Multithreading is Hard

“New programmers are drawn to multithreading like moths to flame, with similar results.”

- Danny Thorpe
Solution

• Extract all hard parts into a boilerplate code.
• Test it. Test again. Test repeatedly.
• Reuse as much as possible.
• Test again. Don’t stop testing.

– or –

• Use existing library.
Bugs Away!

“The code that you don’t write, contains no bugs.”

- Primož Gabrijelčič
Adapt the Algorithm to the Pattern

• don’t write the code for your algorithm
• decompose the algorithm into the patterns

• when everything fails, go low-level
  • tasks first [CodeRage 9: Parallel Programming Library]
  • threads last
Frameworks

- **PPL**
  - Parallel Programming Library
  - XE7+, all platforms, RTL license
  - patterns: For, Future, Join

- **OTL**
  - OmniThreadLibrary
  - 2009+ (patterns), 2007+ (tasks), Windows (VCL/console/service) only (but working on that), OpenBSD license
  - patterns: Async[/Await], Background worker, For, Fork/Join, Future, Join, Map, Parallel task, Pipeline
Patterns

- Async/Await
  - Fire asynchronous tasks
- Future
  - Execute long calculation in background
    https://en.wikipedia.org/wiki/Futures_and_promises
- For
  - Use all of available CPUs when processing large data
- Map
  - Converting data in parallel
Async/Await

Async(code1).
Await(code2)
code 1
code 2
Future
For

For(first, last, code)

code

code

code
Map

Map(source, mapping)

source

mapping

mapping

mapping

result
Important Facts We Learned Today

• Don’t write boilerplate code – use patterns

• Be careful when accessing shared data

• Never access the GUI from a background thread!
Q&A

Primož Gabrijelčič

blog  www.thedelphigeeek
info   primoz.gabrijelcic.org
email  primoz@gabrijelcic.org
twitter @thedelphigeeek
skype  gabr42