Why UpStage Server-end Should Enforce Media Synchronization

Yue Li

May 23, 2014

1 Introduction

UpStage is under a transition from the current legacy implementation towards a total new design with up-to-date open source technologies. On our client's request, team looked into a series of the high priority issue tickets. With the further investigation, team identified the issues that are currently excluded from team's current project scope. The aim of this paper is to highlight the current high priority issues of UpStage Version 3, and further clarify the need of identifying Server-end technology as research area to secure synchronization among UpStage users.

This paper will firstly describe the high priority issues in UpStage Version 3 implementation, then it will produce an initial analysis of the issues. This paper then will further clarify the area as a synchronization issue on the server end. Then this paper will emphasize the impact of such feature in the UpStage implementation and the importance of synchronization feature to the client of UpStage project.

2 Issues

All the issues in this section are from AUT UpStage Team GitHub repository. Please refer to the repository issue list for details.

2.1 Ticket 203, 228, 230, 253, 254, 259, 264

203 holding different avatars on stage

228 voice swap between avatars

230 different users holding the same avatar

253 different backdrops show for different users

254 blank screen after reloading

259 clear button clears all avatars from stage

264 not seeing other avatars

3 Analysis

This section summarize the issues in the previous section. All the issues in section *2.1* share the same attributes as below:

- Multiple users involved.
- Media display vary for different users.
- Users upload media onto stage in different times.
- Users enter stage at different times.
- Situation is fixable by reloading stage swf (small web format) template from UpStage server.

Here is a set of detailed steps to reinstate issues.

Scenario 1 Both Player A (Pa) and Player B (Pb) have the media upload access to a stage (testS). Pa firstly uploads media via (either stage or media upload), then enters testS. Then Pb uploads media, however, this time Pb does not enforce stage to reload (from the stage edit page, the SAVE button was designed for the purpose). Once avatar is uploaded and assigned to the testS, Pb enters the stage.
After the above events. There are several consequences:

- Pa will not see Pb's avatar on the wardrobe.
- Pb will not see any avatar showing on Pa's stage.
- Pb is able to select any avatar in the wardrobe, including the one's Pa is holding.

The same situation can also apply to all other media (backdrops & probs). One noticing item: the "Save Stage" resets media arrangement when clicked. So there is one scenario to show different display problem on multiple users' locations.

Scenario 2 There are PlayerA (pA), PlayerB (pB), PlayerC (pC), and stageT. pA and Pb have entered stageT with loaded media. pA and pB both have avatars showing on the stage. Then pC wants to add more media into the stage media and choose "Save Stage" button. Once pC completes stage save, pA & pB will no longer hold their avatars (there is no notification to pA and pB. They may think they are holding the avatars; both pA and pB do not hold any avatar). Then pC enters stageT. The display results for pA, pB & pC shall be different.

In the above analysis, the both scenarios showcased synchronization issues in the v3 implementation. This situation can be possibly resolved by removing "Save Stage" button from stage edit page. However, the removal of "Save Stage" button causes v3 product fails to meet client's needs for rehearsals – from being overwhelmed by frequent stage reloads.

At this point, the status of stage synchronization is:
All stages can synchronize the numbers of players, audiences and chat information. All stages suffer from synchronizing uploaded media dynamically.

A preferred solution is to have dynamic media upload feature, so that users can avoid frequent stage reloads, and improve performance efficiency. Team has no information

on how long it may take to implement such feature; therefore, synchronization problem should be treated as an independent research topic for future AUT UpStage Teams.

4 Conclusion

Ultimately, AUT UpStage Teams have tried to create robust UpStage software to support UpStage community. Both the performance of current software and client's requirements indicate a need for researching in software synchronization. By fulfilling UpStage will be more usable to serve its target users and audience.