# Final Report

**for**

# Interactive Video Art Project

**Version 1.0 approved**

**Prepared by Stephanie Lukin**

**CS 496 Computer Science Project I**

**13 December 2010**

Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this product is to create an environment for users to create Interactive Video Art. This project can be seen as an add-on or an extension to the Processing environment. The platform and language used are Processing (Java based), which is a free and open source environment specifically aimed at easily creating images, animations, and interactions. In addition to the built in functionalities of Processing are dozens of libraries which users have created to enhance various aspects including but not limited to image processing, video rendering, and full screen display.

Our project will make use of the built in Processing functionalities as well as some user defined libraries. It will also implement an Arduino sensor box to collect information from the physical environment to then interact with the software. This document will provide specifications of the general processing environment as well as the preexisting libraries used. Our extension has been developed to handle sensor interactions and integrate them with Processing video interactions. Rather than develop a full Graphical User Interface for the users, we have created our own library for the users.

This is the first documentation of the product, Version 1.0. It will cover the product features of the first iteration in detail as well as time, cost, schedule and risk estimations. It describes the user interfaces as well as the functional and non-functional requirements of the product. Furthermore, it will address the design issues and solutions, as well as the testing methodology, final results, and a reflection.

## 1.2 Document Conventions

When listing items, their order of priority will be indicated explicitly, although most cases will be from highest to least of important or priority.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for users and developers. The most pertinent sections for the users are Sections 3 and 4 which cover the features and interfaces respectively. This will give them an indication of the product capabilities and should be read before interacting with the product. It will also give an overview of the interface and how to use it. Developers can use this document as a foundation for addition future capabilities. They can see what has been developed and how long it took, providing an estimate of future development. Developers should begin with Section 3 to see the existing features and Section 2 with the estimations.

## 1.4 Product Scope

This product is being designed as a tool for artists to use to express themselves in a video environment. It will allow users to upload their own videos and the system will react and trigger

new events based on specified criteria. Users who are unfamiliar with a programming environment will be exposed to a new medium of expression. It can be seen as a palette, providing the users with many options that allow them freedom to combine and personalize their expressions.

There are many possible features to incorporate into this product. The first iteration will allow the more simple functionalities utilizing the built in video camera on the iMac and the Arduino sensors. After these have been established and approved by the student artists, The video art class with which we are working was broken into groups. Each group had a unique vision for how to implement our library and hardware with their videos. The Gallery at the end of this semester will present four unique video installations that will implement the same software and similar hardware. A description of the four projects are as follows:

### 1.4.1 Project 1: Color

This display will be on a computer screen and consist of six different videos. When the display begins, a still shot of each video will be shows around the edge of the screen. A live feed will be shown in the center. A small table of uniquely colored objects will sit beside the display. Each colored object will correspond to a different video, the theme of which is that color. If the viewer touches an object to the sensor box, the live feed will disappear and the video corresponding to that color will play. After it finishes, the screen will return to the live feed.

### 1.4.2 Project 2: Projector

This display will be projected onto a blank wall and a green screen will stand opposite the wall. Four of the recorded videos will be used as a background into which the viewer will be placed if he or she steps in front of the screen. The backgrounds will be changeable by the color sensor. The fifth video will be an interactive drawing board. An object, such as a ping pong paddle, can be used to erase the current video. Underneath will reveal another video. When reserving the ping pong paddle to expose the other colored side, the original video will be drawn back.

### 1.4.3 Project 3: Looping and pixilation

This display will be on a computer screen and show two videos, looping back into each other. When the viewer approaches the display and claps their hands, the current video will pause and begin to break up into pixels. The pixels will then grow larger and swirl around the screen. After the swirling, the pixels will reform into the other video and begin playing. Also, if the viewer moves within a certain distance from the display screen, the video will change to live feed.

### 1.4.4 Project 4: Music

The final display will be on a computer screen and have ten videos. The color sensor and uniquely colored objects will trigger the changing of videos. Each video will play a different song associated with the video; a pair of headphones will be provided. If there is no motion detected for a certain amount of time, the screen will stop playing the videos and trigger a neutral screen with instructions about how to change the video.

## 1.5 References

“Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.”
Processing Language: <http://processing.org/>

“A collection of step-by-step lessons covering beginner, intermediate, and advanced topics”
Processing Language tutorials: <http://processing.org/learning/>

“Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.”
Arduino Programming language: < http://www.arduino.cc/>

# 2. Project Estimate, Schedule, and Risks

## 2.1 Software Development Process Used

An iterative and incremental development process model will be used for this project. There are many different functionalities that can be incorporated in this project and the final vision is flexible. Therefore, the project will start with more basic and general functionality and increase according to what the client finds compelling.

Each iteration consists of an inception phase, which defines the requirements at a high level, schedule, risks, and scope; an elaboration phase which explores the project architecture and expands upon the functional and non-functional requirements; the construction phase begins the actual implementation of the project based on the collected requirements and designs; and finally, the transition phase produces a user friendly product to be released.

The first iteration will produce a prototype containing basic functionalities of the iMac with a built in video camera and an Arduino sensor interaction. After the initial release, the prototype will be enhanced by adding more complex features and a more user friendly interface if time allows.

## 2.2 Project Task Set (Statement of Work or SOW)

The deadline for the video art student’s final project exhibition begins the week of December 3rd, 2010. Therefore, a working prototype must exist well before then as well as allow the students enough time to play with the product and become familiar with the tools. They must also create their final video project in this time. Allowing the students about a week to create their project, the final product must be delivered before the students leave for Thanksgiving break on November 22nd, 2010. Before this time, working prototypes will be introduced to the class so that they can gain familiarity with the tool before the final product is delivered. After the final delivery date, any

bugs reported or enhancements requested will be fixed as quickly as they are detected, but this would best be avoided.

This leaves six weeks before the final product must be delivered in time for the students to complete their final art assignment. A working prototype should be delivered by the end of week three (November 5th), to get feedback from the students. After the initial release, feedback will be collected and the product enhanced accordingly, delivering another prototype at the end of week four (November 12th). The students will provide further feedback to be studied and enhanced for the final product. There will be a total of two prototypes and a final working product.

Before the first prototype is delivered, the design and implementation of the program must be decided. At this point in the development phase, many ideas have been gathered and they must be worked into a comprehensible product.

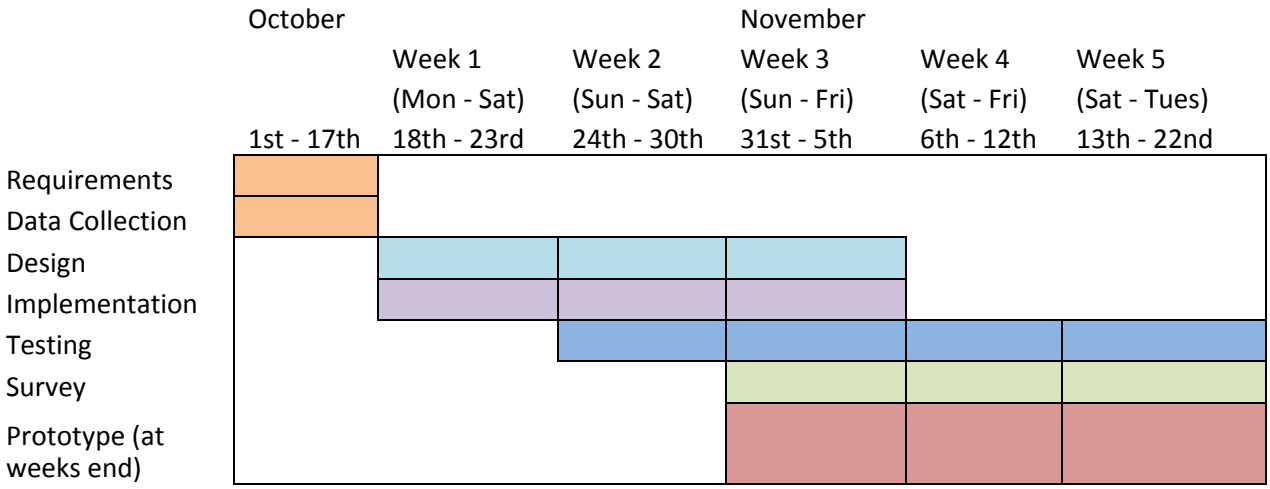
## 2.3 Estimation Techniques Applied and Results

The time estimations were created mainly in order to accommodate the two prototypes to be developed and tested by the students. There will be three iterations: the first is the planning and designing of the initial prototype from week 1 to the end of week 3 (November 5th); the second iteration will take place during week four, working on improving the product after the initial release and releasing another prototype at the weeks' end (November 12th); the third iteration will be week five through the weekend until the beginning of week six (November 22nd) which will release the final prototype.

## 2.4 Reconciled Estimates and Project Resources

In the first three weeks, the Dr. Daniel Schlapbach and the video art students will be interviewed in order to gain a better understanding of what they are expecting. After delivering each prototype, their input and feedback will again be reviewed and incorporated into the next product delivery. Dr. Roger Eastman will be consulted for advice about the software aspect of the product. The Arduino board expert and builder will be Jason McMahon who works as the Technology Support Specialist for Loyola College of Arts and Sciences. The developing of the first prototype will take place in the Computer Science department laboratory on the iMac computer.

## 2.5 Timeline Chart (e.g., a Gantt Chart)*




## 2.6 Project Risks, Risk Impact and Mitigation plan

There are many potential risks associated with this project. They will be listed in order of their probability of occurring, from highest to lowest.

- The art students will not be satisfied with the prototype, i.e. not enough flexibility, did not meet their expectations. Because art is so subjective, some students may not appreciate the final results of the project. In order to fulfill the needs and interests of as many students as possible, it will be necessary to interview some students and the class instructor to see what they have in mind. It will also be vital to present them with a working prototype at regular intervals to see if they are satisfied or what they would like to see changed. If this risk is realized, the students will at least be able to complete their final art assignment, but will not enjoy it.

- The art students will not have a functioning prototype for their final art assignment. This could occur if the time estimation was not properly calculated. At each iteration the project must be backed up and properly documented to guarantee that there will at least be a working version, even if it is an earlier one. Enough time must be allotted to the development of the project as well as incorporate the real potential for unforeseen obstacles.

- The art students will have a difficult time interacting with the prototype. This would be the result of poor planning and documentation. The way to prevent against this risk is by presenting the students with a working prototype and seeing how easily or difficultly they interact with it.

# 3. Overall Description of the Product

## 3.1 Product Perspective

This is a new, self-contained product. The idea was proposed by Professor Dan Schlapbach in the Photography Department of Loyola University Maryland. Inspired by the new and interesting field of interactive video art, Professor Schlapbach wanted to give his students the opportunity to explore this field. He requested the help of Dr. Roger Eastman, Computer Science Department at Loyola University Maryland, who passed on the potential project to the Computer Science Project class.

This project can be seen as an add-on or an extension to the Processing environment. The platform and language used are Processing (Java based), which is a free and open source environment specifically aimed at easily creating images, animations, and interactions. In addition to the built in functionalities of Processing are dozens of libraries which users have created to enhance various aspects including but not limited to image processing, video rendering, and full screen display.

Our project will make use of the built in Processing functionalities as well as some user defined libraries. It will also implement an Arduino sensor box to collect information from the physical environment to then interact with the software. This document will provide specifications of the general processing environment as well as the preexisting libraries used. Our extension has been developed to handle sensor interactions and integrate them with Processing video interactions. Rather than develop a full Graphical User Interface for the users, we have created our own library for the users.

## 3.2 Product Functions

The following interactions and features were implemented and presented to the Video Art class. The students incorporated these into their final project designs.

Triggers
- Distance of viewer to camera/sensor
- Specific color detection and location
- If motion is detected
- The frequency of noise in the room
- Chroma key filtering

Effects
- Change video speed
- Change display style
- Change video display
- Change video zoom
- Change video size
- Greenscreen

## 3.3 User Classes and Characteristics

The main users of this product are Professor Daniel Schlapbach and his Video Art students. While they were not directly exposed to the source code of the projects at this time, the source code is made to be easy to understand and use, even for non technical users. In the future, the source code may be presented to the students so they can explore more options than the programmer could describe or conceive.

## 3.4 Operating Environment

This product will be the most compatible using an Apple operating environment because of the relationship between the Processing environment and the built in video camera. Windows will support Processing, but the synchronization between the video camera and Processing will need to be reconciled by downloading the latest Apply QuickTime installation.

## 3.5 Design and Implementation Constraints

This integration of the computer science department and the video art class in the photography department is a new relationship. As computer scientists think more in terms of what is programmably possible, the artists think about what is creative and stretches the boundaries. Finding and communicating this line is a challenge, especially when the software is being tailored to the wants of the artists. Through rigorous testing, we are finding the limits of the program and the artists are working within those lines. Time has been a major constraint on the potential of this project. Integrating new hardware and unfamiliar software has proven to be very fine-tuned. Instead of creating a full user interface for the artists, an easy to use and flexible library will be developed instead.

## 3.6 User Documentation

The Processing tutorial is referenced in Section 1.5

# 4. External Interface Requirements

## 4.1 Hardware Interfaces

A video camera will need to be connected to the hard drive. An iMac with a built in video camera will be the most compatible with the Processing Environment. An Arduino board will also be required to take full advantage of the interactive options. The sensor will be capable of color recognition and detection, detecting the distance or proximity of a user, if motion has occurred, and measuring the frequency of noise in the room. This sensor box is shown in figure 1.

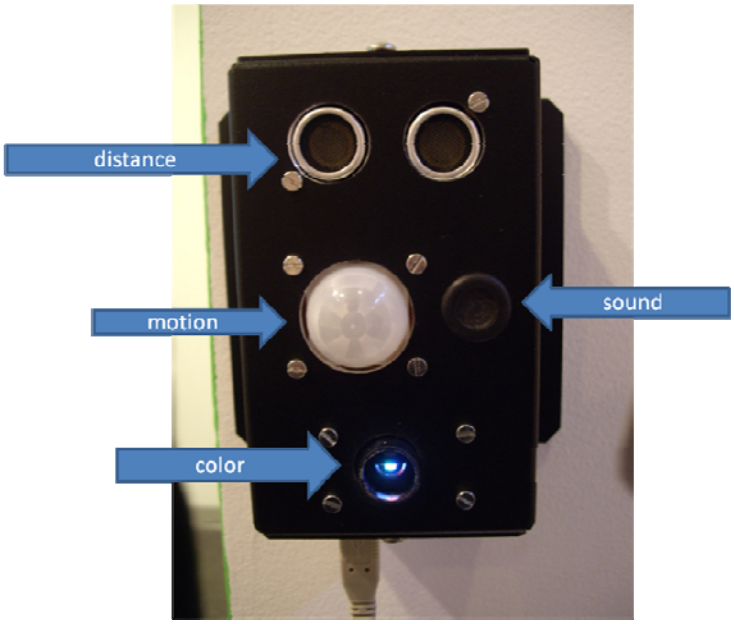



*Figure 1: Sensor box*

## 4.2 Software Interfaces

The software involved is the basic Processing environment. In addition are the following user created libraries: OpenCV for face detection and live video processing, GSVideo for displaying and manipulating videos faster, Serial for interacting with the sensor in the Arduino environment. Our extension combines these libraries to create a new one that offers interaction between the sensors and the way videos appear on the screen.

## 4.3 Communications Interfaces

If the product is going to implement a wireless video camera, a signal must be maintained. Otherwise, the communications are contained to the previously specified hardware and software elements.

# 5. System Features

## 5.1 Triggers

### 5.1.1 Viewer distance from sensor

*5.1.1.1 Description and Priority*
This trigger will allow the user to execute effects based on the distance of an observer to the video camera or Arduino sensor. This is a high priority feature of the system and can be used but not limited to the following effects: video speed, video zoom, video size.

*5.1.1.2 Stimulus/Response Sequences*
If this trigger is enabled, the program will continually compute the distance from the user to the video camera or the Arduino board. As the viewer gets closer or further from the screen, the effect will be executed accordingly.

*5.1.1.3 Functional Requirements*
To compute the distance with the video camera, the face detection analysis will be utilized. The face will be found using a built in Processing algorithm, and the result will be the box in which the face has been detected. The area of this box will be computed and how it varies in size will determine where the user is from the screen, i.e. smaller area, farther away. The Arduino board will use its sensors to determine the distance from the screen.

### 5.1.2 Color detection

*5.1.2.1 Description and Priority*
This trigger will allow the presence of specified colors to execute an effect. This is a high priority of the product interactions. This can allow a viewer to hold up different colored objects to the screen and trigger different response videos based upon the color of each item. This is a high priority feature to develop for the basic tool palette.

*5.1.2.2 Stimulus/Response Sequences*
If this trigger is enabled, the program will be searching for a blob of the specified color. Once that color is found, the effect will take place.

*5.1.2.3 Functional Requirements*
The video camera will record all the pixels in the view during a screen capture. The program will then look through all the pixels, trying to find the specified color in a blob that is at least as large as a certain threshold to avoid noise.

## 5.2 Effects

### 5.2.1 Video Speed

*5.2.1.1 Description and Priority*
This effect will change the speed at which a prerecorded video will play. This is a high priority feature to develop for the basic tool palette.

*5.2.1.2 Stimulus/Response Sequences*
Based on a specified trigger criteria, such as distance from the screen, the video will speed up or slow down.

*5.2.1.3 Functional Requirements*
This effect will be executed in the Processing environment with a simple command to the play speed of a video based on the trigger criteria.

### 5.2.2 Display Style

*5.2.2.1 Description and Priority*
This effect can alter the size of an individual cell of the video or live feed. In normal display, the cell size is the size of a single pixel so there is no distortion or alteration. This effect allows the cell to be scaled larger. This is a high priority feature to develop for the basic tool palette.

*5.2.2.2 Stimulus/Response Sequences*
Based on a specified trigger criteria, such as distance from the screen, the cell size will increase or decrease.

*5.2.2.3 Functional Requirements*
This effect will be executed in the Processing environment with a series of commands to create a cell from a pixel or resize a cell to encompass several pixels.

### 5.2.3 Video Zoom

*5.2.3.1 Description and Priority*
This effect will change how much a prerecorded video will be zoomed. This is a high priority feature to develop for the basic tool palette.

*5.2.3.2 Stimulus/Response Sequences*
Based on a specified trigger criteria, such as distance of a view from the screen, the video will zoom in or zoom out accordingly.

*5.2.3.3 Functional Requirements*
This effect will be executed in the Processing environment by specifying the amount of the video displayed on the screen based on the trigger criteria.

### 5.2.4 Video Size

*5.2.4.1 Description and Priority*
This effect will change the size of the video display on the screen, keeping all of the video intact and with the same dimensions. This is a high priority feature to develop for the basic tool palette.

*5.2.4.2 Stimulus/Response Sequences*
Based on a specified trigger criteria, such as distance of a view from the screen, the video will zoom in or zoom out accordingly.

*5.2.4.3 Functional Requirements*
This effect will be executed in the Processing environment by specifying the new dimension size based on the trigger criteria.

### 5.2.5 Adjust Video Display

*5.2.5.1 Description and Priority*
This effect can alter how the screen looks, either by altering a prerecorded video or the current video feed. Some options include color, black and white, brightness, contrast, and saturation. This is a high priority feature to develop for the basic tool palette.

*5.2.5.2 Stimulus/Response Sequences*
Based on a specified trigger criteria, the display will adjust accordingly. For example, if the trigger is ‘distance from screen’ and the effect is ‘adjust contrast’, as the viewer moves closer to the screen, the contrast in the picture will increase and decrease as they move backwards.

*5.2.5.3 Functional Requirements*
Depending on how the video display is to be changed will determine how the execution will take place. For most of the basic effects, color, black and white, brightness, contrast, and saturation, a simple command to the Processing environment will change the current display.

# 6. Other Nonfunctional Requirements

## 6.1 Performance Requirements

In order to capture live video feed to be used as a trigger, a video camera must be connected to and activated on the computer. Similarly, the Arduino hardware must be connected to the computer so that sensors can be read.

## 6.2 Safety Requirements

This product will involve screen(s) and a hard drive. The screens may be connected to the computer hardware directly, in the form of a laptop or a screen with a built in hard drive. These

screens may be mounted on the wall, the ceiling, or on the ground. Also, a projector may be used to project the scene onto a larger area. The possible danger in this is the safety of the mounted screens or projectors. They must be securely fastened into the wall as not to provide a danger to the exhibitors.

## 6.3 Security Requirements

There may be a violation of privacy involved if a video camera is activated in a public location without the knowledge of the people in the area. This product is set up to only involve an art display in a specified, closed location. People may be notified upon their entrance into the display exhibit that they may be captured on camera, but not recorded, for the exhibition.

## 6.4 Software Quality Attributes

This product is available to anyone who has access to an iMac with a built in video camera or an Arduino board. If the product is to be ported to another operating system, the user may have some trouble synchronizing the Processing environment with the video camera due to the Processing environment. Because the source code is provided in the earlier versions of the product, it can be adapted to include more functionalities and more fine tuned interactions by anyone with an understanding of the Processing language. There is no exact definition of ‘correctness’, but the program will respond to specified user interactions, such as color and distance, however the user defines. It will be flexible and robust so that users can choose from a variety of features and layer them or extract them. The product will be easy for users to use, the sections of source code will be clearly documented and specifications will be placed where the code should be edited and how it will respond. If the user wants to gain a better proficiency at the Processing language, they can visit the processing.org.

## 6.5 Business Rules

Any person interested in creating interactive video art can use this product; there are no restrictions, operating principles, or special circumstances involved.

# 7. Data design

The data from the various sensors is translated and used by the methods that modify videos. The sensor data can be classified as global data because it is available to all projects that implement our library.

## 7.1 Global data structure

Data that is used as triggers for the various projects are described below.

### 7.1.1 Color sensor data

Projects 1, 2, and 4 will use the color sensor. When an object is placed in front of the sensor, the sensor will read color values in red, green, and blue. These values are each three numbers and are put in the buffer as a 9 valued number. The color sensor is the most reliable sensor at this time.

### 7.1.2 Distance sensor data

Project 3 will implement the distance sensor. The sensor sends out waves and determines the time it took for them to be returned and returns this value. It is not scaled properly in terms of units, so the conversion is handled separately.

### 7.1.3 Sound sensor data

Project 3 will implement a sound sensor. However, the sensor built into the Arduino board is not very reliable. Instead, a library that makes use of the built in microphone in the iMac will be used.

### 7.1.4 Motion detector data

Project 4 will implement the motion detector. If motion is detected, a value of 9999 will be sent to the buffer. Otherwise, 0 is returned.

### 7.1.5 Greenscreen data

Project 2 will make use of the green screen effect. This will be implemented by chroma key filtering with an external web cam. This process begins by examining every pixel of the live video feed from the web camera. If the pixel falls within a certain threshold of the predefined green background, then the pixel is replaced by one of the video playing in the background.

# 8. Architectural and component-level design

At present, there are two classes in our project. Ultimately, it would be ideal to separate them further to create a complete standalone and importable library. The first class is called VideoMod and consists of modifications to a video or live feed. These functions can be called from the main class to modify the current video or feed. The main class consists of many calls to the sensor box to read the output information and format it so that our software can handle it. The draw() contains the method calls to read the sensor data and then calls to the VideoMod class to modify based on the sensor output.

## 8.1 Program Structure and Architecture

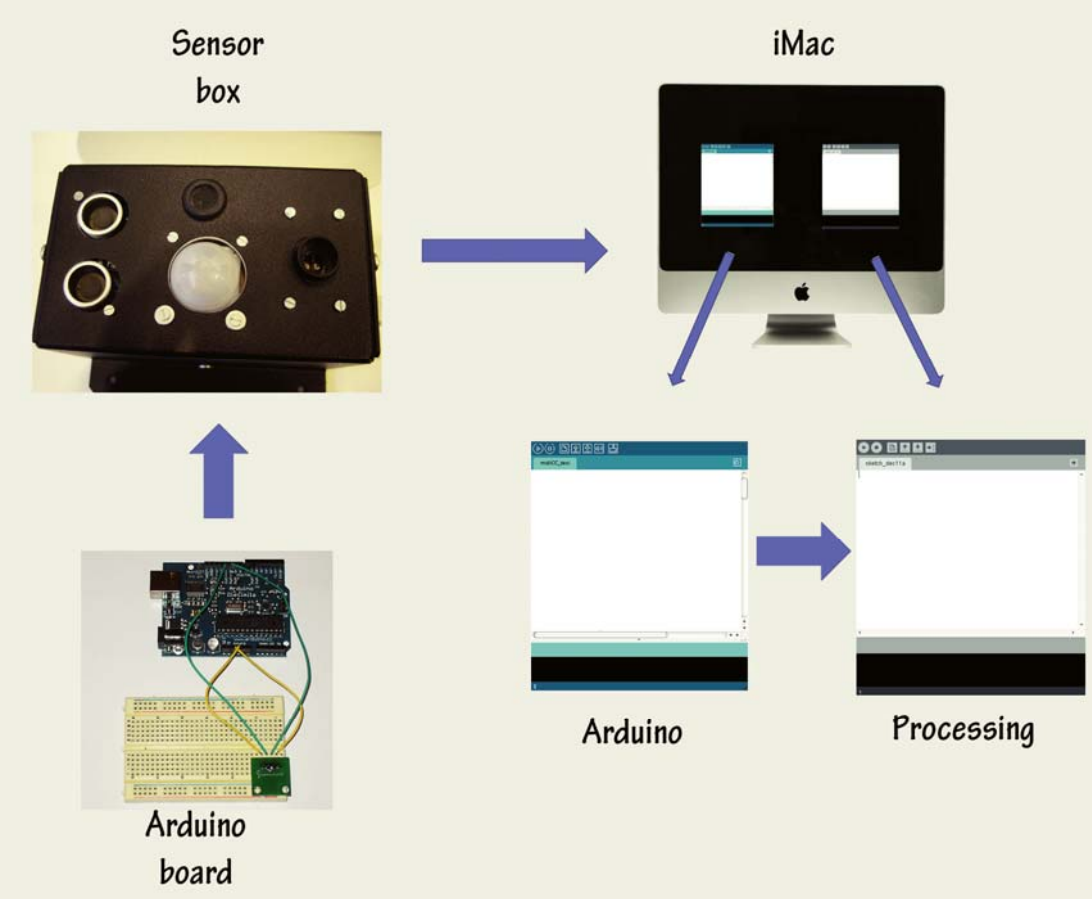



*Figure 2: Schematic*

Figure 2 shows the schematics for the project. It begins with the Arduino board, which Jason McMahon bought. For each board, he installed a color sensor detector, a distance detector, a

motion detection, and a sound detector. He wired these to the board and custom built the black box encapsulating the hardware. The sensor box is then connected to the iMac via USB cable. On each computer, the sensor box must be told how to read the data output by the sensor and determine its associations in the Arduino programming language. For our purposes, the raw data is described as follows: 9 digit integer represents a color detection; a 9999 indicates that motion was detected and a 0 otherwise; a number between 0 and 1000 is used for distance detection, as well as sound frequency levels. With this knowledge installed on each machine, the Processing environment can reference the sensor box as a serial port. On the Processing side, we parse the data so that it can be handled by the video mod functions. We read the data from the serial port and, for example, separate the 9 digit color values into 3 integers of red, blue then green.

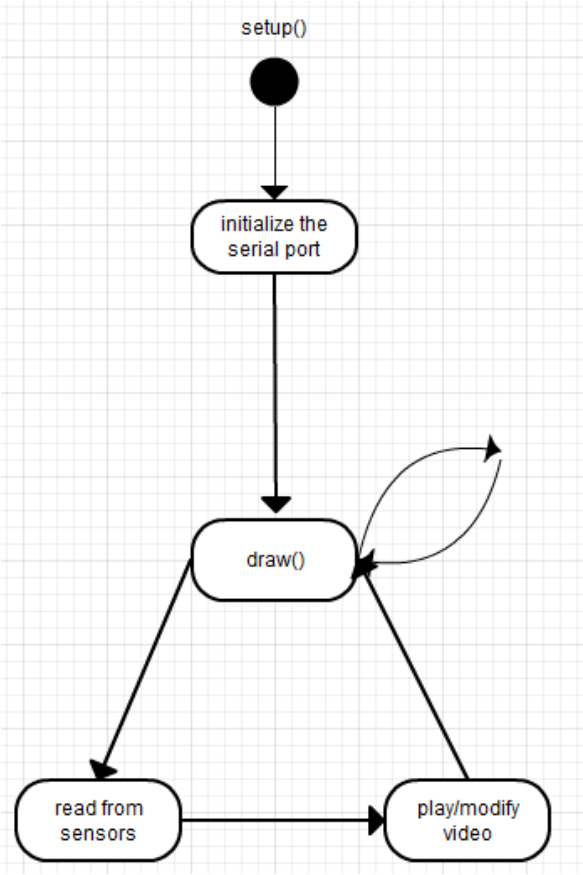



*Figure 3: Main class logic*

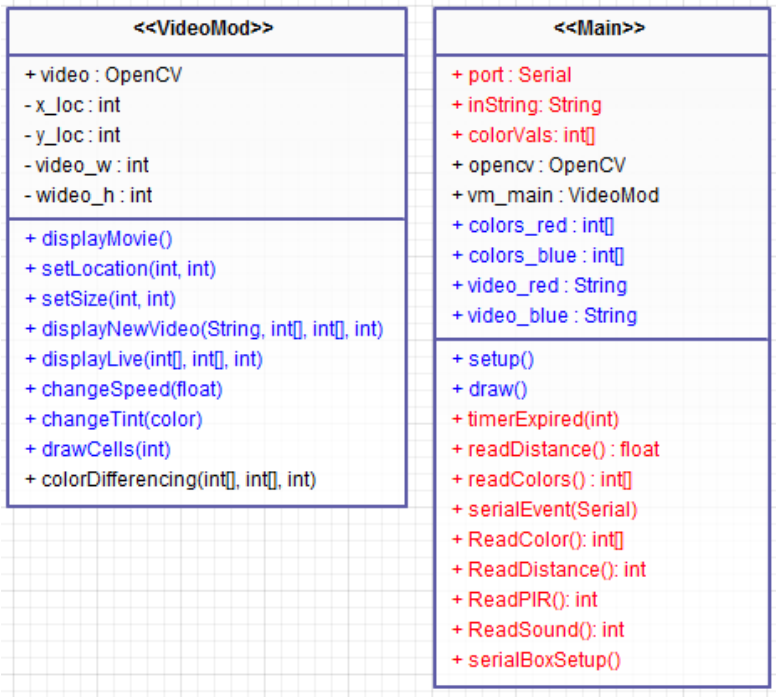



*Figure 4: Class diagrams of a program*

## 8.2 Description for Main

The blue items in the UML diagram in figure 4 are the only ones the user has to concern themselves with. In the main class, the user specifies what colors he or she will use, if any, and what video they will trigger when observed. They will also have the freedom to initialize the display to whatever they wish in the setup() function and the interactions take place in the draw() method. The methods that the user can call are the blue methods in VideoMod, which can move, resize, or trigger new videos based on specifications, such as color observation (colorDifferencing(...)). In the state diagram, the sensor is initalized in the setup() function and the draw function will continue to iterate. The typical project has data being read from a sensor during the draw, and when it meets a certain threshold criteria, an event is triggered within the VideoMod.

## 8.3 Description for Sensor

Figure 4 also shows the items responsible for managing the sensor output. These are highlighted in red. The ReadX() methods get the raw data from the sensor. The readX() methods scale the data so that it can be used for program manipulation. Typically, this sensor data is called in the draw() function and processed as the user wishes.

## 8.4 Description for VideoMod

The VideoMod class is an object which contains the location of the video on the screen as well as the size. These can be changed in draw() if the user so desires. Other operations to apply change are changing the speed of the video, overlaying color, pixel manipulation, pausing a video, switching to a new one, or enabling live feed. The output from the sensors are generally the input to the video manipulation.

## 8.5 Software Interface Description

While implementing the code, only the text based editor is used (see section 4.0). On the other hand, the viewer experiences a variety of different medium during the display and interaction with the projects.

### 8.5.1 External machine interfaces

Projects 1, 3, and 4 will be displayed on an iMac computer with built in video camera. Each project will have their own Arduino sensor box sitting next to the display. Project 2 will be displayed by a projector on a blank, white wall with a MacMini on top of the wall. The sensor box will be to the side of the projected display. Alongside the MacMini will sit a video camera to capture the motion in front of the green screen, which will hang opposite the projected screen.

# 9. User interface design

The project user interface is a simple text editor in the Processing Development Environment (PDE).

## 9.1 Description of the user interface

The Processing Development Environment (PDE), figure 5, consists of a simple text editor for writing code, a message area, a text console, tabs for managing files, a toolbar with buttons for common actions, and a series of menus. When programs are run, they open in a new window called the display window.

### 9.1.1 Screen images

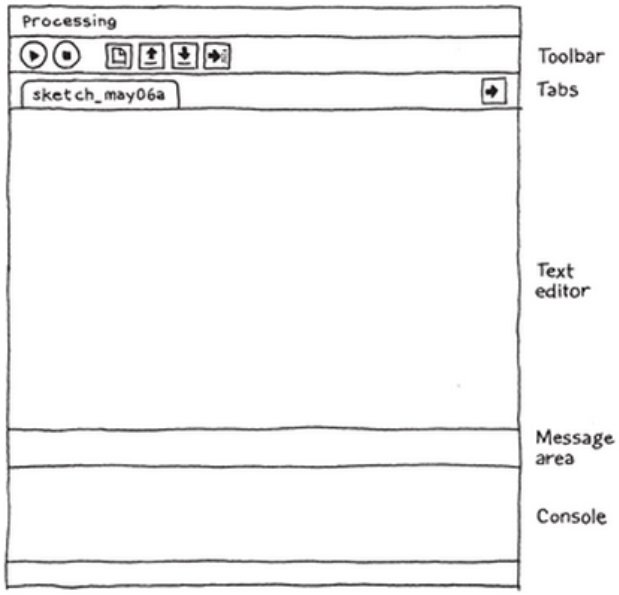



*Figure 5: Processing Development Environment*

### 9.1.2 Objects and actions

*Run.* Compiles the code, opens a display window, and runs the program inside.

*Stop.* Terminates a running program.

*New*. Creates a new sketch (project) in the current window.

*Open*. Provides a menu with options to open files from the sketchbook, open an example, or open a sketch from anywhere on your computer.

*Save*. Saves the current sketch to its current location.

*Export*. Saves the current sketch to its current location.

At the beginning of each program run, the *setup()* method is called once. After this initialization, the *draw()* method is called continually. This is where the major function calls to our developed library take place.

# 10. Restrictions, limitations, and constraints

Due to the aforementioned time constrains and the difficulty of creating a library in a new language with new hardware, a graphical interface could not be developed. To remain true to the Processing environment, a set of function calls have been developed which implement the new functionalities and interactions. Also, the software to read in the hardware output is still being developed for efficiency. Because of this, two or more sensors cannot be used at the same time to collect data from the physical world. Furthermore, the sensor uses a variable buffer to read data. Ideally, a fixed length buffer should be used because the data from the color sensor needs more bites than any other sensor. Because the buffer has not yet been changed, the color sensor cannot be used in conjunction with any other sensor during program run.

In the future it would be ideal to extract all the sensor and video interaction code and create a true library. Because of time constraints, the sensor manipulation is in the same location as the main draw() function in the program.

# 11. Test Plan

There are three types of tests to be performed: unit testing, integration testing, and full scale system testing. They are detailed below:

## 11.1 Testing strategy

Before the four, specific video art projects were conceived, the video manipulation functions were created and tested on their own. There are five types: video placement, video pause, video pixilation, video speed, and video tint. There were three kinds of testing for these functions: testing with static values which were acceptable input, invalid input, or boundary conditions as well as variable values and then with real input values (unit tests), followed by integration with the project idea and finally with the system in the gallery.

The unit tests tested acceptable values, the boundary values, as well as invalid values and will consist of white and black box testing. After the projects were crafted, the functions were written in the main class to perform accordingly. This integrated the building blocks with the structure of the each project. Finally, the projects will be tested in the gallery as a full system.

### 11.1.1 Unit testing

A unit is an instance of a VideoMod, which contains a location, a size, and the string where the video lives. Video placement can be tested by specifying the x and y locations statically (white box) or variably (black box). Video pause is a simple boolean toggle statement which can be triggered statically (white box) or based on certain criteria (black box). Video pixilation and swirling was tested with static values indicating cell size and how to rotate, as well as variable criterion. Similarly, video speed was tested by static values indicating the play back speed as well as

variables. And finally, video tint is tested similarly. After it was confirmed that the functions could perform as they should, input from the sensor was used.

### 11.1.2 Integration testing

Once the video manipulation functions were tested for efficiency, the projects were created, each with its own main class. The setup() and draw() functions were crafted to perform what each project hoped to achieve, integrating the sensors, video cameras, and green screens. Each project is tested separately by manually observing if the project is behaving as it is expected.

### 11.1.3 High-order or System testing

After the projects are working in the lab, each will be ported to their own iMac or MacMini system. Once installed in the gallery, they will be tested in that setting. Tests to perform will include the tests to verify the system performance in the integration testing phase.

## 11.2 Testing tools and environment

The goal for each individual project is ultimately an aesthetically pleasing display with fast video performance. The Processing library and user libraries have already been tested for bugs and efficiency. The testing that must be done on this project is mainly the integration between the sensor data and the existing function calls and achieving a smooth interaction between the hardware and software.

## 11.3 Test schedule

Unit tests began as soon as the video manipulation coding began. Unfortunately, the sensor box wasn't available as early as we had hoped. Therefore, the real input tests are still being performed. At the same time, the projects have begun. They began on Tuesday, November 9th after meeting with the video art students and discussing their project ideas. The sensor box testing began a week before that on Monday, November 2nd. This coincides with our projected prototype delivery dates. On Tuesday, November 23rd, the prototype will be displayed again. At this time, all the projects are well into development and being tested for performance. The gallery installation will begin on Monday, November 22nd, and the projects will be ported on Monday, November 29th. The gallery opens on Wednesday, December 1st.

# 12. Test Procedure

The functions for the video manipulation were created with the possible sensor input in mind. Static values and mouse position values were first used to establish the boundaries of acceptable input to the functions. Then, the sensor data was manipulated and constrained to conform to the expected acceptable input. In order to determine what kind of output the sensors gave and then how to mold it to something to the functions can use, the sensor data was rigorously tested by itself.

## 12.1 Unit test cases

As mentioned in Section 6, there are five video functionalities to test at the unit level. In addition, the sensor output must be examined in order to be properly formatted for function input.

### 12.1.1 Unit test input and expected result for video placement

The static test began by simply altering the x and y values which place the video on the screen. Values on the screen and off the screen were tested. Then, the mouse x and y location was used to determine the location of the video. In our current development, the sensor data does not directly determine the location of the video although it may be easily altered and made to do so in the future.

### 12.1.2 Unit test input and expected result for video pause

This is a simple boolean which pauses or plays the video. It was statically tested with true or false statements, followed by meeting criteria from the color sensor which will be discussed in detail in the color sensor unit test section.

### 12.1.3 Unit test input and expected result for video pixilation

The static test began by drawing an outline around each individual pixel in the video. The cell size was changed statically so that the cells could grow larger. The rotation was also hardcoded as a function of the location of the cell. Next, the mouse position indicated how to increase or decrease the cell size, as well as how to rotate them. In our current development, the sensor data does not directly determine the location of the video, although it may be easily altered and made to do so in the future.

### 12.1.4 Unit test input and expected result for video speed

Video speed was first tested with static values. The mouse x or y location was then used to determine the speed at which the video played. A possible option for the video projects was the have the speed alter with the views distance from the sensor. Careful tresholding and smoothing needed to take place so that the video would not sporadically change speed if the sensor misread a value.

### 12.1.5 Unit test input and expected result for video tint

The tint or brightness of the video was first altered statically by specifying color values. Then, the tint could be changed more dynamically with respect to the location of the mouse on the screen. The sensor input could be distance from the screen or the color the color sensor observes.

### 12.1.6 Unit test input and expected result for color sensor

The color sensor will only read data when an object is close enough to the screen. Sometimes it will misread a color as black if the object is removed too slowly from the screen. This caused problems with the tint overlay, but a check was added to examine the previous color and to ignore black. To determine the color the sensor thought an object was, we printed the color the sensor saw. We averaged the values and hardcoded it as a color to check while running the program. If the currently observed color is within a specified threshold (+- x points for red, green, and blue each), then the color matches.

### 12.1.7 Unit test input and expected result for distance sensor

The distance sensor works well up until a certain distance where the range is too wide and it begins to detect further objects as being close. A smoothing factor was placed on the distance data before it was taken in as input to the video manipulation functions.

### 12.1.8 Unit test input and expected result for motion sensor

The motion sensor will return 9999 if motion is detected and 0 otherwise. The sensor is very sensitive to a very small amount of motion. To test if it was accurately detecting motion or lack thereof, the sensor was placed behind the computer when the test was run. No motion was indicated until an object actually moved past the sensor.

## 12.2 Integration testing

After the video manipulation functions have been tested and the sensor data understood, the integration can take place. Each project has its own specifications and functions to use.

### 12.2.1 Test cases for integration testing for Color project

The color values to be used will be determined by the sensor and hardcoded into the program. When the difference between the observed value and a predetermined value is less than the threshold, the video will change. All of the performed tests have been successful.

### 12.2.2 Test cases for integration testing Projector project

The green screen project will also implement the aforementioned color sensor. In addition, a chroma key filter will be used to subtract the predetermined background color from the display. The threshold depends upon the lighting in the room and cannot be completely perfected until installation.

### 12.2.3 Test cases for integration testing Loop and Pixilation project

This project will play a video and then play another one in its same location. After successfully attaining a looping two videos, the pixilation will be hardcoded. After assessing it will work on the looping videos, the clapping sensor will be the trigger rather than the command to pixelate.

### 12.2.4 Test cases for integration testing Music project

When the color sensor observes the hardcoded colors, the video will change to the corresponding video as specified in the Color project. Concerning the motion detector, the sensor is continually searching for motion. When it is found, a timer is set. With each iteration of the draw() function, the timer is decreased. Upon reaching 0, the sensor looks again for motion. If found, the timer resets and begins the countdown. Otherwise, another video is displayed until the colors are detected again.

## 12.3 High-order testing (System Testing)

The full system cannot be tested until the hardware is installed in the gallery. But the following will be taken into consideration when the time comes.

### 12.3.1 Performance Testing

Once the projects have been installed, the lighting in the gallery will need to be taken into consideration because it will affect the performance of the video cameras. The projection project is the most important to test in this aspect. The hardcoded background color tested in the lab must recognize the same background once we have installed it in the gallery.

### 12.3.2 Load/Stress testing

These displays will be running continuously in the art gallery. We will need to ensure that the computers do not become overheated. Once the installation has been complete, the projects will be run and tested for endurance length.

# 13. Reflections

## 13.1 Clientele and Requirements Collection

The decision to step out of my comfort zone and take on a software development project that included faculty and students in the fine arts department was a good one. This made me realize that clients really don’t know what they want; at the start of the project it was the idea a vague ‘let’s do something cool’. After we began searching for more concrete ideas, I very much appreciated that the faculty and students kept their minds open and continued to bring forth engaging ideas. However it soon became overwhelming because there were too many possibilities. If we didn’t start soon, we could have spent the entire semester pondering what could be rather than narrow it

down to what we will actually do. Once the solid requirements were established, the work could begin.

## 13.2 Change in Requirements and Time Estimation

It is difficult to determine if the requirements have changed, because from the start this project was open to many possibilities. But once we started to figure out what kind of interactions and features we wanted, these were solid. Some additional features the students discussed were not implemented due to time restraints, but in future galleries would be very possible to implement. The time estimation was very accurate. Since the project was dependent on the video art students for ideas and content, the program structure and functionality had to be presented to the students early enough to let them think of ideas, consider what is possible or not, then put the finishing touches on the projects. The full functioning product was delivered on time, but some additional features were lacking due to an unforeseen setback.

## 13.3 Design

Without a proper design process, my project would have been easy to program from a technical level. Rather than designing a reusable and simple structure, hard coding each project independent of the others wouldn't have taken the entirety of the semester. However, I did not take this simplified approach. I realized that there is potential for this program to be used again, incorporating new features and project structure. Knowing that my clients were nontechnical, I designed the program in such a way to make it easy to use and understand with little knowledge of programming.

## 13.4 Testing

The testing performed at the unit level was successful. The video playback methods were tested with small sample video files using OpenCV as the video playback and live feed capture. During integration testing, the small sample videos were used as placeholders for the videos the students would provide for each project. The first student videos were received the Monday of Thanksgiving break, giving us a week and a half to put the finishing touches on the projects for the gallery opening. The OpenCV player we were using up until now worked well with low resolution videos, but was not compatible with the higher resolution videos provided by the students. To change the video player was not a small feat. Almost the entirely of the program design needed to be restructured because of the differences between OpenCV and the new video library, GSVideo. If I had known from the start to use GSVideo, I could have properly structured my program around it, rather than having to create new methods to be compatible with my current framework.
It would also have been more helpful to have more than 3 days to install the programs on the computers as well as physically assemble the gallery. Each sensor box seemed to be slightly different, especially when detecting colors; a color that was calibrated to one sensor box was not always the same on another. In the future, it would be ideal to have each iMac and sensor box available for installation and testing before we even enter the gallery.

## 13.5 Future Galleries: Guidelines for the Future

Overall, the gallery was a huge success. Many students and faculty were very impressed with the concept and the projects featured. The fundamentals have been laid and lessons have been learned so that a future gallery can implement what we have right from the start and have the remainder of the semester to build additional features or optimize the current features. The following is a summary of things to consider at the start of the next development process:

a. Use GSVideo to process videos from the start
b. Export videos at 1280x720 in HD, only if individual pixel processing is not used. In that case, use a lower resolution
c. Test the color sensor for bogus input. We developed a pattern matching filter to disregard invalid input, but this should be stress tested.
d. Test the motion sensor. It worked correctly when it did work, but it sometimes did not detect motion when there was motion. We tried to read the motion data for an interval of time rather than just once, but this caused the videos playing to lag.
e. Test the sound sensor. It was not properly implemented and tested in time for it to be used for the gallery opening
f. Try background subtraction for the greenscreen rather than chroma key filtering. It may be better for edge detection of an object.
g. Use a web camera that does not implement auto white balance for the greenscreen.
h. Allow users the option of taking screenshots of their interactions with a greenscreen or other displays that show the viewer interacting on the screen.