVR SDK/SteamVR SDK	Valve	Multiple vendors
OSVR SDK	Open source	Open Source VR headset
VR One SDK	Zeiss	VR One
krpano	krpano	Web
Pano2VR	Garden Gnome Software	Web, Cardboard
Marzipano	Open source	Web
Oculus SDKs	Oculus VR	Rift, Gear VR, Unity 3D, Unreal, Web, PC

Table 3: Different characteristics or features of 360-degree video SDKs with background

Feature or characteristic
User interaction (Argyriou et al., 2016)
Minimal limitations (Bierbaum and Just, 1998)
Performance (Bierbaum and Just, 1998)
VR hardware support(Bierbaum and Just, 1998)
Low-level API (Bierbaum and Just, 1998)
Programming language (Bierbaum and Just, 1998)
Multiple platforms (Dalmasso et al., 2013)
Low resource usage (Dalmasso et al., 2013)
Content management (Interview)
Web support (Interview)
Access to sensor data (LaValle et al., 2014)
Viewport tracking (LaValle et al., 2014)
Multiple 360 video formats (Own experience)
360 video format detection (Own experience)
DRM protection support (Own experience)
Free / Open source license (Palme et al., 2010)
Ease of implementation (Palme et al., 2010)
Market situation (Palme et al., 2010)

Low resource usage. Playing 360-degree videos can require relatively much computing power for mobile devices. With low resource usage we mean the SDK's ability to keep CPU, memory and power usage on minimal level for example for saving battery resources.

Content management. Some 360-degree video use cases are related to content management. For example a 360-degree video can be a part of an educational web page managed with a content management system. Thus, SDKs could support embedding 360-degree videos in varying content environments. On the other hand, content management inside 360-degree videos can be important as well. For example, sometimes it would be useful to add text on top a video or highlight a part of the it according to associated metadata.

Web support. While many 360-degree video applications are made for mobile devices, a support for web applications can be more important in the future when 360-degree videos become more popular in web.

Access to sensor data. An SDK can support different ways to access sensor data for head tracking. For example, the the head orientation can be retrieved in many formats such as euler angles, quaternions or matrix data. In addition, accessing accelerometer can be needed.

Viewport tracking. With viewport tracking we mean the video player's ability to automatically adapt to user's head orientation. For HMD usage this is basically a required ability but for web applications it can be preferable to navigate with mouse dragging.

Multiple 360 video formats. 360-degree videos come in multiple formats so an SDK should support as many as possible. They can be monoscopic (each frame is monocular equirectangular panorama) or stereoscopic (two vertically-stacked equirectangular panoramas) or the video be can stream, MPEG-4, webm etc.

360 video format detection. In addition to be able to play different 360-degree videos, it would be helpful for an SDK to detect the video format. For example, when using the class VRVideoView of Google VR SDK for Android, it is required to set the video format (monoscopic or steroscopic) in the program code because the player can not detect the video format by itself.

DRM protection support. As traditional videos, 360-degree videos can be protected with digital rights management (DRM) techniques. Not all SDKs offer playback for DRM protected videos.

Free / Open source license. The license of an SDK can affect the usage decision. For example, an individual developer getting familiar with field might want to start with a completely free SDK while a company might want to pay for non-restricted usage.

Ease of implementation. When choosing tools for software development work, the ease of implementa-

tion can be an important aspect. Ease of implementation includes things such as good documentation, familliar technologies, the quality of an API, etc. We include the easiness of integration with other components such as OS's and game engines under this category.

Market situation. Market situation can have an effect on the SDK usage decision. For example new devices can have new features that have the charm of novelty. In addition, the organization's strategy can determine the used platform.

4 DISCUSSION

When starting to work with 360-degree video development, we realized that choosing an SDK for 360-degree video application development is not an easy task. Additionally, we could not find a proper scientific criteria for choosing an SDK. Since 360-degree videos are a growing phenomena, a criteria for choosing the tools for the development work is beneficial for multiple parties.

To evaluate our criteria, we plan to conduct a survey. We chose the method because it is inexpensive, we hope to reach a large respondent group and the responses are anonymous. Disadvantages in using surveys include the inflexibility of the survey form and the lack of human interaction. Piloting the survey beforehand is important.

We plan to make an online survey with Google Forms. Primarily, we will call for participants from an association called Virtual Reality Finland that supports the development of the VR and AR ecosystem in Finland. The survey is planned to be lightweighted and not requiring much time to answer (10-15 minutes). According to plan, the survey has 10 questions divided to four sections: five multiple choise questions, four open field questions and one grid question with a 5-step answer scale. We aim to keep the questions short and simple.

A good survey should be clear, easy to follow, and provide enough information for respondents. There is a danger that some respondents do not understand what we mean with the questions. Answering to the open text questions can be more difficult for some respondents, so is it possible that they will leave the open field empty, even though open text answers could give the most interesting insight for us.

Our own development experience was limited to working with Google VR SDK for Android, Nokia OZO SDK and Unity 3D with simple applications. For that reason, we wanted to interview experts from the industry. However, we only managed to conduct one interview because it turned out to be difficult to get interviews from industry experts. That is one of the reasons we try to reach for a larger respondent group with an online survey.

The group of survey respondents can be expected to be quite exclusive since only developers with experience about 360-degree video SDKs are able to answer. Not any software developer can give proper insight on the topic. Therefore, getting a large enough response set for making meaningful research is not an easy task.

The answers will be analyzed statistically. Open answers naturally require more preparation for analysis but at first we intend to categorize them for further quantitative analysis. We also hope to get enough material for qualitative analysis.

Our criteria reflects our own interests to some extent. We are most interested in some particular use cases (like user logging and user interaction). However, we did not want to restrict the criteria only to those topics but we wanted to gain more wide view on the field, and we found support from the literature for many aspects. On the other hand, we assume that some interesting and important aspects were not included. Therefore, we hope that the planned open questions in the survey will give insight on those factors.

While the criteria and the survey is not the main goal of our research project, it is an important first step to gain knowledge about the field. We realize that there is a gap in current research not providing enough knowledge about developer experience in 360-degree video development. Studying software developers is important because, for example, (Yucel and Edgell, 2015) state that software developers invent uses for devices popular in future and they act as early adopters of technology, so their preferences can have effects on early market advantages.

5 CONCLUSIONS

In this position paper we presented a preliminary criteria for choosing a 360-degree video SDK. The criteria is based on research literature, our own experiences and an interview with an expert on the domain. To evaluate our criteria, we plan to conduct an online survey for software developers working in the field of 360-degree videos. Our eventual goal is to find out on which criteria software developers choose 360-degree video SDKs and what features are expected from them.

The motivation for the work comes from the need to sort out the field for further development of 360-

degree video applications. We need to choose a proper SDK for our use cases which include user interaction and user logging. In addition we hope that we can identify functionality gaps in the current SDKs.

The upcoming survey will be significant because 360-degree videos are gaining popularity among consumers, the developers are early adopters of technology and there are relatively few scientific publications about choosing 360-degree video SDKs.

This work is an initial study for a research project called 360 Video Intelligence. The purpose of the project is to create a 360-degree video platform which provides an easy way to run different kinds of analysis, for example object detection algorithms, on 360-degree videos. The videos with added metadata will be then played on 360-degree video player application. However, the player will not only play the video with visualized metadata but it will also gather user log for further analysis. Practical use cases for user logging include view port prediction that can be used for example on providing better video resolution only to the field of view similarly to work presented in (Ochi et al., 2014). We will also need some kind of UI elements for visualizing the added metadata on 360-degree videos. With the knowledge gained from developing our criteria and the following survey, we can have a better understanding about developing such applications.

ACKNOWLEDGEMENTS

This study was made in a research project called 360 Video Intelligence. We would like to thank TEKES for funding the project.

REFERENCES

Alface, P. R., Macq, J.-F., and Verzijp, N. (2011). Evaluation of bandwidth performance for interactive spherical video. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Argyriou, L., Economou, D., Bouki, V., and Doumanis, I. (2016). Engaging immersive video consumers: Challenges regarding 360-degree gamified video applications. In *Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), International Conference on*, pages 145–152. IEEE.

Bierbaum, A. and Just, C. (1998). Software tools for virtual reality application development. *Course Notes for SIGGRAPH*, 98.

Dalmasso, I., Datta, S. K., Bonnet, C., and Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 323–328. IEEE.

Fagerholm, F. and Münch, J. (2012). Developer experience: Concept and definition. In *Software and System Process (ICSSP), 2012 International Conference on*, pages 73–77. IEEE.

LaValle, S. M., Yershova, A., Katsev, M., and Antonov, M. (2014). Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 187–194. IEEE.

Linowes, J. and Schoen, M. (2016). *Cardboard VR Projects for Android*. Packt Publishing Ltd.

Nykaza, J., Messinger, R., Boehme, F., Norman, C. L., Mace, M., and Gordon, M. (2002). What programmers really want: results of a needs assessment for sdk documentation. In *Proceedings of the 20th annual international conference on Computer documentation*, pages 133–141. ACM.

Ochi, D., Kunita, Y., Fujii, K., Kojima, A., Iwaki, S., and Hirose, J. (2014). Hmd viewing spherical video streaming system. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 763–764. ACM.

Palme, E., Tan, C.-H., Sutanto, J., and Phang, C. W. (2010). Choosing the smart phone operating system for developing mobile applications. In *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, pages 146–152. ACM.

Shibata, T. (2002). Head mounted display. *Displays*, 23(1):57–64.

Yucel, I. H. and Edgell, R. A. (2015). Conceptualizing factors of adoption for head mounted displays: Toward an integrated multi-perspective framework. *Journal For Virtual Worlds Research*, 8(2).