The Projucer

Live C++ with Clang and the LLVM JIT engine

Julian Storer

www.juce.com

Overview

- Background: JUCE, the Introjucer
- Live coding: what, why?
- Projucer: Live demos!
- Dirty tricks required to make it work

The JUCE library

- Large "one-stop-shop" application toolkit
- Cross-platform: OSX/Win32/Linux/iOS/Android
- Several hundred classes: GUIs, graphics, audio, video, threading, openGL, XML/JSON, events, networking, high-level data structures, crypto, etc.
- Big in the audio software industry

JUCE Library Architecture

- Modular design
- About a dozen modules: core, gui, events, audio, video, openGL, plugins, etc
- Direct code inclusion for easier multi-platform projects
- Module format includes a manifest for automated management

The Introjucer

- Introjucer vs Projucer
- Solves the problem of keeping cross-platform projects in sync for multiple compiler/IDE targets
- Like an IDE without a compiler
- Generates project files for building in Visual Studio, Xcode, Linux (make), Android, MinGW
- Manages module linkage and dependencies
- Also has quick-start wizards, resource helpers, etc

Quick Introjucer demo...

Live Coding

- Bret Victor and Light-Table 2012 demos:
- interactive code feedback using javascript/clojure
- GUI editing where the code itself is the document
- Clang and the LLVM JIT

Projucer live demo!

Coding in a Live Style

Encourages good habits:

- Decoupled, robust classes
- Avoidance of state
- Classes with default constructors
- Simple GUI wrappers for working on non-GUI functions - TDD with instant feedback

The messy stuff...

- Integration from code editor → JIT engine → library
- Class database built from AST
- Compile units cached as LLVM modules
- Automatic PCH detection
- Threading and LLVM contexts
- Lazy JIT compiling
- Objective-C classes!

Future Possibilities

- Library of editors for wrapping audio / other types of code
- Custom GUI editors
- Custom refactoring functions, compiled dynamically
- Integrated debugger
- emscripten

www.juce.com