Manual

Design

Programmet är uppdelat i två delar. En serverdel som körs på kameran och en klientdel som körs på användarens dator.

Kamera Protokoll

Servern kan skicka två typer av meddelande till klienten. Den första byten anger typ av meddelande, 0 för *motion-mesage* och 1 för *image-message*. Ett *motion-mesage* anger att kameran har upptäckt rörelse och gått in i *movie-mode*. Ett *image-message* innehåller en bild och tillhörande information på följande format

Image Message			
1 Byte	8 Bytes	4 Bytes	Length
Message Type	Timestamp	Length	JPEG

- Message Type har värdet 1 för *image-message*, som beskrivet ovan.
- Timestamp anger tiden vid vilken bilden togs som long givet av kameraetoden `public void getTime(byte[] target, int offset)`
- Length anger längden på bilden som int givet av kamerametoden `public int getJPEG(byte[] target, int offset)`
- JPEG innehåller bilden som hämtades från kameran.

Klienten skickar vid anslutningen texten *CLIENT* följt av ett *carriage return* för att identifiera sig. Klienten kan även skicka 5 olika meddelande för att kontrollera serverns beteende, dessa är på följande format

1 Byte
Message

Servern kan hantera följande meddelanden

- `CLIENT_IDLE_MESSAGE = 0` Anger att serverna ska gå in i *idle-mode*.
- `CLIENT_MOVIE_MESSAGE = 1` Anger att serverna ska gå in i *movie-mode*.
- `CLIENT_FORCE_NONE = 127` Upphäver effekten av `CLIENT_FORCE_MOVIE` och `CLIENT_FORCE_IDLE`.
- `CLIENT_FORCE_MOVIE = 126` Tvingar server att stanna i *movie-mode*.
- `CLIENT_FORCE_IDLE = 125` Tvingar server att stanna i *idle-mode*.

Server

Servern kan hantera två typer av anslutningar. Dels anslutning med en klient enligt nedan, där bilder skickas kontinuerlig, samt HTTP-requests där endast en bild skickas.

När servern är ansluten till en klient skickas bilder i olika takt beroende på vilket tillstånd servern befinner sig i. Möjliga tillstånd är

- *Movie-mode* - Bilder skickas så snabbt som möjligt.
- *Idle-mode* - En bild var 5:e sekund.

Övergången mellan tillstånd beskrivs i avsnittet *Serverns Livscykel*.

UML-diagram

[DIAGRAM]

Klasser

`Server` är en tråd som innehåller main metoden. Klassen skapar alla andra trådar och monitorer. Därefter lyssnar den efter inkommande anslutningar. Vid anslutning inväntas ett meddelande som anger vilken typ av anslutning det är, *GET* för HTTP-request och *CLIENT* för klientanslutning. Vid ett HTTP-request hanteras anslutningen direkt av `Server`. Vid klientanslutning skickas klientens socket till `Monitor` och övriga trådar tar över ansvaret.

`CameraThread` är en tråd som hämtar bilder och tidsstämpel från kameran i full hastighet och kontrollerar rörelsedetektorn. Skapar ett `ImagePackage` utifrån hämtad data och placerar i `Monitor`.

`ImagePackage` motsvarar ett *image-message*. Innehåller header och JPEG som byte array enligt ovanstående beskrivning för ett *motion-message*.

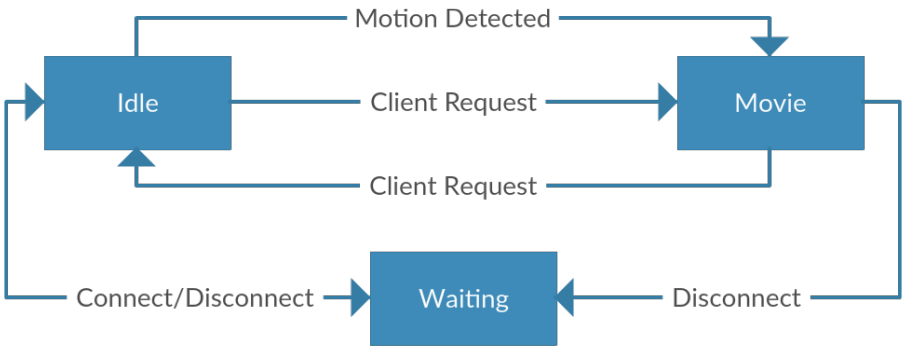
`InputHandler` är en tråd som hanterar `InputStream` delen av anslutningen. När ingen klient är ansluten är den blockerad i väntan på anslutning. När en klient är ansluten tar den emot meddelanden och anropar motsvarande metoder i `Monitor`.

`OutputHandler` är en tråd som hanterar `OutputStream` delen av anslutninge. När ingen klient är ansluten är den blockerad i väntan på anslutning. När en klient är ansluten hämtar den `ImagePackage` från `Monitor`, ett var femte sekund i idle-mode och så snabbt som möjligt i movie-mode. Dessa skickas sedan till klienten. När servern går in i *movie-mode* skickas även ett *motion-message* till klienten.

`Monitor` hanterar serverns tillstånd och gemensamma data. Exempelvis

- Om det finns någon klient ansluten.
- Aktuellt mode (idle/movie).
- Senaste `ImagePackage` från `CameraThread`.

Serverns Livscykel



Servern kan befinna sig i tre tillstånd waiting, idle och movie. Vid uppstart är alla trådar blockerade i väntan på anslutning till klient. När en klient har anslutit sig befinner sig servern initialt i idle-mode. Övergång till movie-mode kan ske på två sätt.

1. `CameraThread` upptäcker rörelse i senaste bilden och anropar monitor metoden `setMovieMode(true)`.
2. `InputHandler` tar emot ett movie-mode meddelande från klienten och anropar`setMovieMode(true)`.

Övergång från movie- till idle-mode sker när `InputHandler` tar emot ett idle-mode meddelande från klienten. Servern återgår till waiting när anslutning till klienten bryts.

Client

Klienten kan hantera hantera 2 serveranslutnignar samtidigt och har ett grafiskt användargränsnitt för att visa bilderna och kringliggande information. Beroende

på fördröjningen mellan bilderna som tas emot kan visningen ske

- *Synkront* - Med fast tidsfördröjning från när bilden togs.
- *Asynkront* - Så fort som möjligt efter att bilden har tagits emot.

UML-diagram

Klasser

`Client` innehåller main metoden och skapar alla andra klasser och monitorer.

Varje kamera använder en egen instans av klasserna `ConnectionMonitor`, `InputHandler`, `MessageHandler` och `MessageBuffer`.

`ConnectionMonitor` är en monitor som hanterar anslutning med en server. Har metoder som blockerar `InputHandler` och `MessageHandler` när ingen server är ansluten.

`InputHandler` är en tråd som har ansvar för `InputStream` delen av anslutningen med en server. Den skapar ett `CameraImage` objekt utifrån data som läses på strömmen och placerar det i `ImageBuffer`. Om ett *movie-message* erhålls från servern placeras ett `CLIENT_MOVIE_MESSAGE` i den andra serverns `MessageBuffer`.

`MessageHandler` är en tråd som har ansvar för `OutputStream` delen av anslutningen med en server. Den hämtar meddelande från `MessageBuffer` och skickar de till servern.

`MessageBuffer` är en monitor som innehåller meddelanden som ska skickas till en server. Möjliga meddelandetyper beskrivs i avsnittet *Kamera Protokoll*. Meddelanden placeras i buffern på grund av användarinteraktion eller av `InputHandler` enligt ovan.

`ImageBuffer` är en monitor som innehåller en prioritetskö med `CameraImage` objekt sorterad efter tidsstämpel. Klassen har även ansvar för att hålla reda på om visning av bilder ska ske synkront eller inte.

`ImageUpdater` är en tråd som hämtar bilder från `ImageBuffer` och uppdaterar användargränsnittet.

`Camera Image` motsvarar en bild som hämtats från servern samt information om denna, bland annat tidsstämpel och vilken server den kommer ifrån.

Utöver dessa finns ett antal GUI klasser.

- `GUI` motsvarar huvudfönstret.
- `ImagePanel` visar bilder och information om dessa.
- `ConnectionHandling` motsvarar knapparna som används för att ansluta till en server.

- `ConnectionDialog` motsvarar dialogrutan där man väljer vilken server man vill ansluta till.
- `DebugFrame` är fönstret med extra inställningar gällande idle/movie-mode samt synchronized/asynchronized-mode.