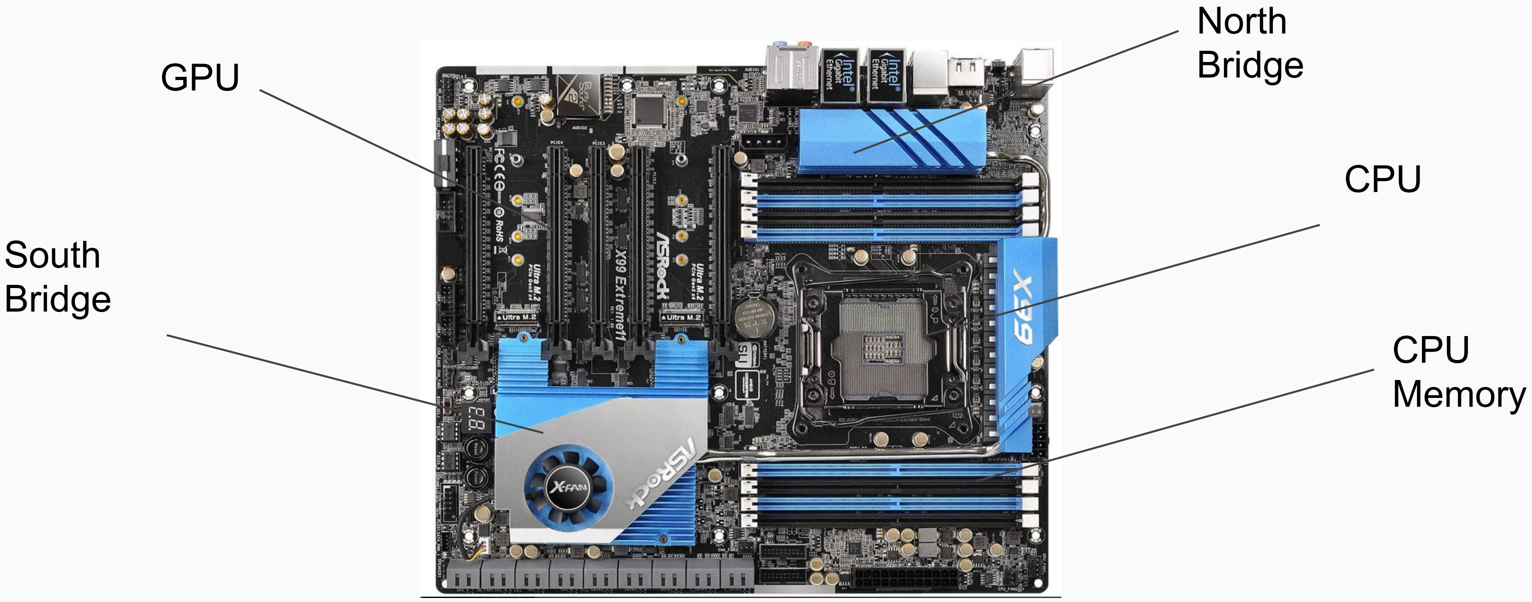
Introduction to Multimedia

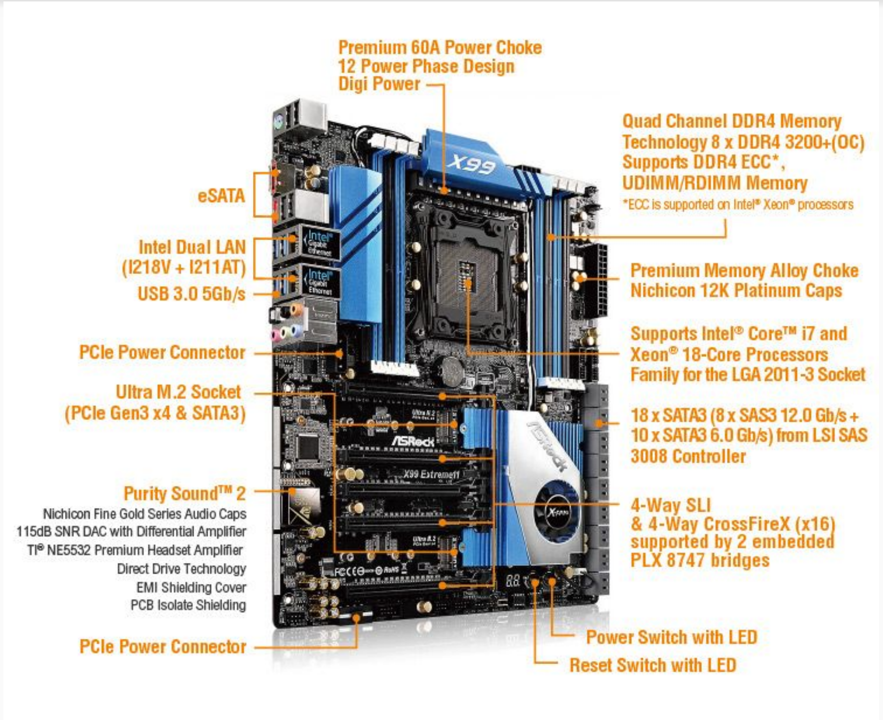
Integration with hardware

Topology of hardware in a system



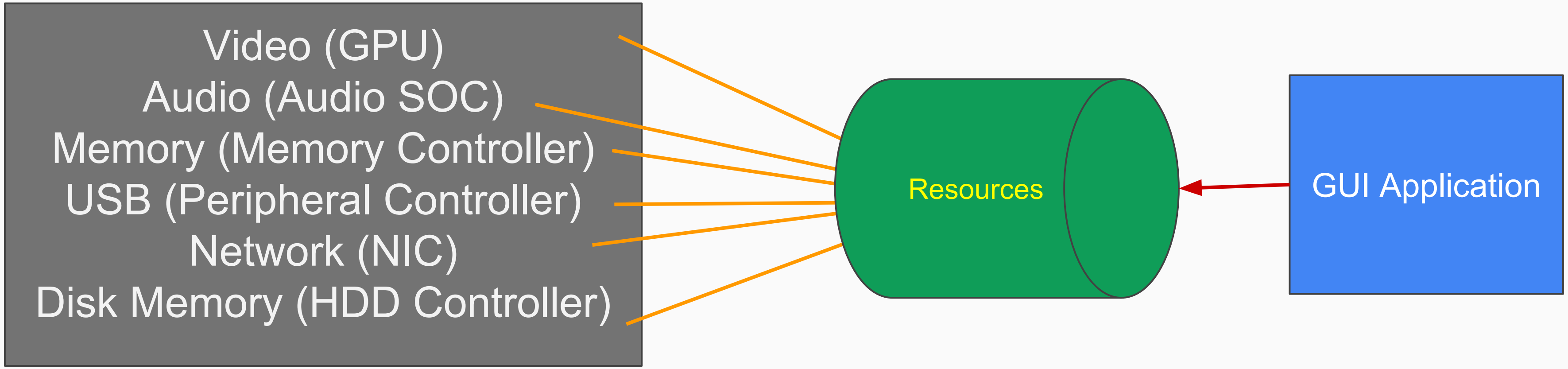
Internal ASICs form a network of chips(Resources)

Video (GPU)
Audio (Audio SOC)
Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)



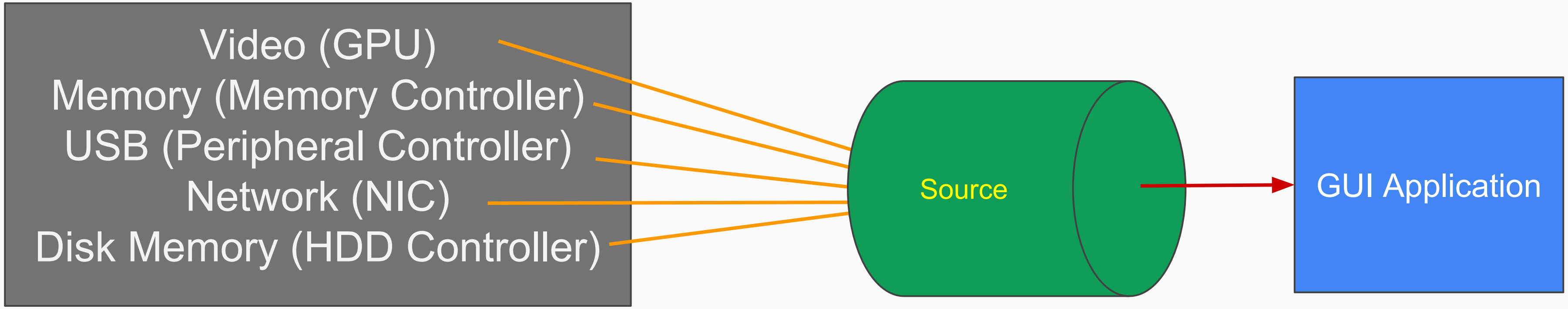
Premium 60A Power Choke
12 Power Phase Design
Digi Power
Quad Channel DDR4 Memory Technology 8 x DDR4 3200+(OC) Supports DDR4 ECC*, UDIMM/RDIMM Memory
*ECC is supported on Intel® Xeon® processors
eSATA
Intel Dual LAN (I218V + I211AT)
USB 3.0 5Gb/s
Premium Memory Alloy Choke
Nichicon 12K Platinum Caps
PCIe Power Connector
Supports Intel® Core™ i7 and Xeon® 18-Core Processors Family for the LGA 2011-3 Socket
Ultra M.2 Socket (PCIe Gen3 x4 & SATA3)
18 x SATA3 (8 x SAS3 12.0 Gb/s + 10 x SATA3 6.0 Gb/s) from LSI SAS 3008 Controller
Purity Sound™ 2
Nichicon Fine Gold Series Audio Caps
115dB SNR DAC with Differential Amplifier
TI® NE5532 Premium Headset Amplifier
Direct Drive Technology
EMI Shielding Cover
PCB Isolate Shielding
4-Way SLI & 4-Way CrossFireX (x16) supported by 2 embedded PLX 8747 bridges
PCIe Power Connector
Power Switch with LED
Reset Switch with LED
X99
ASRock
X99 Extreme11

GUI concepts -Resource

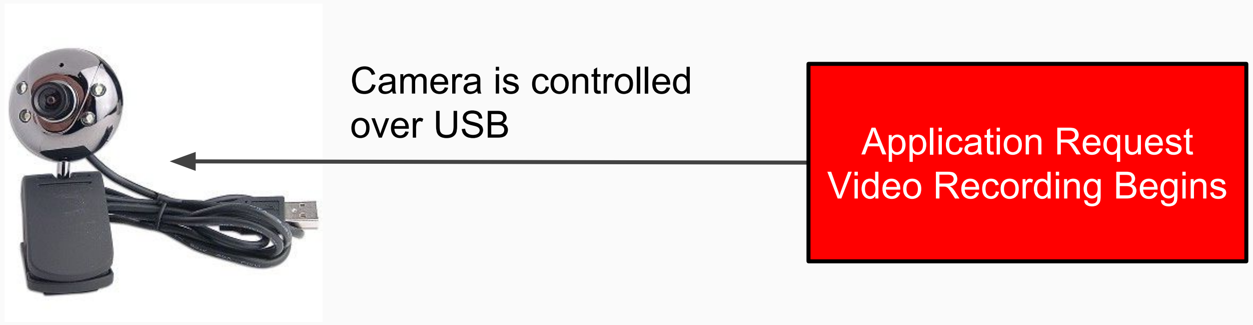

Video (GPU)
Audio (Audio SOC)
Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)
Resources
GUI Application

GUI Concepts - Sources

After Allocation of resources, the hardware becomes a source for the Graphical Application


Video (GPU)
Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)
Source
GUI Application

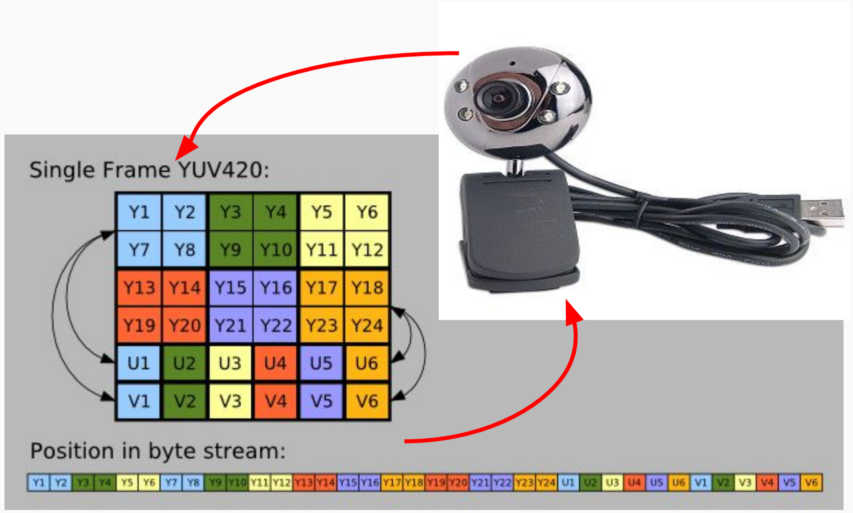
Webcam Example


Camera is controlled over USB
Application Request Video Recording Begins

Camera is enabled as a Resource

Webcam Example

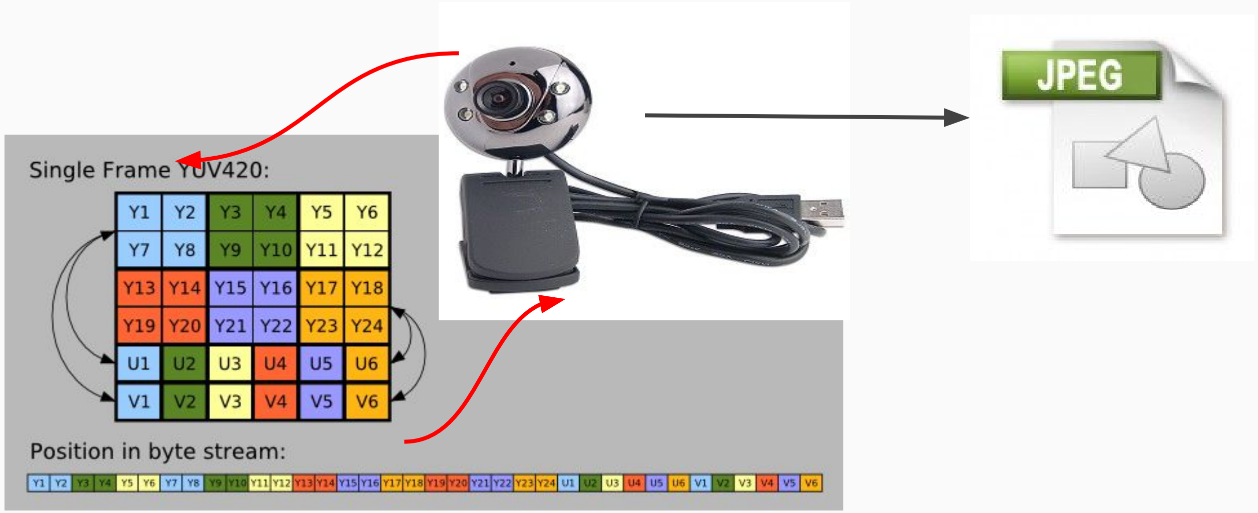
The Webcam handles the operations of capturing the image without the help of the computer



The image data is stored in an efficient data format

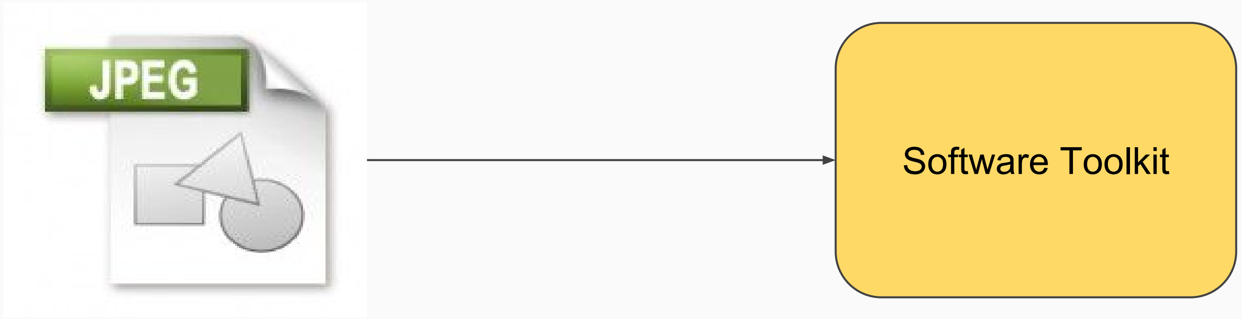
Webcam Example

Camera becomes a source that produces a JPEG image at a negotiated “framerate”.



Webcam Example

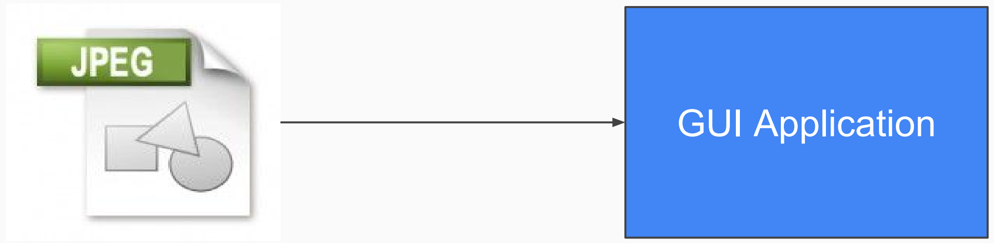
Why is the conversion important?



The software toolkit likely does not handle YUV format natively. JPEGs and Bitmaps are easier for toolkits to convert to a visual output. (Draw to a framebuffer)

Webcam Example

The toolkit (or driver interface) will then hand the application the data.



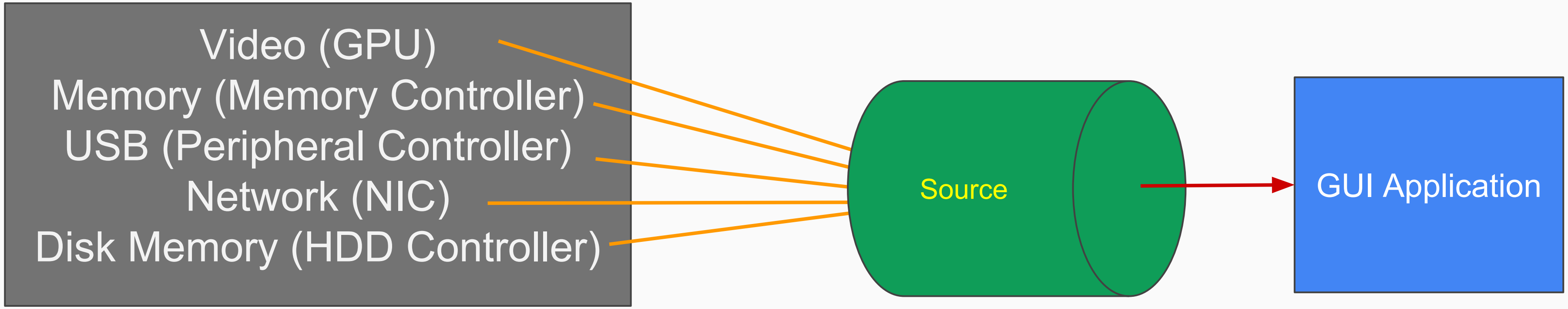
More Reading on Camera (Qt/QML)
http://doc.qt.io/qt-5/cameraoverview.html

The application can then decide what to do with image -

- Display it
- Write it to a file
- Send it over a network
- Analyze it - OpenCV?

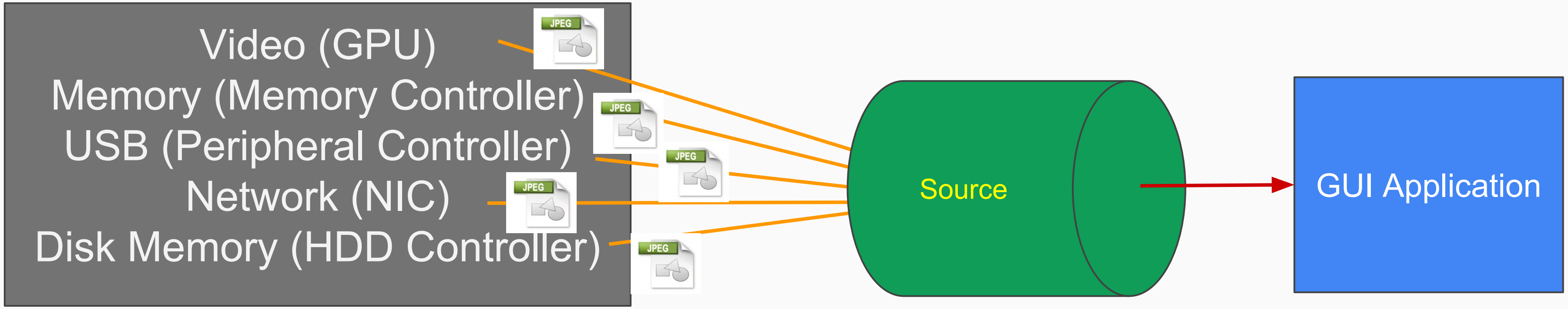
GUI Concepts - Sources

After Allocation of resources, the hardware becomes a source for a Graphical Application


Video (GPU)
Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)
Source
GUI Application

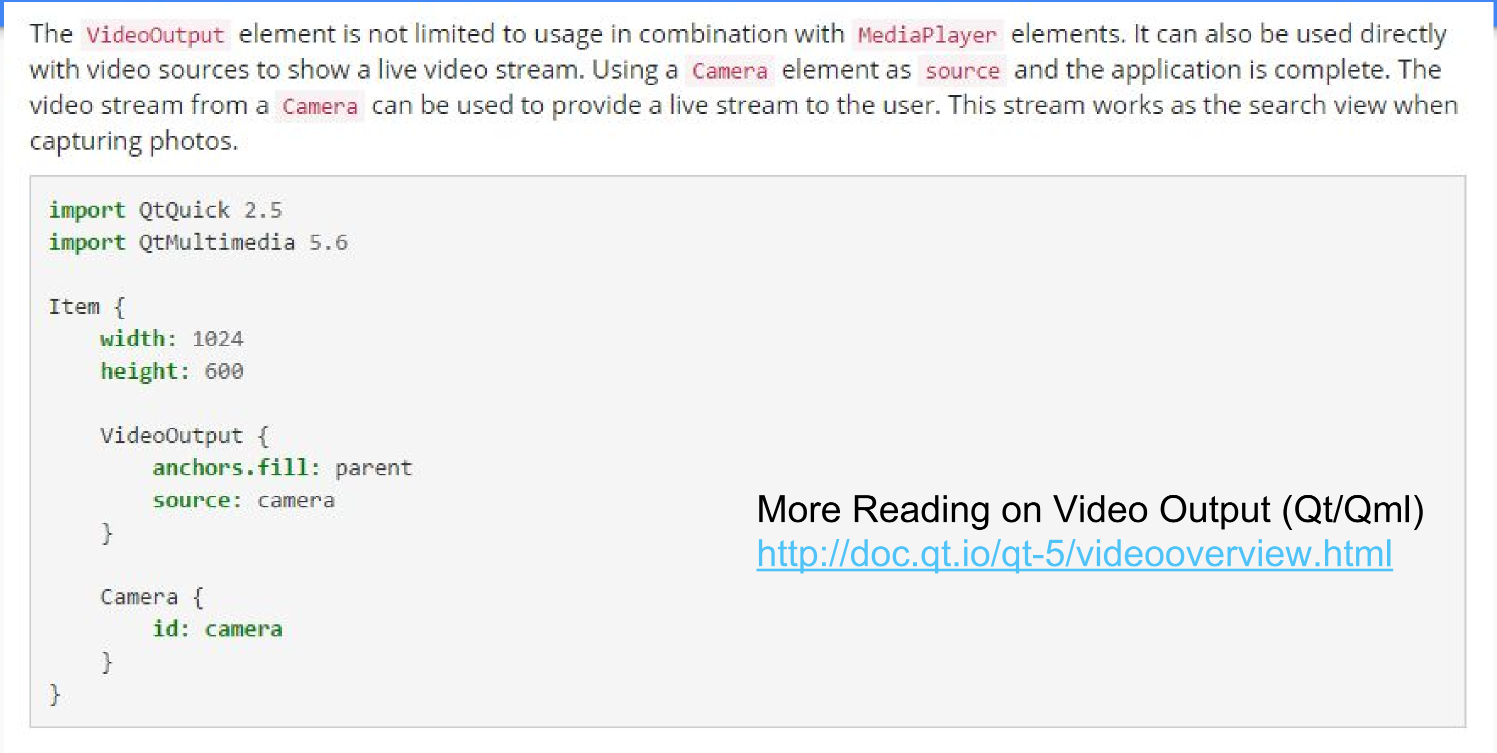
GUI Concepts - Sources

Any of these resources could provide a JPEG, or other data set representing a Pixel Buffer


Video (GPU)
Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)
JPEG
Source
GUI Application

In Application code - write to handle a specific type of data but allow for multiple sources (class hierarchy)

Camera and VideoOutput with Qml



The `VideoOutput` element is not limited to usage in combination with `MediaPlayer` elements. It can also be used directly with video sources to show a live video stream. Using a `Camera` element as `source` and the application is complete. The video stream from a `Camera` can be used to provide a live stream to the user. This stream works as the search view when capturing photos.

```
import QtQuick 2.5
import QtMultimedia 5.6

Item {
    width: 1024
    height: 600

    VideoOutput {
        anchors.fill: parent
        source: camera
    }

    Camera {
        id: camera
    }
}
```

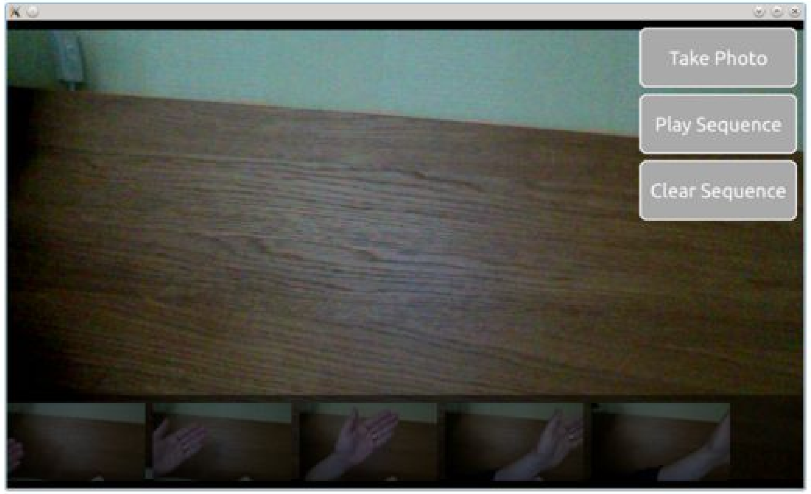
More Reading on Video Output (Qt/Qml)
http://doc.qt.io/qt-5/videooverview.html

Camera

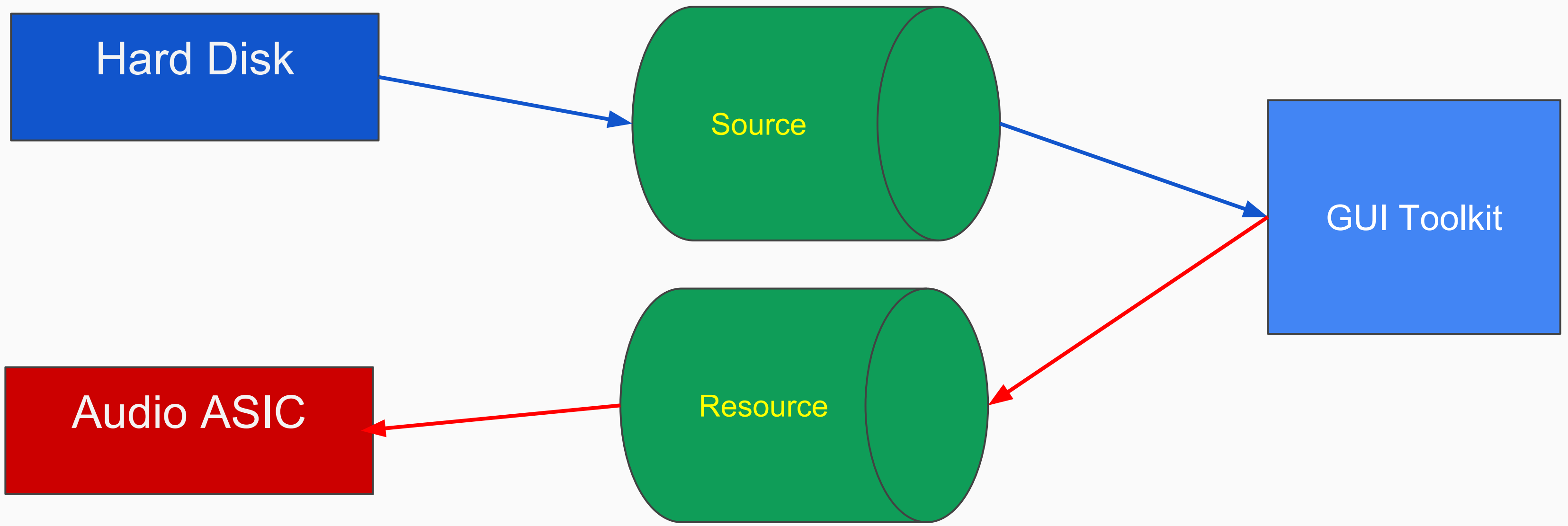
- Still Pictures
- Video Preview
- Video Record

One of the key features of the `Camera` element is that is can be used to take pictures. We will use this in a simple stop-motion application. In it, you will learn how to show a viewfinder, snap photos and to keep track of the pictures taken.

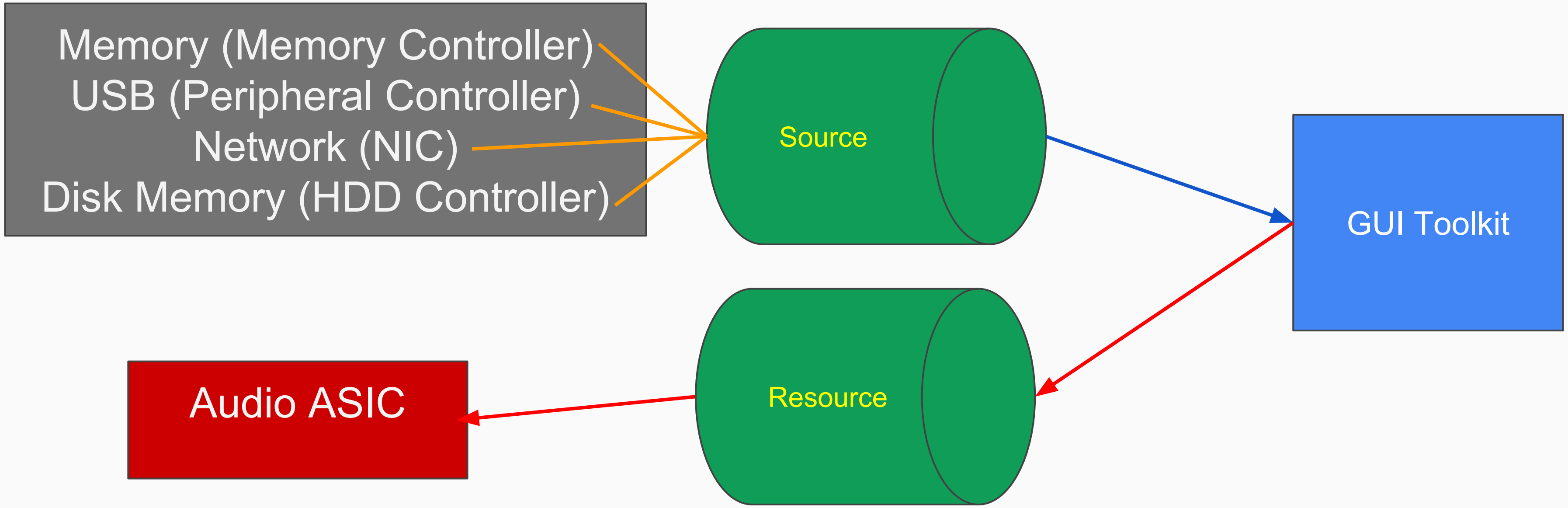
The user interface is shown below. It consists of three major parts. In the background, you will find the viewfinder, to the right, a column of buttons and at the bottom, a list of images taken. The idea is to take a series of photos, then click the Play Sequence button. This will play the images back, creating a simple stop-motion film.



GUI Concepts - Sound (Load Song)

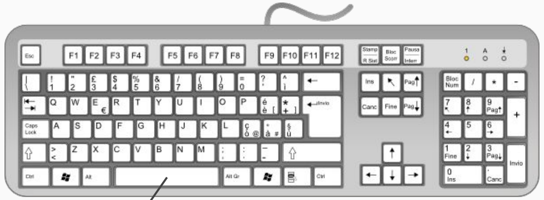


GUI Concepts - Sound (Load Song)


Memory (Memory Controller)
USB (Peripheral Controller)
Network (NIC)
Disk Memory (HDD Controller)
Source
GUI Toolkit
Resource
Audio ASIC

GUI Concepts - Sound (Latency Example)

Events are complex - timing and synchronization is important



Space Key Pressed (Fire)

GUI Concepts - Sound (Latency Example)

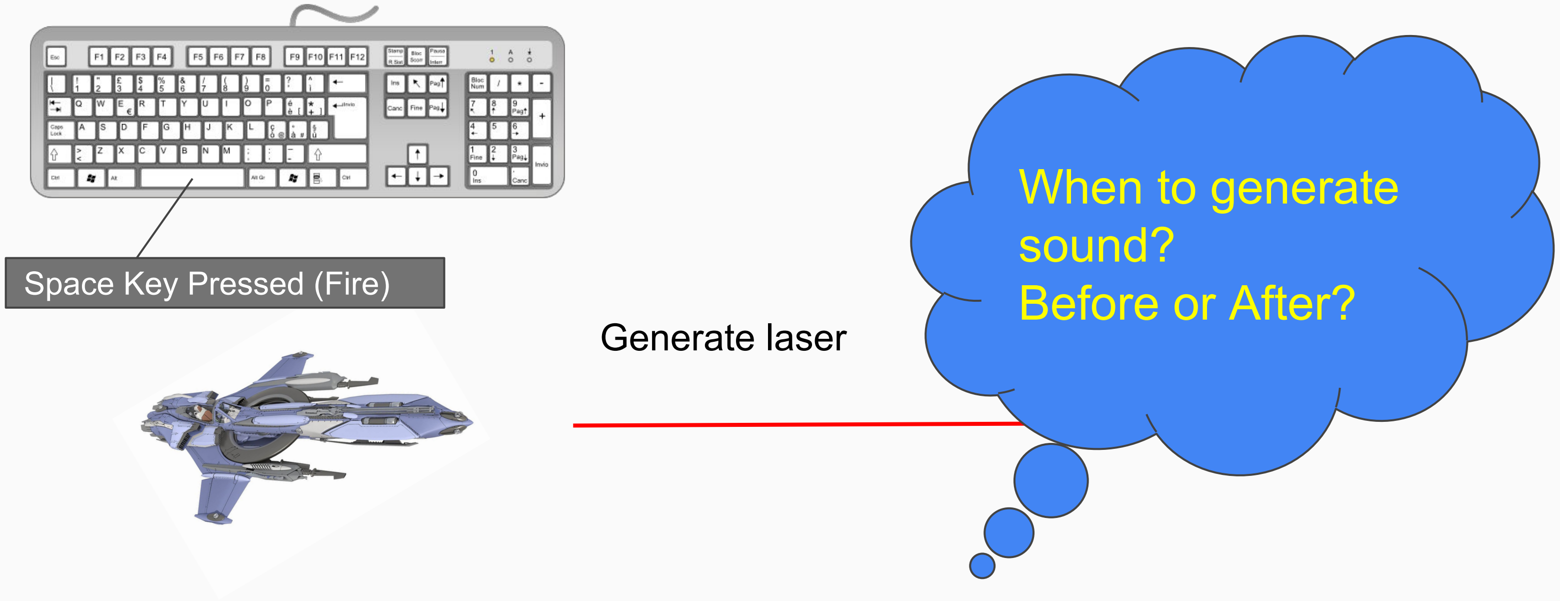
Events are complex - timing and synchronization is important

Space Key Pressed (Fire)

Generate laser

GUI Concepts - Sound (Latency Example)

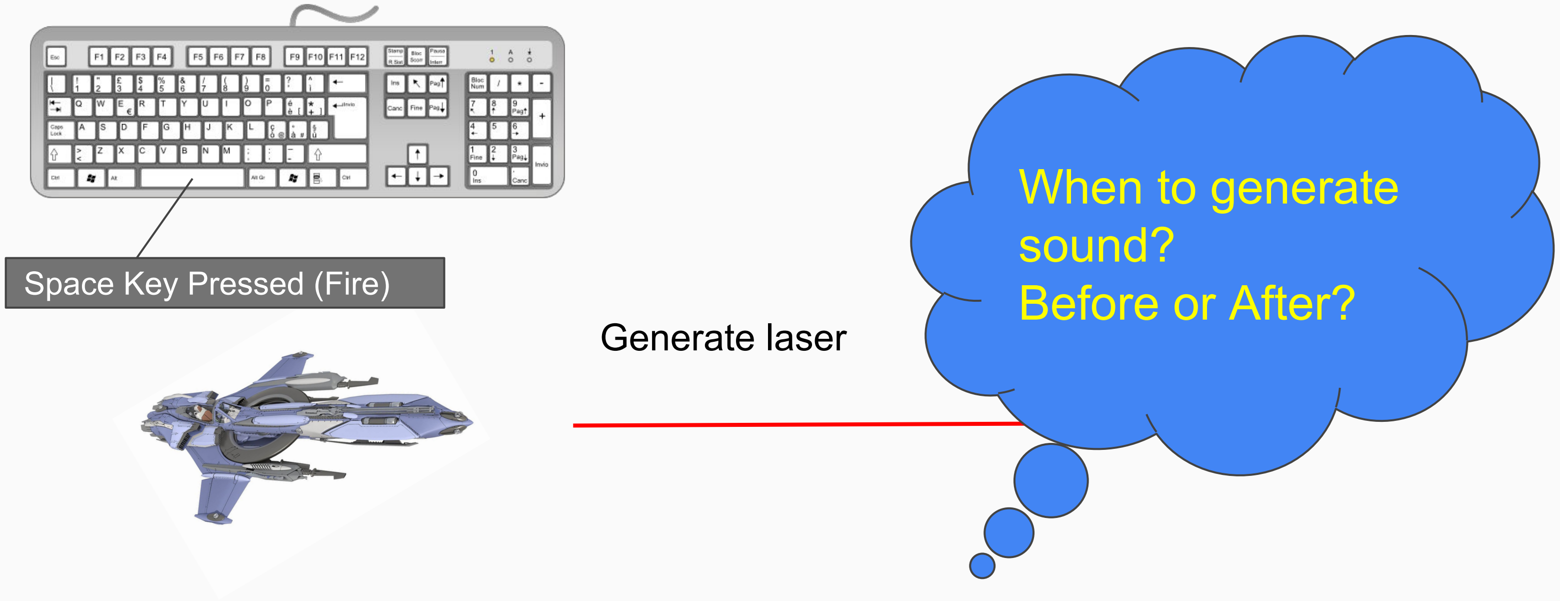
Events are complex - timing and synchronization is important



When to generate sound?
Before or After?

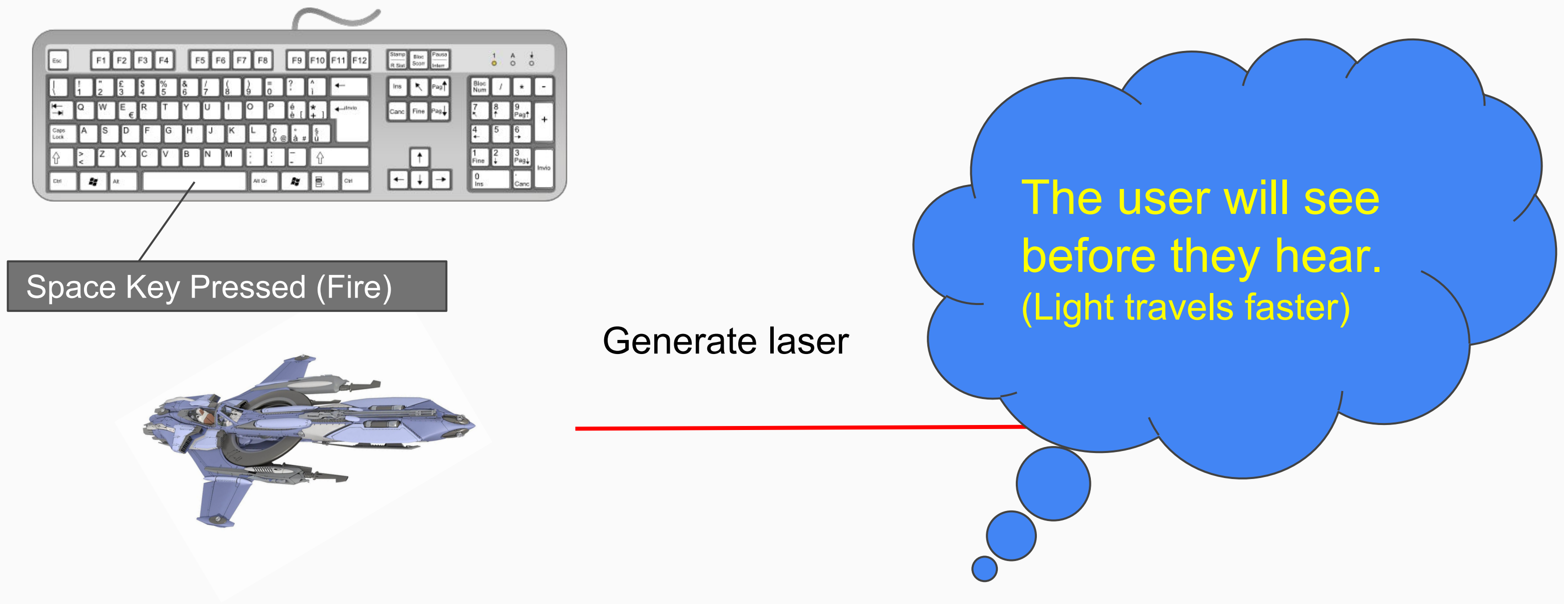
GUI Concepts - Sound (Latency Example)

Events are complex - timing and synchronization is important



GUI Concepts - Sound (Latency Example)

Events are complex - timing and synchronization is important



GUI Concepts - Sound (Latency Example)

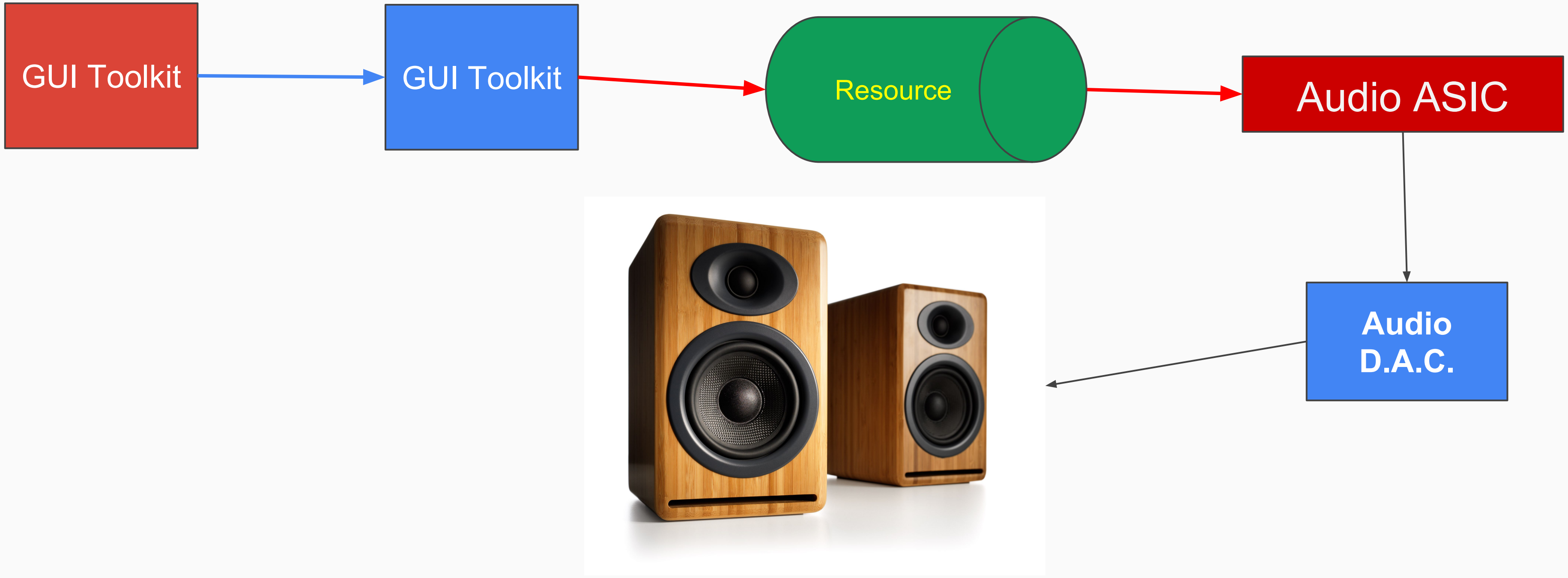
Events are complex - timing and synchronization is important

Space Key Pressed (Fire)

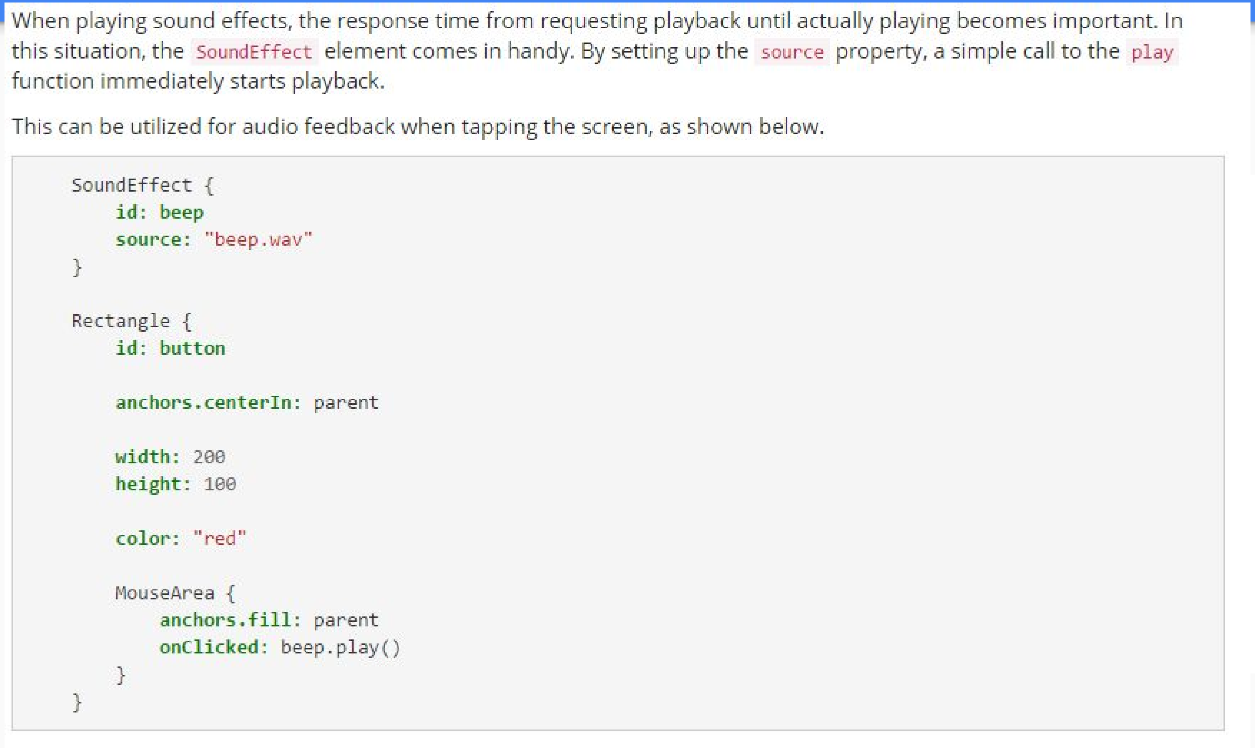
So often the sound needs to start towards the user's senses before the visual

Generate Sound, Generate laser (visual)

GUI Concepts - Sound (Latency Example)



Sound Effect



When playing sound effects, the response time from requesting playback until actually playing becomes important. In this situation, the `SoundEffect` element comes in handy. By setting up the `source` property, a simple call to the `play` function immediately starts playback.

This can be utilized for audio feedback when tapping the screen, as shown below.

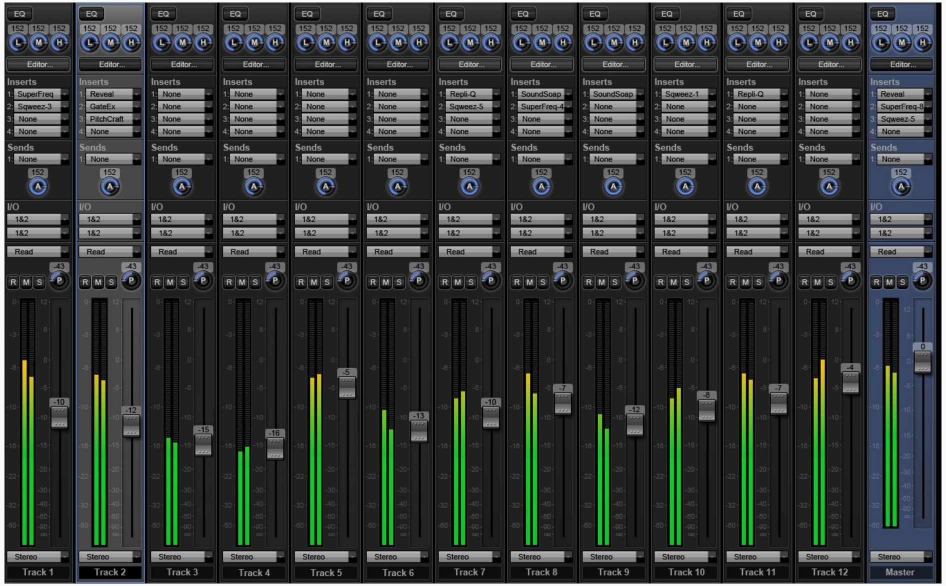
```
SoundEffect {
    id: beep
    source: "beep.wav"
}

Rectangle {
    id: button

    anchors.centerIn: parent

    width: 200
    height: 100

    color: "red"

    MouseArea {
        anchors.fill: parent
        onClicked: beep.play()
    }
}
```

Produce a low latency sound

QML adds another layer of latency to sound calls. Complex sound operations perform best in C++ with thought put into channel allocation

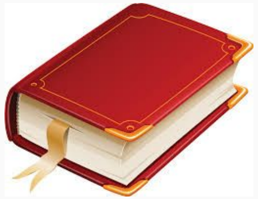
Multiple Channels

Software APIs allow for “Digital Mixing” - using multiple channels



More Reading (Qt/QML):
http://doc.qt.io/qt-5/audiooverview.html

Reading This Week:



QML Book (Chapter 10)
http://qmlbook.github.io/en/ch10/index.html

Qt Multimedia
http://doc.qt.io/qt-5/multimediaoverview.html