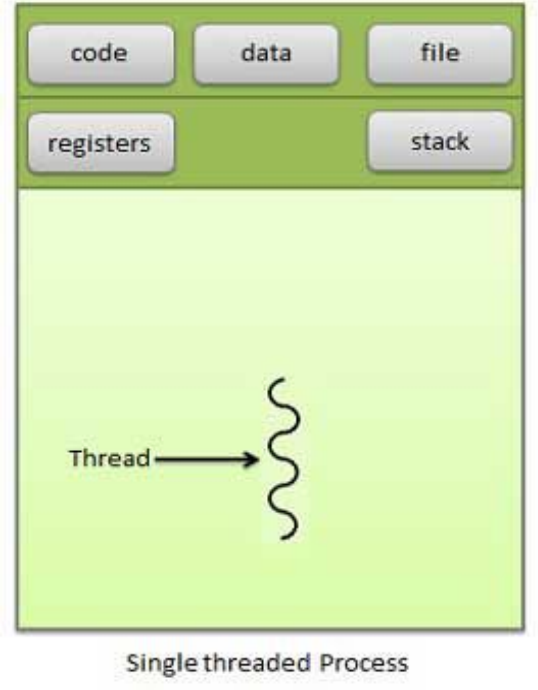
Advanced GUI Techniques

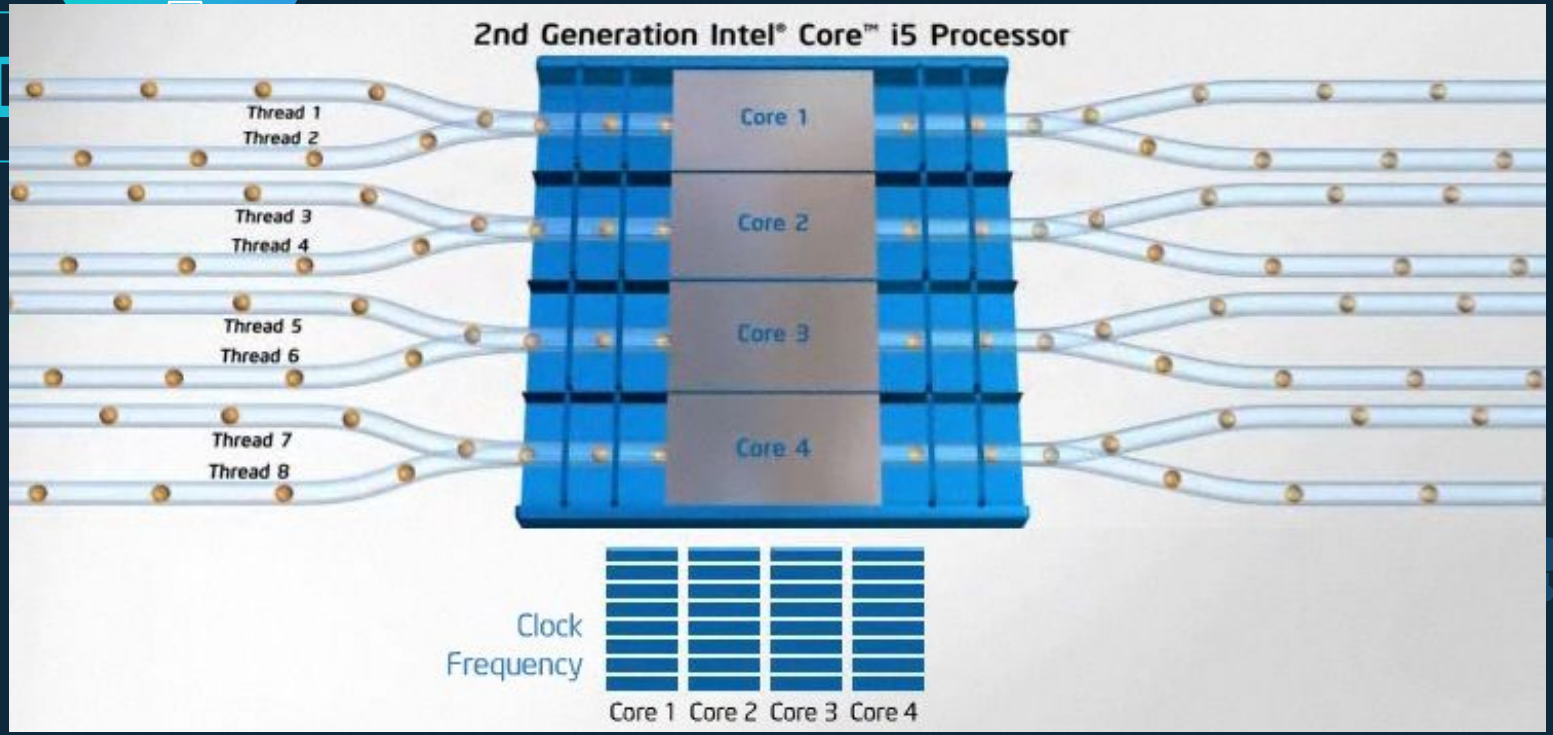
Concepts of Threading

What is a thread-



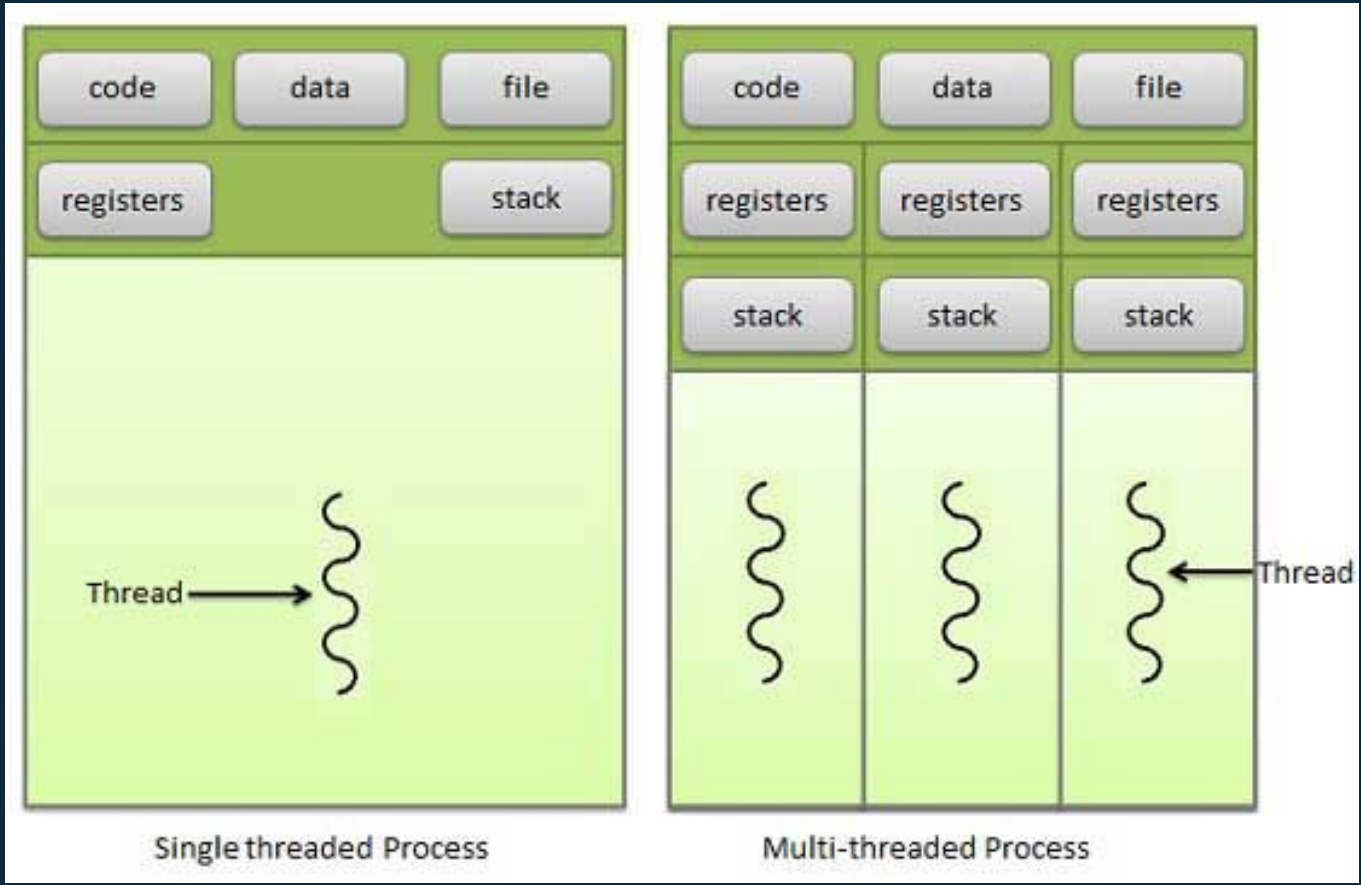
Threads live inside a process. Multiprocessors can also handle multiple threads

Basic idea – Concurrency



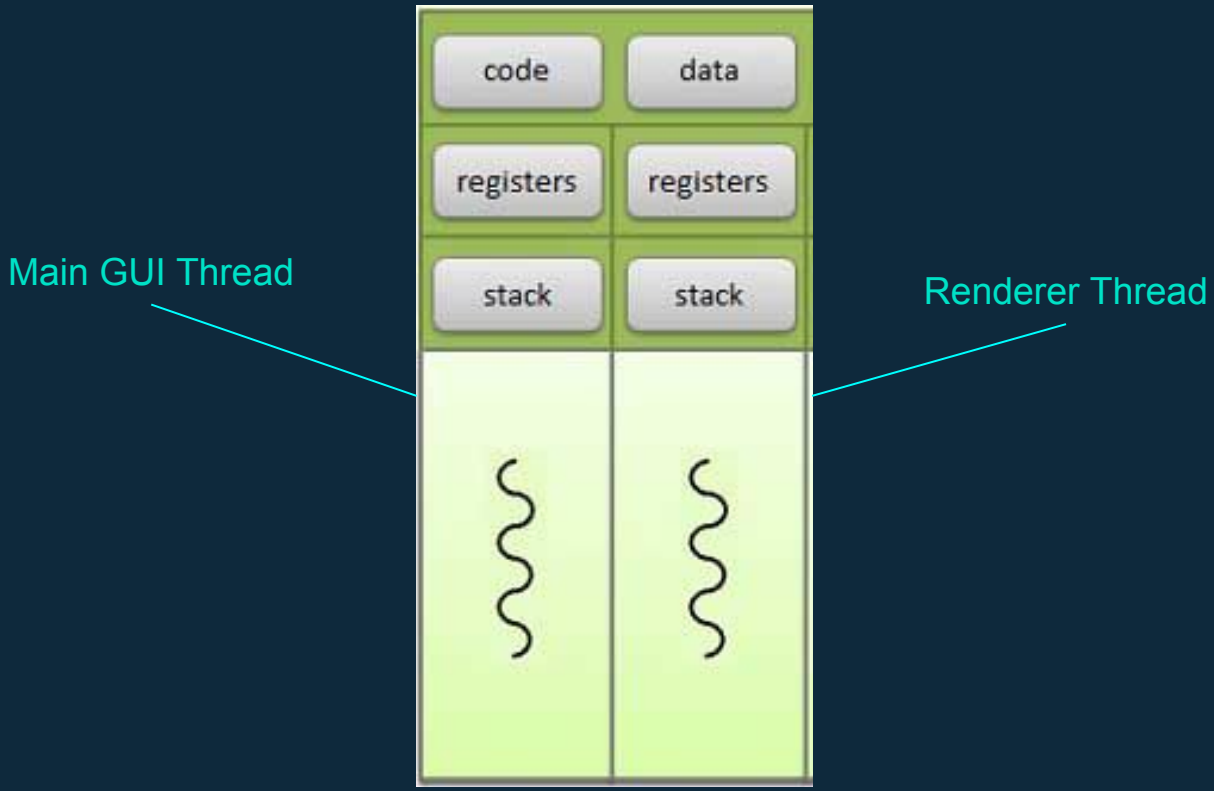
We utilize the hardware more efficiently and get soft/real-time concurrency

Single vs Multi



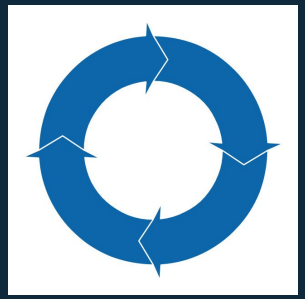
Notice the different registers and stacks between threads.

Modern Toolkits are threaded

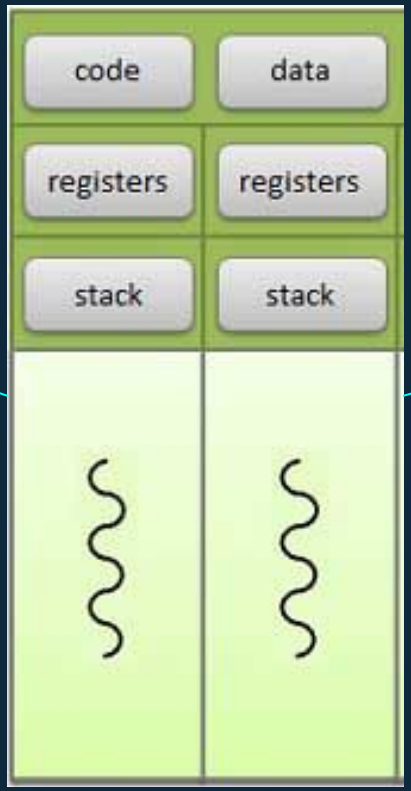


Modern Toolkits are threaded

Main GUI Thread



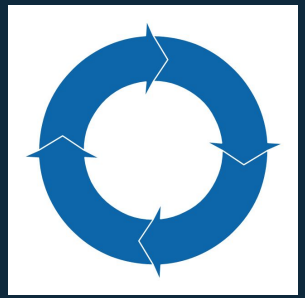
Manage Events (I/O)



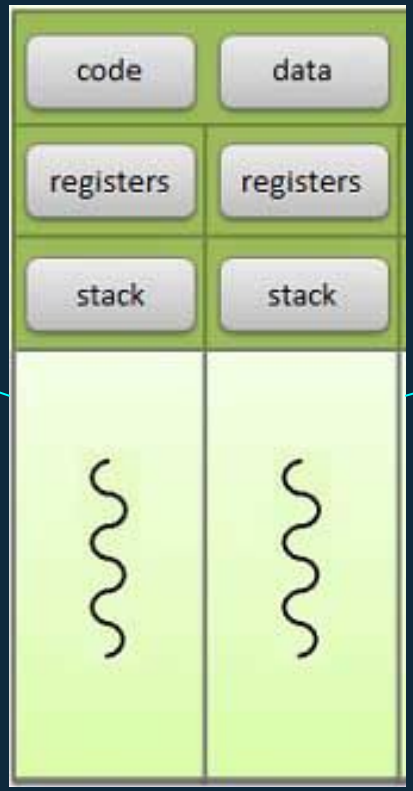
Renderer Thread

Modern Toolkits are threaded

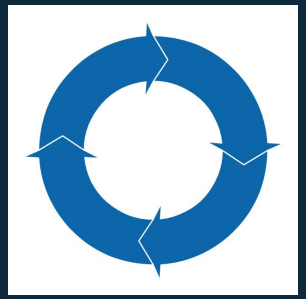
Main GUI Thread



Manage Events (I/O)



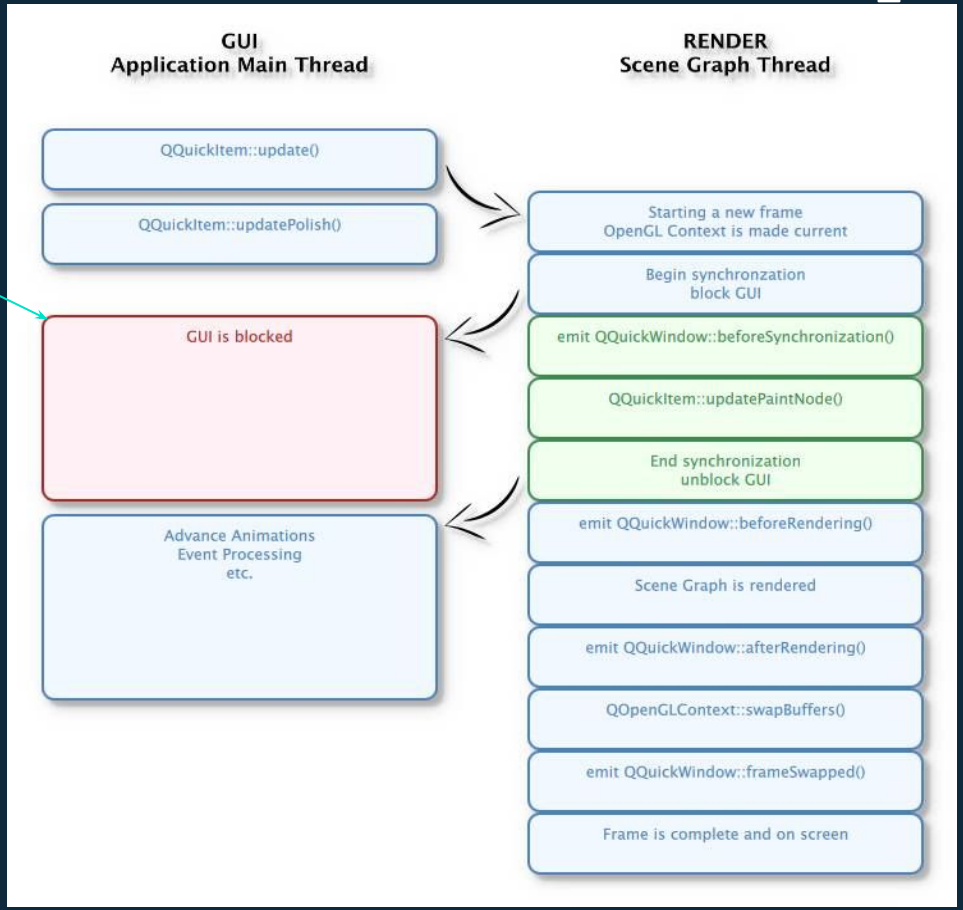
Renderer Thread



Drawing and Graphical SYNCHRONIZATION

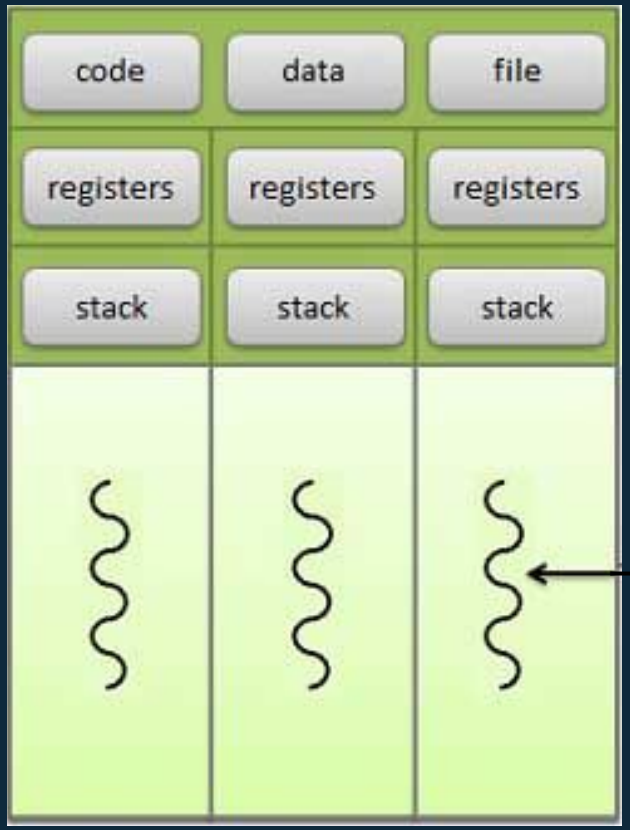
Qt/QML Relationship

Notice the blocking

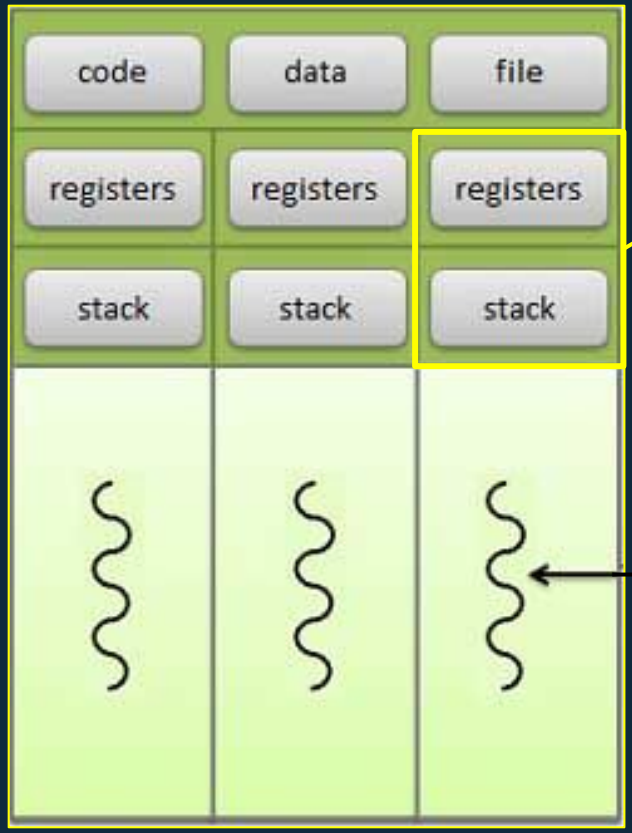


It is like a dance

Adding a worker thread to the mix



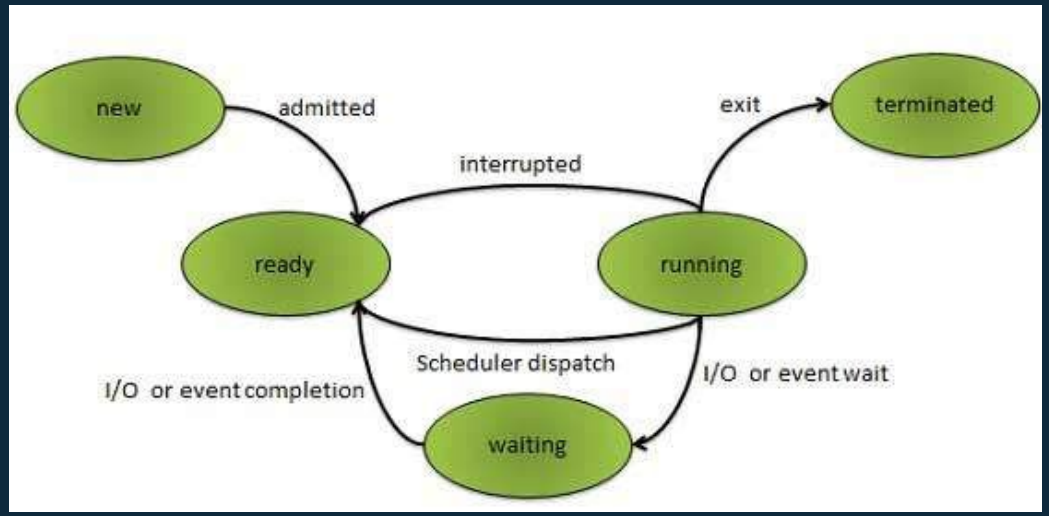
Small workloads might not warrant the overhead



Notice the Overhead

Worker Thread

For each thread added,
the system gains overhead



- Thread created
- Data copied
- Thread switch

Remember your application is one of *Many*

What does this mean for gui?

Proper Use of Thread

```
void  MyThread:run()
{
    initializeVariables();

    // begin long running process
    // or event system
    ...
   return 0;
}
```

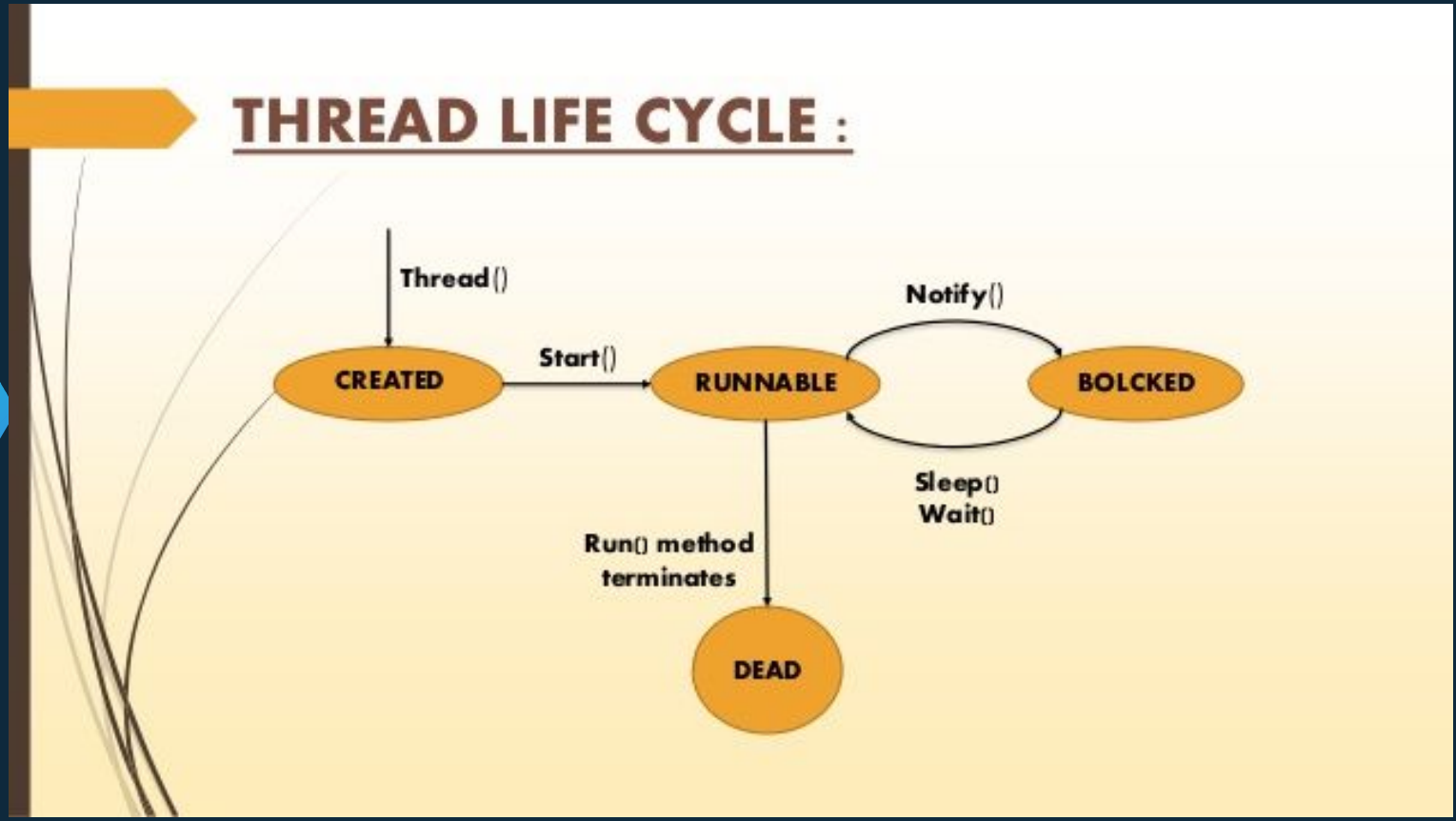
Improper Use of Thread

```
void  MyThread:run()
{
   cout << “Hello World”;
   return 0;
}
```

The opposite of a *handler*, threads should process **BIG** tasks.

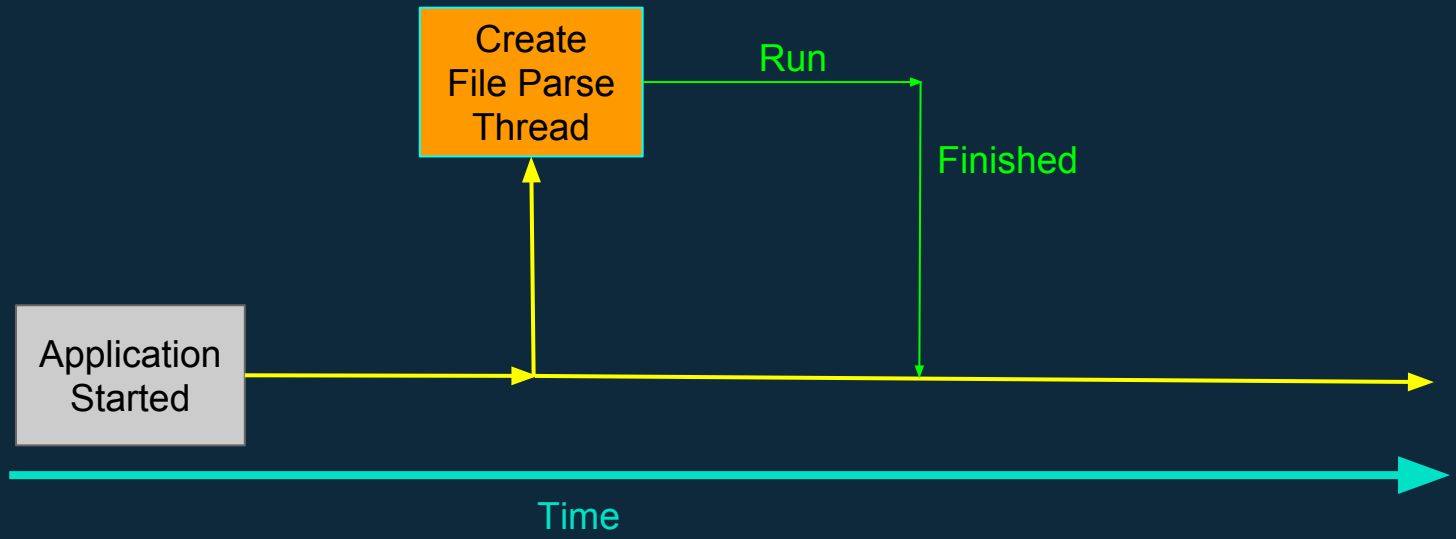
http://doc.qt.io/qt-5/thread-basics.html

Important concept – Lifecycle



Threads have a lifecycle, with corresponding states.

Important concept – *Lifetime*

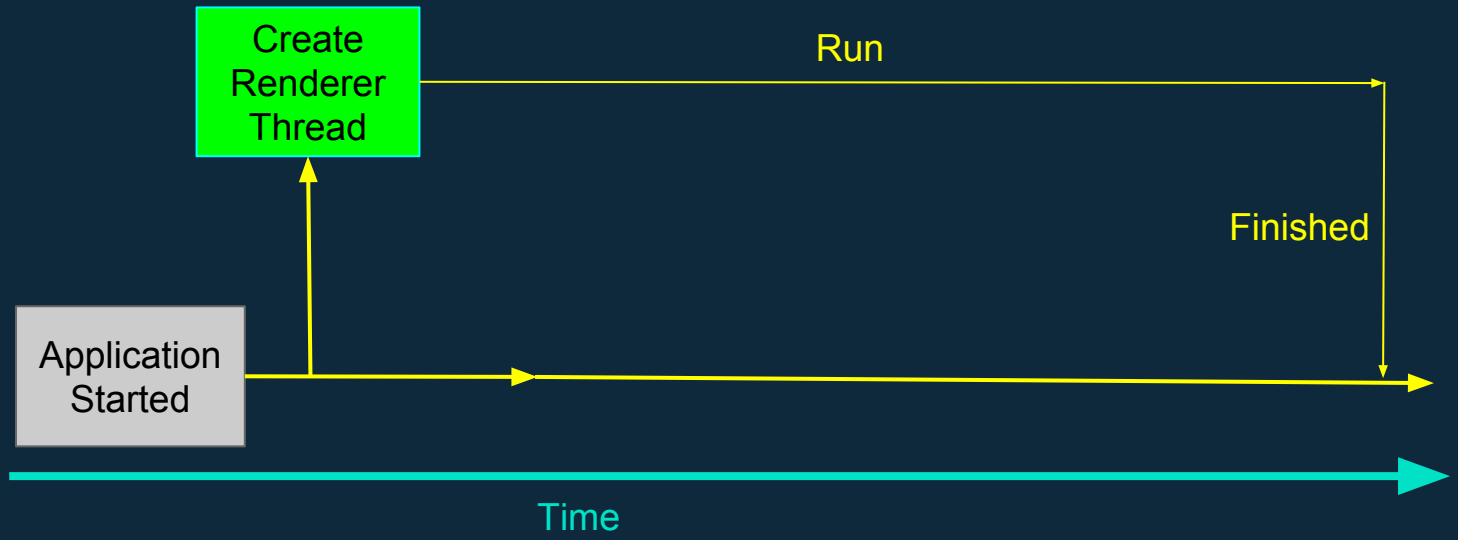


One-time-run or threads with short lifetimes - are created, perform their blocking operations, and then are terminated. (Thread pool, and “runnable” thread objects)

(Short)

http://doc.qt.io/qt-5/qthreadpool.html

Important concept – *Lifetime*



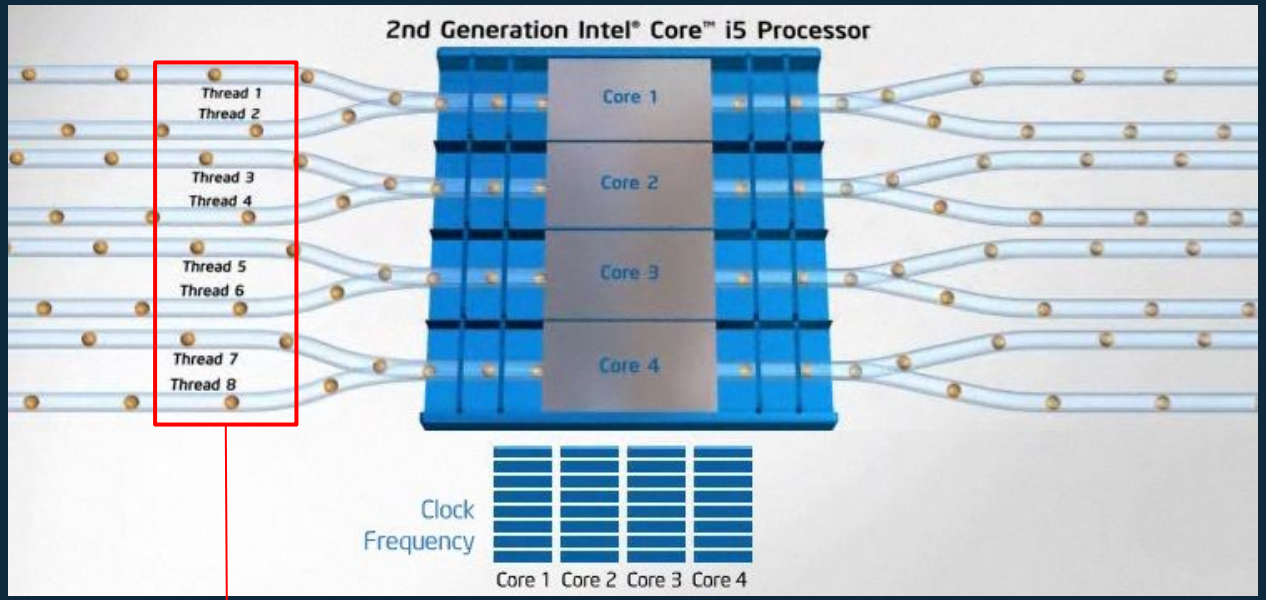
Some thread lifetimes are coupled to the main thread. In multi-threaded GUI toolkits, threads are generated implicitly to handle rendering and other tasks without effort of the developer.

(Long)

http://doc.qt.io/qt-5/qthread.html

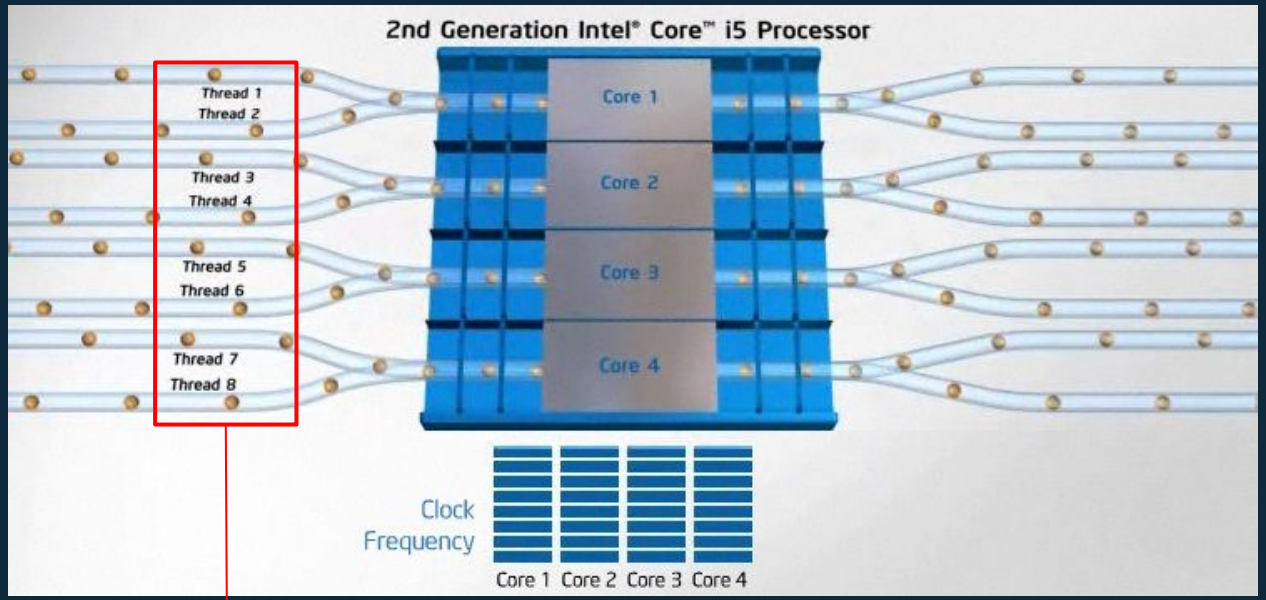
So More Threads the better?

So More Threads the better?



Threads are a resource - Always respect system resources!

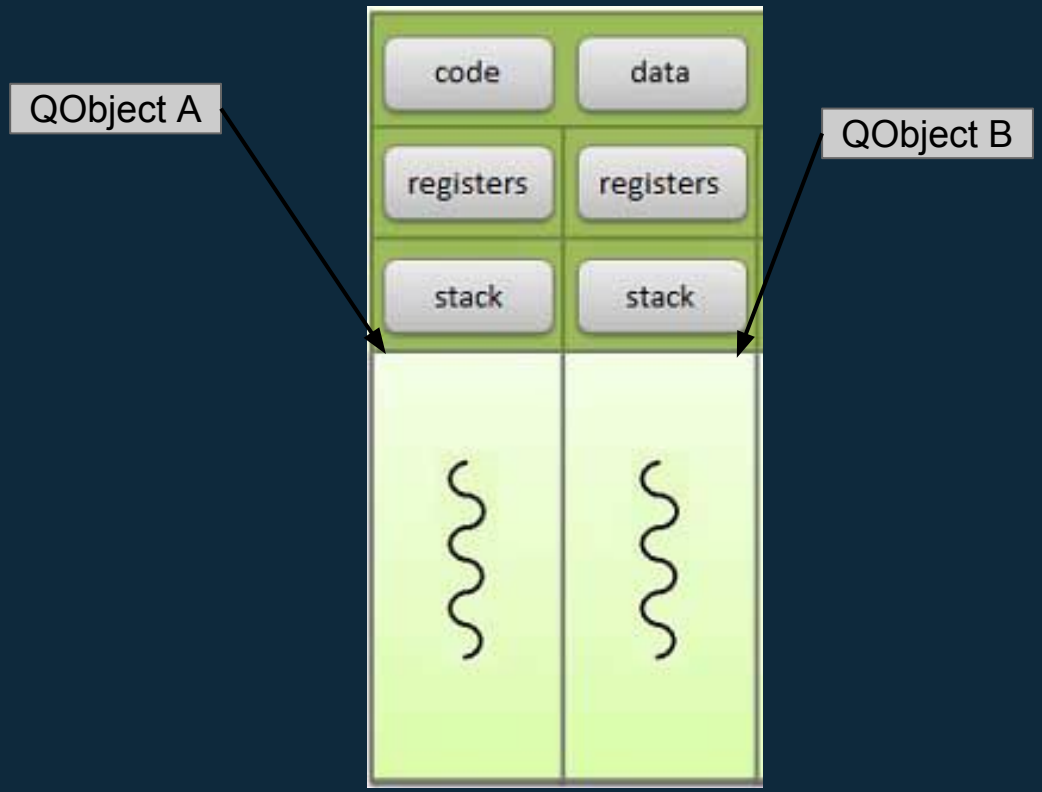
What is the right number of threads?



Threads are a resource - Always respect system resources!

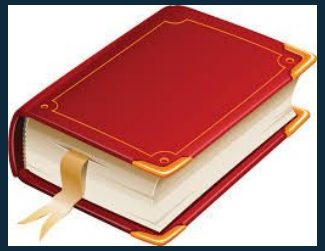
Developers must make the effort to choose an appropriate thread count for their target hardware. The toolkit APIs can help with this.

Object Thread Affinity



Objects live in the stack they were created in

Readings This Weekend (week 7)



Common Qt Classes (Ch 15)
http://qmlbook.github.io/en/ch15/index.html#common-qt-classes

WorkerScript
http://doc.qt.io/qt-5/qtquick-threading-example.html

QThreads
http://doc.qt.io/qt-5/qthread.html

Qt/Qml Signals
http://doc.qt.io/qt-5/qtqml-syntax-signals.html

QRunnable
http://doc.qt.io/qt-5/qrunnable.html

Live Code Demo - WorkerScript Sort

WorkerScript

Lecture 1

WorkerScript contains an example of using a WorkerScript to offload expensive calculations into another thread. This keeps the UI from being blocked. This example calculates numbers in Pascal's Triangle, and not in a very optimal way, so it will often take several seconds to complete the calculation. By doing this in a WorkerScript in another thread, the UI is not blocked during this time.

When the UI needs another value, a request is sent to the WorkerScript:

Detailed Description

Lecture 2

The QThread class provides a platform-independent way to manage threads.

A QThread object manages one thread of control within the program. QThreads begin executing in run(). By default, run() starts the event loop by calling exec() and runs a Qt event loop inside the thread.

You can use worker objects by moving them to the thread using QObject::moveToThread().