Product Planning

Team Free Pizza

1. Introduction

Planning is, of course, a very important part of every project, for as it is said, failing to plan is planning to fail. This report will cover our ideas of the planning during this project. The planning is not final and might change during the project, depending on the situations we encounter. This is due to the fact that the process of developing software is different for every project and thus difficult to predict.

2. Product

2.1 High-level product backlog

Must-haves

The features in this list are required for correct functionality of the application.

- Create presets for different cameras.
- Create a digital script with these presets using a graphical interface.
- Saving and loading of digital scripts.
- It should be possible to go to the next shot with just one click of a button.

Should-haves

The features in this list are not required for correct functionality of the application, but are highly preferable.

- Automatic loading of presets for the cameras.
- It should be possible to make a preset dynamic, making it possible to program camera movements like zooms and pans.
- When the camera is actual it should not be allowed to switch presets.
- When a camera is switching to a preset it should not be allowed to become actual.
- It should be possible to configure the track of a camera beforehand.
- During live production it should be possible to override the script and control everything manually.

Could-haves

The features in this list will only be implemented when there is enough time to do so.

- Dynamic presets could be set to use not just a linear curve, but also a quadratic, logarithmic or custom curve for the movements.
- A director can view a preview of the whole concert when all cameras are configured.

Would-haves

The features in this list will most likely not be implemented in this timeframe, but are interesting to implement at a later point in time.

- Cameramen get live notifications about abrupt changes in the script during live production from both directors and other fellow cameramen.
- If an actual camera gets stuck or does not work for some reason, then a camera not in use can automatically take over its task.
- Communication between cameras, making the work of the controllers less tedious.

2.2 Roadmap

The roadmap is a general planning of the product development. It is more of an outline than a detailed planning. If it turns out that the planning in this roadmap is not viable we can still make changes in order to get back on track. Features that will be implemented in the first few sprints will have priority.

2.2.1 Sprint 1

This sprint will be more of a design and setup phase. We will write a product vision and product planning and make sure to configure all the tools we plan on using. We will also start to gather knowledge about the different API's we plan on using.

- Write product planning.
- Write product vision.
- Draw draft of the GUI.
- Create architecture design.
- Configure all the tools.
- Learn about the API's we plan on using.

2.2.2 Sprint 2

During this sprint we will start with actual programming and implementing must-have features.

- Create a basic GUI.
- Implement back-end creation of scripts.

2.2.3 Sprint 3

The must-haves are probably not all implemented yet so we will take more time to implement them.

- Continue working on the GUI.
- Implement saving and loading of scripts.
- Add the back-end of creating presets

2.2.4 Sprint 4

If everything is going according to plan the backend and GUI should be integrated this sprint.

- Integration of GUI with back end.
- Implement going to the next shot during live production.

2.2.5 Sprint 5

All must-haves should be implemented and we can start working on the should-haves

- Automatic loading of presets for the cameras.
- Create interface to configure the dynamic presets.

2.2.6 Sprint 6

This sprint there will be more time to implement should-haves

- Continue working on the dynamic presets.
- Implement restraints on switching presets on a camera that is actual.
- Implement restraints on making a camera actual that is switching presets.

2.2.7 Sprint 7

This sprint the last should-haves will be implemented

- Implement manual control during live production
- Optimise GUI with feedback from PolyCast

2.2.8 Sprint 8

All must-haves and should-haves should be implemented right now and we can use the remaining time to fix bugs and focus on maintainability. If there is enough time we can start working on could-haves.

- Solve remaining bugs.
- Improve maintainability.
- Improve code quality.
- Write additional tests where necessary.

2.2.9 Sprint 9

The last sprint will be used to document our product and prepare for the final presentation.

- Document difficult code.
- Write final report.
- Prepare presentation.

3. Product Backlog

3.1 User stories of features

As a cameraman I want to spend less time on configuring cameras and more time on creating an artistic shot. That is why it should be possible to create presets for cameras.

As a director I want an easy way to create a digital script.

As a director I want to load my previously created scripts and save my current scripts for later use.

As a director I do not want to spend time finding the right button for the right camera. I just want to press one single button to go to the next camera.

As a cameraman I do not want to spend all my time loading presets, so the presets should be loaded automatically.

As a director I do not want to show a camera that is switching presets, so this should be restrained.

As a director I want to know the track of my cameras beforehand, so this should be configurable.

As a director I want to take over control of the program when something unexpected happens.

As a cameraman I want to know when the script changes during live recording, so I should get a notification when this happens.

As a director I want the system to automatically cover for a defective camera.

3.2 User stories of know-how acquisition

As a programmer I want to receive feedback on the interface I created.

As a programmer I want receive information about the cameras we have to control.

As a programmer I want to receive the API on how to control the cameras.

As a client I want to know how the development of the product is coming along.

3.3 Initial release plan

We will release a new version of our product every week. Therefore, the release plan will be in line with the roadmap.

Week	Milestone
1	All documents have been created
2	A basic GUI has been made and scripts can be made in the back-end
3	Saving and loading of scripts is now possible.
4	GUI and backend will be linked and there is now a first working version. Also, a global “next shot” button for live production will be made.
5	Presets can be automatically loaded according to the script and an interface for script creation is made.
6	The system will now warn a user if (s)he wants to switch to/from an actual camera to a moving camera.
7	The director can now manually take over control during live production.
8	Remaining bugs are looked into and additional tests are made
9	Final version release

4. Definition of Done

In this section we will define when we consider a feature, a sprint or the final product as done. The definition of done is a checklist of qualities that something must have before we can mark it as done.

A feature is done if all aspects have been coded and the code has been approved by another team member using a GitHub pull request. Also, all unit tests for the particular feature and all pre-existing tests should pass. By also requiring the pre-existing tests to pass we try to make sure the newly created feature does not break existing functionality, this is called regression testing. A feature with bugs can be marked done, but only when the bugs do not interfere with the user experience.

A sprint is done when all features that should be implemented in the sprint are marked done as described above. Possible bugs introduced by integrating the features should not have an impact on the user experience.

The final product is done when all sprints are done, all must-haves and a large part of the should-haves are implemented and all code is thoroughly tested. This means unit tests, integration tests and user tests. All code should be documented accordingly and should pass the test by the SIG.

When a feature is labeled as done, this means we will stop working on them. However, when the customer has feedback on a feature that it should be changed, that feature will no longer be considered done.

5. Glossary

GUI

Graphical User Interface

SIG

Software Improvement Group