Product Planning, TI2806

Blazin and the Goons

Authors:

Shane Koppers,	4359062
Floris List,	4380258
Hidde Lycklama,	4397614
Joris Oudejans,	4393694
Jordy Weijgertse,	4247124

Contents

1 Introduction

We as a group chose to do this project. In the start we did not know exactly what the project would consist of and what we were expected to build. After visiting the PolyCast office the goal became a lot more clear. The members of the PolyCast team are eager to have a more improved and automated workflow. There our help comes in of course. We, as a team, decided we wanted to do just that. Our product will be an application that automates a couple of tasks that had to be done manually before. The goal of this application is to create a platform that speeds up the overall workflow. We want to achieve this by enabling the team members of the PolyCast team to create a script using a tool we designed. During recording sessions the team members are also going to be able to edit the script on the go and that everyone has a up-to-date view of the script. Next to that we want to improve camera controls and camera presets. PolyCast team members will get a lot more functionality from the camera usage.

2 Product

In this chapter we will specify the High-level product backlog(2.1) and the Roadmap(2.2), which specifies which features will be implemented in which sprints.
We divide the features in four different groups according to the MoSCoW-method:
Must haves features that are essential for the application to function
Should haves features that have a high priority and should be done, but could be satisfied in a different way if needed.
Could haves low priority features that could be implemented if time permits
Won't haves features that will not be implemented, but could be features for a follow-up project

2.1 High-level product backlog

Must haves:

- User-interface that can be used by all members of the PolyCast team
- Tool to create, manage and remove scripts
- Script database with structures for script actions, cameras and presets
- Real-time active camera view
- Interface optimized for iPad for recording mode that is synchronized with the other operators and director.
- Recording director controls to control the script flow
- Ability to control cameras over IP based on script actions
- Taking over recording button

Should haves:

- Camera position presets
- Camera preset creation tool
- Schedule system to match script actions together with camera presets in order to determine a working order and find conflicts in a script.

Could haves:

- The video sequence that is being live recorded can be exported to a file format that defines only the sequence of video (and not the video itself) that can be imported by PolyCasts' video editing software.

Won't haves:

- Elaborate learning algorithm to grade certain camera positions according to given heuristics.
- Algorithm to synchronize camera settings like warmth or brightness.

2.2 Roadmap

Design phase
In this phase we will decide our approach of the implementation of the product. Our goal is to:

-

Sprint 1
The first week of the implementation.

- Set-up all tools and environments for the development.

Sprint 2
The main goal of this sprint is to start implementing the must-haves.

-

Sprint 3

-

Sprint 4

-

Sprint 5

-

Sprint 6

-

Sprint 7

-

Sprint 8
Final sprint

- Refactoring the code if needed
- Check if the product is done according to the Definition of done defined in chapter 4

Release

- Make the final report
- Document the product well

3 Product backlog

3.1 User stories of features

A user should be able to make changes to a script during the recording of the concert. The user does this by clicking the desired part of the script that must be changed and a window will become visible where the changes can be made.

A user wants to switch to the next scheduled camera. The user does this by clicking a button on the screen that clearly says that it will switch to the new camera. As a result of that click the script queue will advance and the camera will be switched.

3.2 User stories of defects

A user wants to switch to the next camera, but when the user takes the right steps to do so the internet connection dies and the switch of camera won't happen.

3.3 User stories of know-how acquisition

3.4 Initial release plan

4 Definition of Done

This chapter describes what needs to be done on the different subjects in order to consider it as done.

4.1 Features

A feature is considered to be done when the following conditions are met. It should be explained when some condition isn't fully met.

- The feature is fully developed along with all it's sub-functionality in order to realize the feature.
- The code connected to the feature must have the following properties.
    - Enough tests have been developed to test this feature (both unit and integration tests) such that the code coverage is 75%. An exception can be made for user-interface related functionality.
    - Checkstyle does not give any warnings for the code.
    - The code must not have any software architecture defects or bad practices, discovered by PMD.

This ensures that the feature is ready to be merged with the main project. The feature works, is well documented and is (at least) reviewed by two other members of the team.

4.2 Sprints

We consider a sprint done when all the tasks of the sprint backlog have been fulfilled. Every sprint should release a product which is an improvement of the product of last week. The release of the improved product should have a version number and a tag before it can be considered done.

4.3 Releases

A release is considered done when all the sprints are considered done. It is important that all the stakeholders are satisfied with the product.
All the must haves should be implemented and some of the should haves as well. The code should also be implemented efficiently, well documented and approved by the SIG test.

5 Glossary

UI - User Interface. This is the interface users see and use to operate the system.
Bug - A flaw in the code that causes errors.
PolyCast - Company that is the end-user of the application
Sprint - A development cycle in which tasks in the backlog are performed