Product Name: Reposte

Customer Name: Terry Yoo

Team Name: Goobernauts

Team Member Names:

Stephen Goodridge, Logan Geiser, David Geng, Ryan Giles, Heath Miller

Date: 10/5/2024

Version: 1.0

System Requirements Specification

Table of Contents

1. Introduction

This capstone project is for the development of a video referee replay system software, in partial fulfillment of the Computer Science BS degree for the University of Maine. The goal of this project is to program an open-source video replay software for fencing referees, and should also work with operating systems such as MacOSX, Windows, and the possibility of Linux. The project will provide replay capabilities to fencing clubs, and improve the competitive experience.

1.1 Purpose of This Document

The purpose of this document is to outline the system requirements for a video referee replay system. It is intended to be read by the development team and project sponsors. This document gives an overview of the functional requirements, non-functional requirements, user interface guidelines, and testing criteria.

1.2. References

Yoo, Terry (2024). "Video Referee Replay System Software for Sport Fencing Competition" Project Proposal, University of Maine

1.3. Purpose of the Product

The USA Fencing National Referee requires a video replay system to help referees review close calls during fencing matches. Currently, the system is outdated and needs to be replaced with updated, open-source software. This product aims to give a solution to clubs and competitions

1.4. Product Scope

The video referee replay system will allow fencing referees to review footage of important moments during matches, such as hits and fouls, at different speeds and resolutions. It will be compatible with both MacOS and Windows (Possibly Linux) and be capable of capturing videos at 60 frames per second. The program will also store video for future review and support interface integration with scoring machines.

2. Functional Requirements

1. System shall record real time video at 60 frames/second so that review can be accurate. (**USE CASE #1**)
    a. Acceptance Criteria: Program can capture high-definition video at a minimum of 60fps from camera input and display in a synchronized manner.
2. System shall be able to replay specific moments immediately after they occur so the referee can make accurate judgements. (**USE CASE #1**)
    a. Acceptance Criteria: Program provides user interface that allows the referee to quickly replay recent moments, slow down playback, or pause the video.
3. System shall record audio along with the video so the audio cues will be captured as well. (****USE CASE**#2**)
    a. Acceptance Criteria: Program can capture audio from a connected microphone and synchronize it with the video recording in real time.
4. System shall work on both MacOS and Windows, and eventually Linux, so it can be used regardless of operating system. (**USE CASE #4**)
    a. Acceptance Criteria: The program can be installed and run on MacOs and Windows platforms with identical functionality across all platforms.
5. System shall save the recorded video and audio with event markers, so that it can be reviewed and shared after a match. (**USE CASE #3**)
    a. Acceptance Criteria: The program can export the recorded video and audio files into standard video format.
6. System shall interface with an Arduino device to capture signals from fencing equipment and log those events during video capture. (**USE CASE #5**)
    a. Acceptance Criteria: The program can read signals sent from an Arduino device via USB and initiate the replay functionality.
7. System shall interface with USB, USB-C, and HDMI inputs. (**USE CASE #6**)
    a. Acceptance Criteria: Program can receive signals and interface with USB, USB-C, or HDMI inputs.
8. System shall change the replay speed so that the referee can better understand what actions occurred. (**USE CASE #7**)
    a. Acceptance Criteria: Program can slow down the replay at 10% intervals from 100% speed to 10% speed.
9. System shall save recorded videos with a naming convention provided so the referee/official can find saved replays easier. (**USE CASE #8**)
    a. Acceptance Criteria: Program can save recorded video with a desired naming convention.
10. System shall have the ability to function in fullscreen mode so the referee has the best view of the replay. (**USE CASE #9**)
    a. Acceptance Criteria: Program functions in a fullscreen mode without frame drops.

2.1. Use Cases

Number	1	
Name	Capture and replay video	
Summary	Program shall capture video and be able to replay the video.	
Priority	5	
Preconditions	Must have a trigger for when a hit/foul is called.	
Postconditions	Video should immediately be able to be watched for review by the referee.	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Arduino USB signal, Click on spacebar by video ref	
Main Scenario	Step	Action
	1	Referee calls hit or foul
	2	Video referee hits trigger
	3	Program captures video of what was happening before the hit or foul
	4	Referee watches playback
	5	Allow options for multiple speeds for video

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● Do we want to use a manual trigger?	

Number	2	
Name	Capture Audio	
Summary	Program shall capture audio alongside with video	
Priority	5	
Preconditions	Must have laptop microphone	
Postconditions	Audio should be recorded and sync with video	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Video recording in progress	
Main Scenario	Step	Action
	1	Referee starts recording
	2	Microphone records audio
	3	System matches audio to video timestamps
	4	Referee watches playback with audio

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● Quality of microphone ● Managing memory usage on 8GB laptop ● Audio during slower replay speeds	

Number	3	
Name	Save recorded video and audio to be shared/reviewed	
Summary	Program shall save recordings of the match to standard video format.	
Priority	4	
Preconditions	Video should be recorded and saved in the system	
Postconditions	Referee can share that video	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Referee uses the ‘Play/Pause” button (Spacebar)	
Main Scenario	Step	Action
	1	Referee presses “Play/Pause” button to start recording
	2	Video of match is recorded
	3	Arduino signal is detected
	4	Video recording is halted
	5	Referee is presented 2-4 second playback
	6	After Review referee presses “Play/Pause” button to start recording

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● How do we buffer the video for replay? ○ Circular/Ring buffer? ● Managing memory on 8GB laptop	

Number	4	
Name	Operating System compatibility	
Summary	Program shall be able to be installed on both MacOS and Windows using the same source code.	
Priority	5	
Preconditions	Must have laptop with one of the two operating systems	
Postconditions	Program shall install and function the same regardless of OS.	
Primary Actor	Official Referee	
Secondary Actors	Video Referee	
Trigger	Installation	
Main Scenario	Step	Action
	1	Official chooses to use Reposte
	2	Official installs the program on their desired OS
	3	Program installation is successful
	4	Program runs the same on both OS

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● How do we make installation easy and accessible? ○ Use GitHub?	

Number	5	
Name	Interface with Arduino	
Summary	Program shall interface with an Arduino to receive a signal to initiate replay	
Priority	5	
Preconditions	Must have a trigger for when a hit/foul is called.	
Postconditions	Signal from the Arduino should be taken by the program and initiate replay.	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Arduino USB signal.	
Main Scenario	Step	Action
	1	Fencing equipment sends signal to Arduino
	2	Arduino receives signal and transmits it via USB
	3	Program interprets signal then initiates replay functionality

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	• How to test • Knowledge acquisition on microcontroller programming	

Number	6	
Name	Interface with USB, USB-C, and HDMI	
Summary	Program can receive signals and interface with USB, USB-C, and HDMI.	
Priority	5	
Preconditions	USB, USB-C, or HDMI device is plugged into laptop	
Postconditions	Program receives signals through those respective ports.	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	USB, USB-C, or HDMI input	
Main Scenario	Step	Action
	1	Referee plugs in necessary devices
	2	System receives signal from device
	3	System interfaces with device

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	- How to test - Memory management - What products to support? - USB/USB-C: Microphone, Additional Camera? - HDMI: Monitor support? Max resolution?	

Number	7	
Name	Adjustable replay speed	
Summary	Program can slow down replay speed at 10% intervals(10%-100% speeds)	
Priority	5	
Preconditions	Must have a trigger for when a hit/foul is called.	
Postconditions	Replay video will have adjustable speeds.	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Arduino USB signal, Click on spacebar by video ref	
Main Scenario	Step	Action
	1	Replay function is triggered
	2	Replay footage is presented
	3	Replay footage offers adjustable playback speeds
	4	Referee watches playback at desired speeds

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● Image quality at reduced speeds ● Audio during slower replay speeds	

Number	8	
Name	Custom naming conventions	
Summary	Program can save recorded video with a desired naming convention	
Priority	3	
Preconditions	User runs software.	
Postconditions	Video should save in desired naming conventions.	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger		
Main Scenario	Step	Action
	1	User runs program
	2	User is prompted to enter save file name
	3	User enters desired save file name
	4	Program saves recording file with given name

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● Memory management ● UI design	

Number	9	
Name	Full Screen mode	
Summary	Program can run in full screen	
Priority	4	
Preconditions	Program and dependencies installed.	
Postconditions	Program runs in full screen mode	
Primary Actor	Video Referee	
Secondary Actors	Official Referee	
Trigger	Program ran	
Main Scenario	Step	Action
	1	Official opens program
	2	Program runs in full screen mode
	3	Referee can view replay system in full screen

Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	● How to test ● Resolutions constraints ● Memory management	

2.2. Testing

Through a series of test cases, the following will ensure that the system meets performance and functionality requirements, as well as evaluate the system's ability to capture, save, and replay videos for the referee, while integrating with external devices.

1. **Capture and Replay Video**
   Test Case 1.1: Verify system captures video at 60fps
      Steps:
         1: Start recording
         2: Verify the video playing is 60 fps
         3: Trigger replay system
         4: Ensure video replay never falls below 60fps
      Result: Video replay stays above 60fps
   Test Case 1.2: Verify user can replay video immediately after capturing
      Steps:
         1: Start recording
         2:Trigger replay system
         2: Verify video replay is instant
      Result: Video replay is immediate once triggered

2. **Capture Audio**
   Test Case 2.1: Record audio with video
      Steps:
         1: Start recording
         2: Trigger replay system
         2: Audio is playing with video
      Result: Audio is playing with video
   Test Case 2.2: Sync audio with video
      Steps:
         1: Start recording
         2.Trigger replay system
         3: Audio is playing with video
         4: Verify audio is in sync with video
      Result: Audio is synced with video and aligns with video cues

3. **Save Recorded Video and Audio to be Shared/Reviewed**
   Test Case 3.1: Program should save recordings of the match to standard video format
      Steps:
         1: Start recording
         2: Trigger replay system

3: Verify audio and video works and is in sync
4: Click the save button
5: Verify file is saved to a directory in standard video format
6: Ensure video and audio plays and are sync

Result: Video replays are saved to a directory and can be played for review or shared.

Test Case 3.2: Program should display event markers of saved video locations

Steps:

1:Start recording
2: Trigger replay system
3: Verify video and audio play
4: Click the save button
5: Check file directory for file containing event markers of when replay occurred

Result: Files display event markers for when replay happened. This promotes easier search for specific replays

4. **Operating System Compatibility**

Test Case 4.1: Verify program can be installed on MacOS

Steps:

1: Acquire installation for MacOS
2: Install program
3: Open application and select webcam and microphone
4: Play/Stop video
5: Trigger replay system
6: Verify file is saved to a directory in standard video format
7: Ensure video and audio play and are in sync

Result: Program installs and works as intended on MacOS

Test Case 4.2: Verify program can be installed on Windows

Steps:

1: Acquire installation for Windows
2: Install program
3: Open application and select webcam and microphone
4: Play/Stop video
5: Trigger replay system
6: Verify file is saved to a directory in standard video format
7: ensure video and audio play and are in sync

Result: Program installs and works as intended on Windows

(Optional) Test Case 4.3: Verify program can be installed on Linux

Steps:

1: Acquire installation for Linux
2: Install program
3: Open application and select webcam and microphone
4: Play/Stop video
5: Trigger replay system
6: Verify file is saved to a directory in standard video format
7: ensure video and audio play and are in sync

Result: Program installs and works as intended on Linux

5. **Interface with Arduino**

Test Case 5.1: Verify program receives signal from Arduino to initiate replay

Steps:

1: Verify Arduino is hooked up to computer and fencing equipment
2: Test Arduino signals
3: Start program
4: Verify Arduino triggers replay system by referee stop, hit, or foul

Result: Arduino sends replay signal when referee needs review

6. **Interface with USB, USB-C, and HDMI**

Test Case 6.1: Program receives signals and interfaces with USB, USB-C, and HDMI

Steps:

1: Verify USB, USB-C, or HDMI are properly connected
2: Test Arduino signals
3: Start Program
4: Verify system receives signal from device
5: Verify system interfaces with device

Result: USB, USB-C, and HDMI are all viable sources to receive signal

7. **Adjustable Replay System**

Test Case 7.1: Program can slow down video that varies from 10% - 100%

Steps:

1: Start recording
2: Trigger replay system
3: Select correct recording file from the directory
4: Replay offers adjustable speeds with corresponding number keys for video speed. (i.e. 1 = 10%, 2 = 20%, etc.)

Result: Video files have varying slow motion speeds for accurate hit/foul detection

8. **Custom Naming Convention**

Test Case 8.1: Each replay saved has a custom naming convention for organization

Steps:

1: Start recording

2: Prompt displays to enter save file name

3: Type desired name

4: Program saves each recording with that naming convention

Result: Each saved file is named to correspond with the match for organization purposes and easier to find within the directory

Test Case 8.2: User enters naming convention already in use in directory

Steps:

1: Start recording

2: Prompt displays to enter save file name

3: Type desired name

4: Program prompts user to enter new name for saved file

Result: Program prompts user to enter a new naming convention as the one entered is already taken

9. Full Screen Mode

Test Case 9.1: Program can run full screen and windowed

Steps:

1: Start program

2: Select a window button to bring program in or out of full screen

3: Trigger replay system

4: Select video recording for replay in directory

5: Select a window button to bring replay in or out of full screen

Result: Program runs smoothly in full screen or windowed mode and seamlessly transitions between the two.

Test Case 9.2: Program can run at different resolutions

Steps:

1: Start program in computers native resolution

2: Start recording

3: Trigger replay system

4: Select video replay in the directory for playback

5: Ensure 60 fps is consistent at that resolution

6: Apply same test at a range of resolutions (800x600 - 4k)

Result: Video recording and playback stay at or above 60 fps at any resolution

3. Non-Functional Requirements

1. The system must capture and process video and a minimum of 60 frames per second(FPS) and audio in real-time, without dropping frames or introducing noticeable latency.
    a. Priority: 5
    b. Metric: Video recording maintains at least 60FPS on systems with 8GB RAM.
2. The software must run on MacOS, Windows, and preferably Linux, using a single codebase with cross-platform support.
    a. Priority: 5
    b. Metric: System builds and runs successfully on all two/three platforms, with no major differences in functionality.
3. The system must use CPU, memory, and disk resources efficiently, ensuring it does not cause excessive strain on the user's machine, particularly during extended periods of use.
    a. Priority: 4
    b. Metric: CPU usage must remain below 50%, and memory usage below 4GB on a system with 8GB RAM during operation.
4. Software must process video, audio, and input signals from the Arduino device with minimal delay to ensure accurate real-time analysis
    a. Priority: 5
    b. The system must process each frame and audio input within 16.67 milliseconds(for 60 FPS), ensuring latency does not exceed 100 milliseconds.
5. System must provide backup of video files to ensure data isn't lost.
    a. Priority: 4
    b. System will automatically save replays to a secondary storage location once the recording ends.
6. System must put video files in a secure location to prevent unauthorized access.
    a. Priority: 5
    b. Files will be secured behind user authentication and logins that only authorized personnel should have access to.
7. Software must be created to ensure that it is easy to update and make modifications to, to ensure that new developers can update code.
    a. Priority: 3
    b. New developers should be able to update code easily, read the preexisting code, and make these changes without messing up previous functionalities.
8. Software must minimize the use of the systems CPU and GPU while idle.
    a. Priority: 3
    b. While software is idle, CPU and GPU should not go over 20-30% usage.
9. Software must have a simple, yet effective, UI to ensure the referees can use it with

minimal training.
    a. Priority: 4
    b. New users/referees should be able to stop/start recording, initiate a replay, change speed of playback, etc. within a few minutes of familiarizing themselves with software.
10. System must be able to handle high-definition video inputs without lowering performance.
    a. Priority: 5
    b. Software should be able to handle video up to 1080p resolution without a drop in frames.

4. User Interface Requirements

In Reposte, user interface design will be minimal as to save computer resources. As the project progresses, the UI design will change to accommodate certain specifications or quality of life updates. At that time, we will assess the needed accessibility features to optimize inclusion and usability.

At the most basic level the user interface will be handled through the keyboard. The functions that will be handled by the keyboard are play/pause, replay speed(10-90%), and ‘halt & replay’. Utilizing the spacebar for play/pause functions. The number keys 1-9 will control the speed of the replay. 1 being 10% speed and 9 being 90% speed. There will be a need for a ‘Halt and Replay’ button. This button, after being pressed, will stop video recording and provide the user with the replay functionality. Client has added a request for a slider for managing the replay speed and area to tag competitors in the replay footage.


Bell Bank
FENCING
FENCING
PLAY/PAUSE