Product vision and Planning

The First Order

Chris Berg 4216776
Martin Koster 4371011
Hung Nguyen 4232410
Christian aan de Wiel 4396286
Ruben Wiersma 4214250

28 apr 2016

Contents

1 Introduction

The Multimedia context project 2016 is an assignment for the company PolyCast. This company records classical concerts of an orchestra. Many aspects are involved in the recording and every aspect with its problems. First there is an aspect of what to film at a given moment. For this aspect a script is written in advance. A problem for this is that you cannot use the same camera for two consecutive cues. Another aspect is to control the cameras during the performance. An example of a problem is that some estimated view from a camera is different than expected. Finally you also have the editing after the performance to fine tune the audio and video. This are only a few aspects and problems there are many more As you see if this has to be done manually there are also many possible mistakes which can be made.

At this moment the company has a program to use the cameras. In this camera you can set preset of positions for different cameras. During a performance you can just click on a particular preset which is defined in the script. But this is only one problem which is solved by this application, so there are still lots of problems to solve.

The following section will explain what our goal is of our project, which problems we would like to solve. Thereafter in section three is explained how we want to accomplish this during our sprints. To do this optimally we have to set priorities which are stated in section four. Finally to check whether we have done a feature we must define what we define by done, this is done in section five.

2 The goal of our project

2.1 The target market

There are different people who will use the application. There are people who are experts in the audio and visual aspects. So these people have experience with computers and with editing applications. Other people with possible less experience with computers are the directors. The directors have much knowledge in the piece which is going to be played. The directors will choose when and to what the camera position has to be changed. Both these people have to be taken into account when designing the application.

Within PolyCast there are a few people which may use the application. First the script has to be made which is done in corporation with the director of the orchestra. During the performance the camera positions has to be changed according to the script. There is also someone who reads the music to indicate when there is a cue coming. For the last person it is necessary he can read the music without problems.

2.2 Our vision on the project

We are going to develop an application which offers support by the process of scheduling and scripting a recording of a classical concert. Creating the schedule is facilitated by a visualization of the concert hall with the positions of the cameras in it. The application will present the created schedule in different formats, fulfilling the needs of the people involved in the process of recording.

2.3 Our unique selling point

We are going to stand out from other planning tools by providing a visualization of the concert hall with the positions of the cameras in it. This visualization can, among others, be used to check whether a camera is available for filming during a certain queue.

3 Road map

In the road map, we discuss the planning done throughout the project. The road map consists of iterations called sprints, in which we complete certain tasks.

Sprint 1 We lay the foundation of the project. We decide which elements are important in the application and how they are going to look like.
Tasks: GUI, dataframework, vision.

Sprint 2 We focus on the basic features of the application: building scripts and the visualization of the concert hall.
Tasks: Script creation, interactive map of the hall, interactive score

Sprint 3 We focus on how new maps of the hall are constructed and how camera positions can be changed in a quick and easy way.
Tasks: UI tweaks (e.g. dragging camera's and Map editing), Map making

Sprint 4 We take care of saving and loading projects, and exporting the created schedule.
Tasks: Data export, Saving and loading projects, Saving and loading maps

Sprint 5 We focus on the part of the program which is used in live mode. We create the different formats the schedule can be presented in to the people involved in the process of recording.
Tasks: Creating different views used when in live mode

Sprint 6 We add extra features of the application, in order to enlarge the abilities of the application.
Tasks: Being able to export an XML file which links footage to queues for a video editor project file

4 Priorities

We use the MoSCoW-method (Dai and Richard, 2004) in order to prioritize the features we want to embed in the application. The MoSCoW-method consists of four principles, which are understood as follows:

Must Requirements which have to be contained in the project, else we have to consider the project a failure.

Should Requirements which are important, but do not have to be in the project.

Could Requirements which are not important, but may be included in the project.

Would Requirements which should not be included in the project at this time.

4.1 Must have

- The application shows a view of the score
- The application shows a view of the map
- The application shows a summary of the created camera action
- A user must be able to add a cue to a timeline view of a score
- A cue is added to a point given by its x coordinate in the score
- The x coordinate determines the order of the cues
- There is a map of the concert hall that displays camera positions and instrument positions
- A user is able to enter camera positions into the system
- A user is able to enter instrument positions into the system
- A user can set up cues by selecting a camera, and an instrument in the map view
- A user can set the zoom of the camera
- A user can set the action of the camera
- A user can set the duration of a camera action
- A user can see the availability of a camera
- Cameras cannot be selected when they are not available

- Camera actions are saved in a script to a database
- The database is separate from the application
- The application can export a script to PDF
- The application can export a script to XML
- The application runs in a web environment
- The interface works asynchronously

4.2 Should have

- A preview of the action is shown in the map (viewlines from camera)
- A cue is stored with the bar that it's placed in
- A user can appoint bars in the score
- Camera and user positions can be saved
- Camera and user positions can be restored from a file
- When a instrument position is added to the map, the name of the instrument should be auto-completed
- The application has a view for live performances
- In the live view, the score reader can progress the cue by clicking a next cue button
- The operator view shows the operator which cue they have next

4.3 Could have

- The user can see a preview of the camera on the instrument
- The operator view is different for each operator based on the cameras they operate
- The director can add constraints to the use of the camera

4.4 Would have

- Bars are automatically detected in the score
- Notes are automatically detected in the score
- In a live performance the script is automatically progressed
- Camera's are automatically adjusted to what's in the script
- Scripts are automatically generated

5 Definition of done

To end this report this section explains the definition of done.

Every feature which is implemented has to been tested for at least 75% with unit test. Of course the preferable amount of tests is 100% but this isn't always possible. Also the code has to be documented fully and has to be coded by the standard of Checkstyle, FindBugs and PMD. Finally Travis CI build must not have any errors.

For every sprint is has also been tested and coded according to the rules of a feature. Additionally to the features there has to be user tests to check if the system is working and according to the view of the user.

The end product also have the same rules the sprints. Here there are some extra tests. We will ask the users to use the program to get their view of the application. This will be done before the deadline so we can change the things they would prefer to see different. The end product is done when all the must have requirements are implemented. These requirements are necessary to get a working end version. For the should haves we would like to implement at least 50% of the requirements but this is a vague border.

6 References

- Clegg, Dai; Barker, Richard (2004-11-09). Case Method Fast-Track: A RAD Approach. Addison-Wesley. ISBN 978-0-201-62432-8.

A User stories