Worcester Polytechnic Institute

“Jikken”: High-Speed Camera Control and Bubble Counting Software

Software Design Document for Major Qualifying Project at Shibaura Institute of Technology, Tokyo, during the B22 Term

Uri Dvir

December 16, 2022

Table of Contents

1.0 Introduction

1.1 Purpose

This document lays out the design for the *Jikken* project. The design is intended solely for the project advisor, Professor Weinstock, for the purpose of overseeing the completion of this MQP. In other words, it is for internal use and not client use. The document should be read alongside the requirements document and the specification document.

1.2 Overview of Document

The **Class Diagram** shows the components of the program, C++ classes. wxWidgets and OpenCV are object-oriented libraries, and this project's codebase is written in an object-oriented style. The **Description** describes how each class operates, and its role in the context of the full program. The **Overview of Requirements** section, like in the specification document, enumerates how the design satisfies the established requirements.

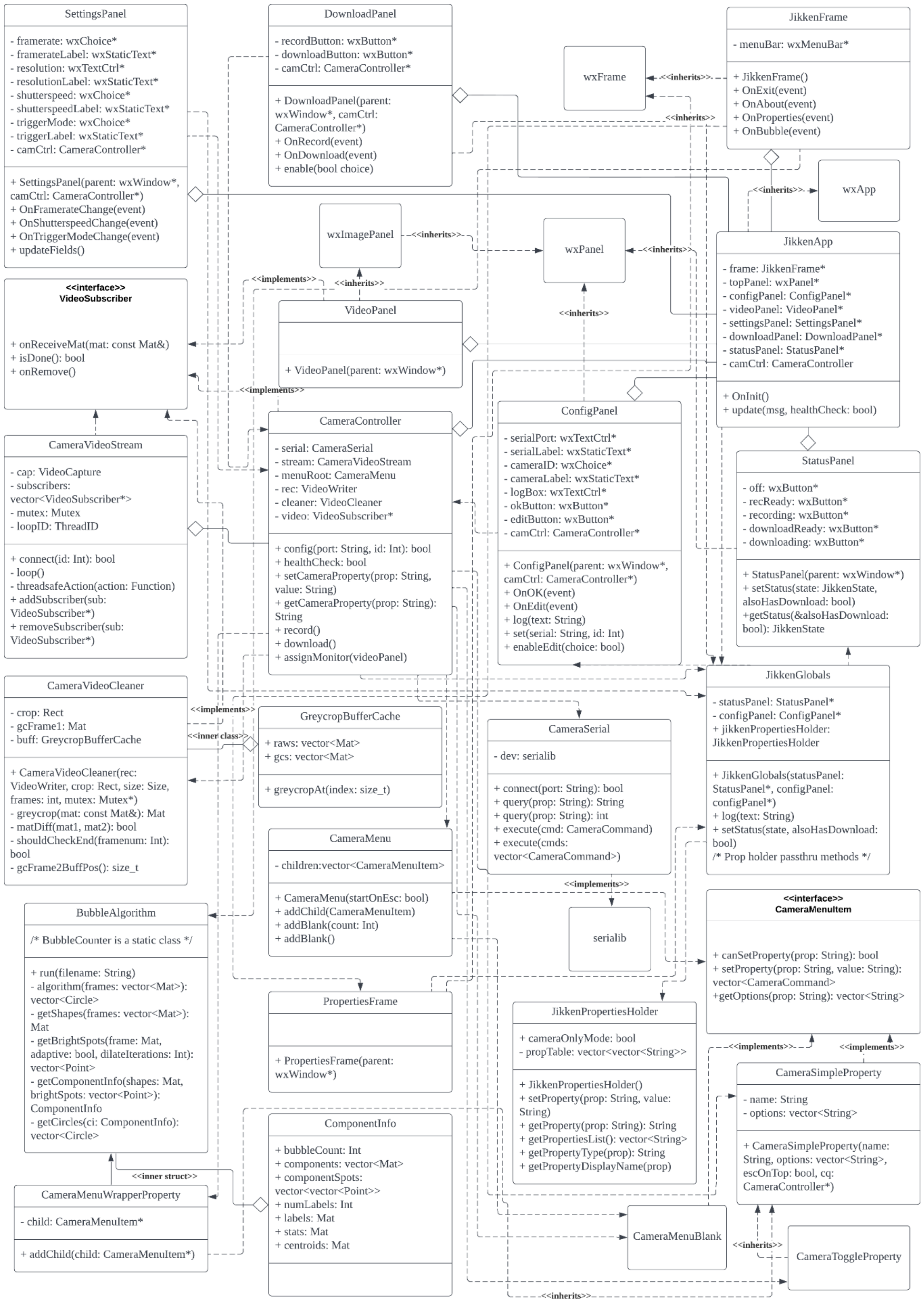
1.3 Version History

Document Version	Date	Changes	Other Notes
0.1	Dec 7, 2022	First version	N/A
0.2	Dec 16, 2022	Accounting for design changes in the implementation	N/A

6.0 Design

6.1 Class Diagram

See the following page for the diagram. An extra note about StatusPanel, ConfigPanel, JikkenApp, and CameraController: these classes implement the pure virtual classes StatusSetter, Logger, MainManager, and CameraQuerier respectively for the sole purpose of avoiding circular dependencies leading to linker errors. Those pure virtual classes are omitted from the class diagram for simplicity.


SettingsPanel
- framerate: wxChoice*
- framerateLabel: wxStaticText*
- resolution: wxTextCtrl*
- resolutionLabel: wxStaticText*
- shutterspeed: wxChoice*
- shutterspeedLabel: wxStaticText*
- triggerMode: wxChoice*
- triggerLabel: wxStaticText*
- camCtrl: CameraController*
+ SettingsPanel(parent: wxWindow*, camCtrl: CameraController*)
+ OnFramerateChange(event)
+ OnShutterspeedChange(event)
+ OnTriggerModeChange(event)
+ updateFields()
DownloadPanel
- recordButton: wxButton*
- downloadButton: wxButton*
- camCtrl: CameraController*
+ DownloadPanel(parent: wxWindow*, camCtrl: CameraController*)
+ OnRecord(event)
+ OnDownload(event)
+ enable(bool choice)
JikkenFrame
- menuBar: wxMenuBar*
+ JikkenFrame()
+ OnExit(event)
+ OnAbout(event)
+ OnProperties(event)
+ OnBubble(event)
wxFrame
<<inherits>>
<<inherits>>
wxApp
<<inherits>>
wxImagePanel
<<inherits>>
wxPanel
<<inherits>>
JikkenApp
- frame: JikkenFrame*
- topPanel: wxPanel*
- configPanel: ConfigPanel*
- videoPanel: VideoPanel*
- settingsPanel: SettingsPanel*
- downloadPanel: DownloadPanel*
- statusPanel: StatusPanel*
- camCtrl: CameraController
+ OnInit()
+ update(msg, healthCheck: bool)
<<interface>>
VideoSubscriber
+ onReceiveMat(mat: const Mat&)
+ isDone(): bool
+ onRemove()
<<implements>>
<<inherits>>
VideoPanel
+ VideoPanel(parent: wxWindow*)
<<inherits>>
<<implements>>
CameraVideoStream
- cap: VideoCapture
- subscribers: vector<VideoSubscriber*>
- mutex: Mutex
- loopID: ThreadID
+ connect(id: Int): bool
- loop()
- threadsafeAction(action: Function)
+ addSubscriber(sub: VideoSubscriber*)
+ removeSubscriber(sub: VideoSubscriber*)
CameraController
- serial: CameraSerial
- stream: CameraVideoStream
- menuRoot: CameraMenu
- rec: VideoWriter
- cleaner: VideoCleaner
- video: VideoSubscriber*
+ config(port: String, id: Int): bool
+ healthCheck: bool
+ setCameraProperty(prop: String, value: String)
+ getCameraProperty(prop: String): String
+ record()
+ download()
+ assignMonitor(videoPanel)
ConfigPanel
- serialPort: wxTextCtrl*
- serialLabel: wxStaticText*
- cameraID: wxChoice*
- cameraLabel: wxStaticText*
- logBox: wxTextCtrl*
- okButton: wxButton*
- editButton: wxButton*
- camCtrl: CameraController*
+ ConfigPanel(parent: wxWindow*, camCtrl: CameraController*)
+ OnOK(event)
+ OnEdit(event)
+ log(text: String)
+ set(serial: String, id: Int)
+ enableEdit(choice: bool)
StatusPanel
- off: wxButton*
- recReady: wxButton*
- recording: wxButton*
- downloadReady: wxButton*
- downloading: wxButton*
+ StatusPanel(parent: wxWindow*)
+ setStatus(state: JikkenState, alsoHasDownload: bool)
+getStatus(&alsoHasDownload: bool): JikkenState
CameraVideoCleaner
- crop: Rect
- gcFrame1: Mat
- buff: GreycropBufferCache
+ CameraVideoCleaner(rec: VideoWriter, crop: Rect, size: Size, frames: int, mutex: Mutex*)
- greycrop(mat: const Mat&): Mat
- matDiff(mat1, mat2): bool
- shouldCheckEnd(framenum: Int): bool
- gcFrame2BuffPos(): size_t
<<implements>>
<<inner class>>
GreycropBufferCache
+ raws: vector<Mat>
+ gcs: vector<Mat>
+ greycropAt(index: size_t)
CameraSerial
- dev: serialib
+ connect(port: String): bool
+ query(prop: String): String
+ query(prop: String): int
+ execute(cmd: CameraCommand)
+ execute(cmds: vector<CameraCommand>)
JikkenGlobals
- statusPanel: StatusPanel*
- configPanel: ConfigPanel*
+ jikkenPropertiesHolder: JikkenPropertiesHolder
+ JikkenGlobals(statusPanel: StatusPanel*, configPanel: configPanel*)
+ log(text: String)
+ setStatus(state, alsoHasDownload: bool)
/* Prop holder passthru methods */
CameraMenu
- children:vector<CameraMenuItem>
+ CameraMenu(startOnEsc: bool)
+ addChild(CameraMenuItem)
+ addBlank(count: Int)
+ addBlank()
<<implements>>
serialib
<<interface>>
CameraMenuItem
+ canSetProperty(prop: String): bool
+ setProperty(prop: String, value: String): vector<CameraCommand>
+getOptions(prop: String): vector<String>
BubbleAlgorithm
/* BubbleCounter is a static class */
+ run(filename: String)
- algorithm(frames: vector<Mat>): vector<Circle>
- getShapes(frames: vector<Mat>): Mat
- getBrightSpots(frame: Mat, adaptive: bool, dilateIterations: Int): vector<Point>
- getComponentInfo(shapes: Mat, brightSpots: vector<Point>): ComponentInfo
- getCircles(ci: ComponentInfo): vector<Circle>
PropertiesFrame
+ PropertiesFrame(parent: wxWindow*)
JikkenPropertiesHolder
+ cameraOnlyMode: bool
- propTable: vector<vector<String>>
+ JikkenPropertiesHolder()
+ setProperty(prop: String, value: String)
+ getProperty(prop: String): String
+ getPropertiesList(): vector<String>
+ getPropertyType(prop): String
+ getPropertyDisplayName(prop)
<<implements>>
<<implements>>
CameraSimpleProperty
- name: String
- options: vector<String>
+ CameraSimpleProperty(name: String, options: vector<String>, escOnTop: bool, cq: CameraController*)
<<inner struct>>
CameraMenuWrapperProperty
- child: CameraMenuItem*
+ addChild(child: CameraMenuItem*)
ComponentInfo
+ bubbleCount: Int
+ components: vector<Mat>
+ componentSpots: vector<vector<Point>>
+ numLabels: Int
+ labels: Mat
+ stats: Mat
+ centroids: Mat
CameraMenuBlank
<<inherits>>
CameraToggleProperty
<<inherits>>

6.2 Description

Class	**Details**
JikkenApp	At the very top of the hierarchy. JikkenApp sets the sizer and adds children to JikkenFrame. JikkenApp holds pointers to the panels and owns the CameraController.
JikkenFrame	JikkenFrame gets its children set by JikkenApp, however it is fully responsible for creating and managing the menu bar. This corresponds to the main window.
JikkenGlobals	As the name suggests, the global singleton for this program. This is the only globally assigned variable. It has pointers to the status and config panels (for logging and setting status). It also holds the JikkenPropertiesHolder, because many subsystems regularly check program properties/settings (auto download, etc.).
JikkenPropertiesHolder	Holds program properties. Properties set in the properties panel are included, while camera settings are not considered properties in this case. However, camera configuration settings (serial and ID #) are in fact held by JikkenPropertiesHolder. Properties are always kept in sync with file on disk.
PropertiesFrame	A subwindow which lets the user set program properties. The class is extensible in the sense that adding properties is as simple as updating propTable.
ConfigPanel	Lives on the left side of the main window. The user can set serial port and camera ID up top, and the log text view is in the middle. At the bottom there is an OK and edit button. One button is always locked. If “OK” is locked, the camera is connected. If “Edit” is locked, the controls are ready for new config settings. ConfigPanel calls config() in CameraController.
VideoPanel	Lives in the center of the main window and shows the live feed from the Kodak Motion Corder. VideoPanel is usually subscribed to CameraVideoStream (see below to see what this means).
VideoSubscriber	Pure virtual class implemented by classes that receive video frames from CameraVideoStream. onReceiveMat() is called when there’s a new frame. If the implementor is done intaking frames, they should return true in the isDone()

	function, which will automatically unsubscribe them. When unsubscribed, onRemove() will be called.
CameraVideoStream	Manages live video feed from the camera. Video subscribers are fed frames as they come in. Notably, the acquisition loop runs in its own thread, so video subscribers must keep good multithreading hygiene. The loop blocks on a mutex when the video stream is down / not set up.
SettingsPanel	Lives on the right side of the window. The user can set framerate, shutter speed, and trigger mode. Unlike the config panel, there is no OK button to confirm: the settings controls are synced with the current settings on the Kodak Motion Corder. Calls setCameraProperty() in CameraController.
DownloadPanel	Lives directly below the settings panel. The user can record and download video. Calls record() and download() in CameraController.
StatusPanel	Lives on the bottom of the window and shows program state to the user.
CameraController	Controls the camera and owns the CameraVideoStream. It also owns CameraSerial and VideoCleaner. Its child, mainMenu, represents the top page in the camera menu/settings screen. CameraController calls setProperty() in mainMenu to set framerate, resolution, and trigger mode. CameraController simply uses the UP and DOWN controls on the camera to set shutter speed, since this setting is not located in the Kodak Motion Corder menu screen. CameraController subscribes to CameraVideoStream while it is configuring the camera, to check the health of the video stream. It unsubscribes when camera setup is done.
CameraSerial	Sends commands to the camera over serial to simulate button presses and uses query codes (also over serial) to get current property values. Must be manually initialized with connect().
CameraMenuItem	We consider the camera's menu as a tree, with the submenus being the nodes. Submenus that set properties are considered leaves. The nodes have type CameraMenuItem. A CameraMenuItem can do one thing: set a

	property. When it's told to set a property, it will return an ordered list of inputs that will perform the setting needed. A CameraMenuItem can also be asked whether it can set a given property. Since we are working with a tree, the setProperty() and canSetProperty() methods go down the tree in a depth first search, calling the corresponding methods in other nodes.
CameraMenu	A type of CameraMenuItem which represents a page in the menu. Pages don't have properties, but their children might. Some pages start off with the cursor selecting ESC. This class supports both types.
CameraMenuBlank	A type of CameraMenuItem which represents a part of the camera menu not implemented by the navigation logic. Therefore, the program should treat it as a menu element that is present but cannot be interacted with.
CameraSimpleProperty	A type of CameraMenuItem which represents a property page in the menu. This is like a normal page, but each option sets the property to a different value. Some properties have ESC at the top, and some at the bottom. This class supports both types.
CameraToggleProperty	A type of CameraMenuItem which supports properties that toggle through their options when hitting ENTER. These properties don't have a page that opens, and there is no ESC. It is derived from CameraSimpleProperty and has all methods and fields in common, simply with different behavior.
CameraMenuWrapperProperty	A type of CameraMenu item supporting properties scrolled with the UP and DOWN keys which have a menu-type child. This is implemented for the sake of the shutter speed property.
CameraVideoCleaner	When downloading video from the camera, cleans frames and saves them to a video file. It omits lead-in and loop-to-start when saving the video.
GreycropBufferCache	A buffer/cache combo for video frames that have been converted to greyscale and cropped, used internally by VideoCleaner. Doing this on all the copies of frame #1 at the start is wasteful, hence the buffer/cache.
BubbleAlgorithm	Implements the bubble-counting algorithm. It takes in a video filename, in run(), and outputs its

	results in a folder with matching name to the video file.
ComponentInfo	A struct which collects various results from OpenCV's connected component finder, used internally by BubbleCounter.

6.3 Overview of Requirements

1. Implemented using the BubbleCounter class.
2. Implemented with DonwloadPanel and its dependencies.
3. Implemented with SettingsPanel and its dependencies.
4. GUI top-level class is JikkenApp.
5. Implemented with VideoPanel and its dependencies.
6. May be implemented in CameraController.
7. Manual control menu is not implemented (optional requirement).
8. Implemented with PropertiesFrame and the property table in JikkenPropertyHolder.
9. Implemented with ConfigPanel and its dependencies.
10. Text is in Japanese, which satisfies the stated requirement (English is non-essential for the client, who are Japanese). There is no English mode.
11. As stated in the specification document, OpenCV encoder support is relatively good. Thus, MP4 is the current choice of format, notwithstanding issues which necessitate a format change.
12. CameraController ensures recorded video has no display text. VideoCleaner subscribes to CameraVideoStream and writes to a video file.
13. Same as stated in the specification. Implemented in BubbleCounter.

In addition to the above, a README is provided to the client with detailed instructions for setting up project dependencies. These are the Visual C++ Redistributable, OpenCV, and wxWidgets. These instructions tell the client how to install DLLs and update environment variables.

Appendices

Appendix A: README

じっけんプログラムが DLL エラーから始まれませんなら、このページをお読んでください。

じっけんプログラムには、「Microsoft Visual C++ Redistributable 2022」と「OpenCV 4.6.0」と「wxWidgets 3.2.1」が必要です。こちらの物をしなさい：

「Microsoft Visual C++ Redistributable 2022」を「https://aka.ms/vs/17/release/vc_redist.x64.exe」からインストールしなさい。

「OpenCV 4.6.0」を「https://opencv.org/releases/」からインストールしなさい。インストーラーが新しいフォルダーを作ります、“opencv”。このフォルダーをローカルディスク（例えば“C:￥”）へ運びなさい。“C:￥opencv￥build￥x64￥vc15￥bin”をシステム環境変数の Path に付け加えなさい。

「wxWidgets 3.2.1」を「https://www.wxwidgets.org/downloads/」からインストールしなさい。“Download Windows Binaries”をクリックしなさい。64 ビットのオプションから、“Release DLLs”をクリックしなさい。ZIP ファイルから、新しいフォルダーを作って、”C:￥“へ運びなさい。このフォルダをシステム環境変数の Path に付け加えなさい。

英語訳もあります：

If the program cannot start and gives DLL errors, please carefully follow these instructions.

Jikken Program needs Microsoft Visual C++ Redistributable 2022, OpenCV 4.6.0, and wxWidgets 3.2.1 to be installed. Please do the following:

Install Microsoft Visual C++ Redistributable 2022 from https://aka.ms/vs/17/release/vc_redist.x64.exe.

Install OpenCV 4.6.0 from https://opencv.org/releases/. The installer will create a folder, “opencv”. Move this folder to C:\. Now, open the folder C:\opencv. Open these folders in order: build >> x64 >> vc15 >> bin. You are now in the correct folder. Add this exact folder to the system environment variable “Path”.

Install wxWidgets 3.2.1 from https://www.wxwidgets.org/downloads/. Click “Download Windows Binaries”. In the 64-bit section, click “Release DLLs”. Unzip the file into a folder in C:\. Add this exact folder to the system environment variable “Path”.