DESIGNING A SECURITY WEBCAM SYSTEM



Voicu Sara-Ioana

Technical University of Cluj-Napoca

Group 30431

*CONTENTS*

1. Introduction
   1. Context
   2. Specifications
   3. Objectives
2. Bibliographic study
   1. History
   2. Drivers
   3. Motion detection
3. Design
   1. UML Diagrams
   2. Graphical Interface
4. Implementation

4.1 Classes

1. Testing

5.1 Types of software testing

5.2 Testing the security webcam system

1. Conclusion
2. *Introduction*

1.1 Context

Security is one of the most important problems all over the world. Today, in our society, it is one of the major issues and having a 24\*7 human eye is just impossible.

This project concerns on the use of automatic motion detection applications using webcams. The purpose is to create a system that allows the user to see on the Internet an object, a room and to be alerted with a sound in case the object is moving or something strange is happening. For example, if we have an empty room and someone is entering the room, the user will hear an alerting sound.

This can be helpful in many situations: when we want to supervise an empty room, so that no one should be entering it, for example thieves, when we want to look after a sleeping baby and to know when he/she is moving or waking up etc.

1.2 Specifications

For this project I will work in Java programming language, and I will use the webcam of my laptop to capture images. I will study and use several libraries such as: *JavaCV library, webcam-capture library.*

Java is the most powerful high level programming language, an object-oriented programming language that uses interfaces, classes, and functions. Usually, Java doesn’t provide easy access to computer hardware, that’s why it might be tough to access the webcam. But this can be done with the help of the mentioned libraries, that allow us to capture images by accessing the webcam.

**JavaCV-platform library** provides a Java implementation of the OpenCV project. It uses wrappers from the JavaCPP Presets of commonly used libraries by researchers in the field of computer vision and provides utility classes to make their functionality easier to use on the Java platform, including Android. JavaCV comes with hardware accelerated full-screen image display, user-friendly geometric and color calibration of cameras and projectors, a set of classes that implement direct image alignment of projector-camera systems.

**Webcam-capture library** allows you to use your build-in or external webcam directly from Java. It’s designed to abstract commonly used camera features and support various frameworks. This library has many features: offers ready to use motion detector, offers possibility to expose images as MJPEG streams, swing component to display video feed from camera, swing component to choose camera etc.

1.3 Objectives

* Use the webcam of personal laptop to see live images from an empty room
* Detect motion or anything that is moving/disturbing the view
* Trigger an alarm in case of detection and alert the user
* Take pictures when motion is detected and save them along with the date and time

1. *Bibliographic study*

The library that I decided to use for my project is Webcam-Capture.

2.1 History

The creator initially started working on Webcam Capture as a simple proof-of-concept after reading Andrew Davison’s fantastic book entitled Killer Game Programming. He found out how Java APIs were not allowing you to capture images from webcams very easily. Once you choose a specific API, you cannot change it without modifying large parts of the code. So, he decided to change this situation and write general purpose wrapper for various APIs (like JMF, OpenCV, VLC etc.). We can now change underlying frameworks just by replacing the webcam driver. If there is no driver for a particular framework, it is very easy to write it yourself.

2.2 Drivers

Webcam Capture API defines WebcamDriver interface which has been also implemented in several capturing drivers build on top of well-known frameworks used to work with multimedia and cameras. The default driver consists of small, refined part of OpenIMAJ framework wrapped in thread-safe container. Other drivers that will be used further in the project are:

* **JMF Driver -** uses Java Media Framework to access UVC webcam devices
* **JavaCV Driver –** uses JavaCV binding for OpenCV to gain access to the UVC camera device

2.3 Motion detection

Motion can be detected in many ways using the libraries described above. For example, there are many comparison approaches, such as frames that are converted from RGB format to GRAY scale format and then compared. Again, the frames are converted back to RGB format before storing. The gray scale conversion is done to decrease the pixel values. Whereas the frames are compared pixel by pixel. The difference in the frame gives only the moving objects on the frame.

It can also be detected with the WebcamMotionDetectorAlgorithm interface, which provides many methods to calculate the difference between two images, representing the movement or not. For this project, I decided to use the WebcamMotionListener public interface, where the motion listener is used to signal motion detection. It has only one method, which will be called after the motion is detected. It contains one parameter representing the motion event.

[**motionDetected**](https://javadoc.io/static/com.github.sarxos/webcam-capture/0.3.12/com/github/sarxos/webcam/WebcamMotionListener.html#motionDetected-com.github.sarxos.webcam.WebcamMotionEvent-)**(**[**WebcamMotionEvent**](https://javadoc.io/static/com.github.sarxos/webcam-capture/0.3.12/com/github/sarxos/webcam/WebcamMotionEvent.html)**wme){}**

1. *Design*
   1. UML Diagrams

An UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts, or classes, to better understand, alter, maintain, or document information about the system. I will illustrate the Use-Case Diagram (Behavioral UML Diagram) and the Class Diagram (Structural UML Diagram).

The **Use-Case Diagrams** are used to analyze the system’s high-level requirements. We have three main components: functional requirements, actors, and relationships.

Diagram

Description automatically generated

**Use Case 1**: see live images from the camera Primary actor: user Main Success Scenario: the user can see live images from the room that he/she wants to supervise

**Use Case 2**: hear an alerting sound when the motion is detected Primary actor: user Main Success Scenario: the user will hear an alerting sound when someone enters the room

**Use Case 3**: receive a picture taken in the moment the motion was detected, along with the hour and date Primary actor: user Main Success Scenario: the user receives a screenshot from the time when the motion was detected, so that he/she can see the person that entered the room. He/she will also receive the date and time from that exact moment.

The **Class Diagram** is the most common diagram type for software documentation. Since most software being created nowadays is still based on the Object-Oriented Programming paradigm, using class diagrams to document software turns out to be a common-sense solution. This happens because OOP is based on classes and the relations between them. Class diagrams contain classes, alongside with their attributes (data fields) and their behaviors (member functions). Each class has 3 fields: the class name at the top, the class attributes right below the name, the class operations/behaviors at the bottom. The relation between different classes (represented by a connecting line) makes up a class diagram.

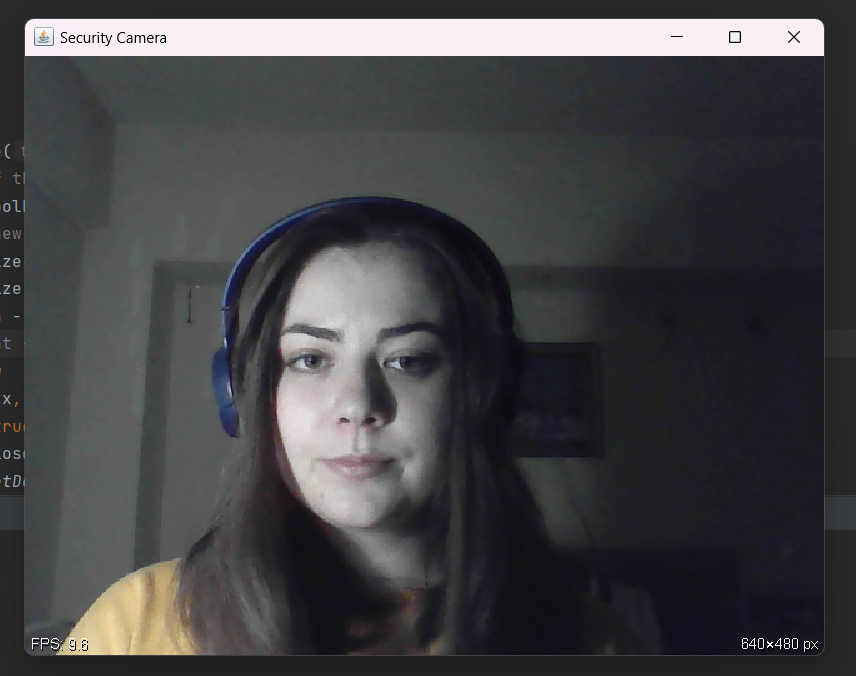
Diagram

Description automatically generated

* 1. Graphical Interface

Graphical User Interface (GUI) in Java is an easy-to-use visual experience builder for Java applications. It is mainly made of graphical components like buttons, labels, windows etc. through which the user can interact with an application. GUI plays an important role to build easy interfaces for Java applications.

I created my interface in the WebcamGUI class. This class extends the JFrame class which is a type of container that works like the main window where components like labels, buttons, text fields are added to create the GUI. I created a webcam object with the help of Webcam-Capture library and then I set all the attributes for my webcam and added it to the frame. The graphical interface looks as below and displays a live video, using the webcam from my laptop.



1. *Implementation*
   1. Classes

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

A Class is like an object constructor, or a “blueprint” for creating objects.

My project contains 4 classes: Main, WebcamGUI, Alarm and DetectMotion.

* ***Main Class:***

Main class has only one method main () where I instantiated the WebcamGUI and DetectMotion objects, so that the user can play with the application.

* ***WebcamGUI Class:***

This is the class where the graphical interface is created. It extends the JFrame Class. Inside the constructor it has a frame that displays the live image. I also selected some attributes for the frame like size, mirrored, title etc.

**A person taking a selfie

Description automatically generated with medium confidence**

The frequency measured by frames per second(fps) is displayed in the left corner of the webcam. In the right corner the resolution is displayed.

* ***Alarm Class:***

This class contains a method named playSound (), that helps us to play an alarming sound, when we are calling it. For the sound, I added a file “alarm\_sound.wav” to the project. To play it I used an AudioInputStream, a Clip and the methods open () and stop ().

The sound only plays for 3 seconds when it detects the motion, then it stops. When motion is detected again, it plays again.

Text

Description automatically generated

* ***DetectMotion Class:***

This class implements the WebcamMotionListener Interface. This interface is used to signal motion detection and it has one attribute, an object of type Alarm and one method motionDetected (WebcamMotionEvent wme). This method is called when the motion is detected. Besides these, the class also contains a constructor:

**Text

Description automatically generated**

When the motion is detected, I will play an alarming sound, so that the user will know that something is wrong, and a person/object appeared in the room. In the motionDetected() method I will call the method alarm.playsound() in order to play the sound.

By using the Files class, I also created a file where I save save date and the hour when the motion was detected. Also, the live photos are taken when the motion is detected, and the pictures are saved in different files:

Text

Description automatically generated

1. *Testing*

5.1 Types of software testing

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs, and improving performance.

There are many different types of software tests, each with specific objectives and strategies. **Acceptance testing** means verifying whether the whole system works as intended, **integration testing** ensures that software components or functions operate together, **functional** **testing** checks functions by emulating business scenarios, based on functional requirements, usability testing validates how well a customer can use a system or web application to complete a task.

5.2 Testing the security webcam system

The main aim of video surveillance security cameras is to protect your customers both directly and indirectly. They record suspicious activity and let the user know when and where it happened.

The system of this project will detect any motion, then play an alarming sound and take a picture of the live camera in the exact moment when the motion was detected. The picture will be saved in a file.





Javax.imageio is a package that allows us to load images from an external image format into the internal BufferedImage format used by Java 2D. The WebcamLibrary allows us to capture the image by using the method webcam.getImage().

In another file, I would collect data about the time and date when the motion was detected.

Text

Description automatically generated

1. *Conclusion*

This project was good for remembering OOP concepts that I learned last year. I also discovered new libraries and how to use them, like WebcamCapture library, how to access the webcam from my laptop, how to take pictures and see live images.

Besides this, I managed to solve a real problem that many people encounter nowadays. I designed a webcam security camera, that has an alarm and takes live photos when the motion is detected. In the future, I would like to save a video instead of just a picture when the motion is detected and maybe give the object that is moving a different color (red, for example).

This project was done with the help of the next sources:

* <https://github.com/sarxos/webcam-capture>
* <https://www.youtube.com/playlist?list=PLhs1urmduZ28_IFafEsXNq3fjdqXLfpu>
* <https://www.youtube.com/watch?v=TErboGLHZGA&t=394s>