



Buildyard

Multi-Project Build Tool

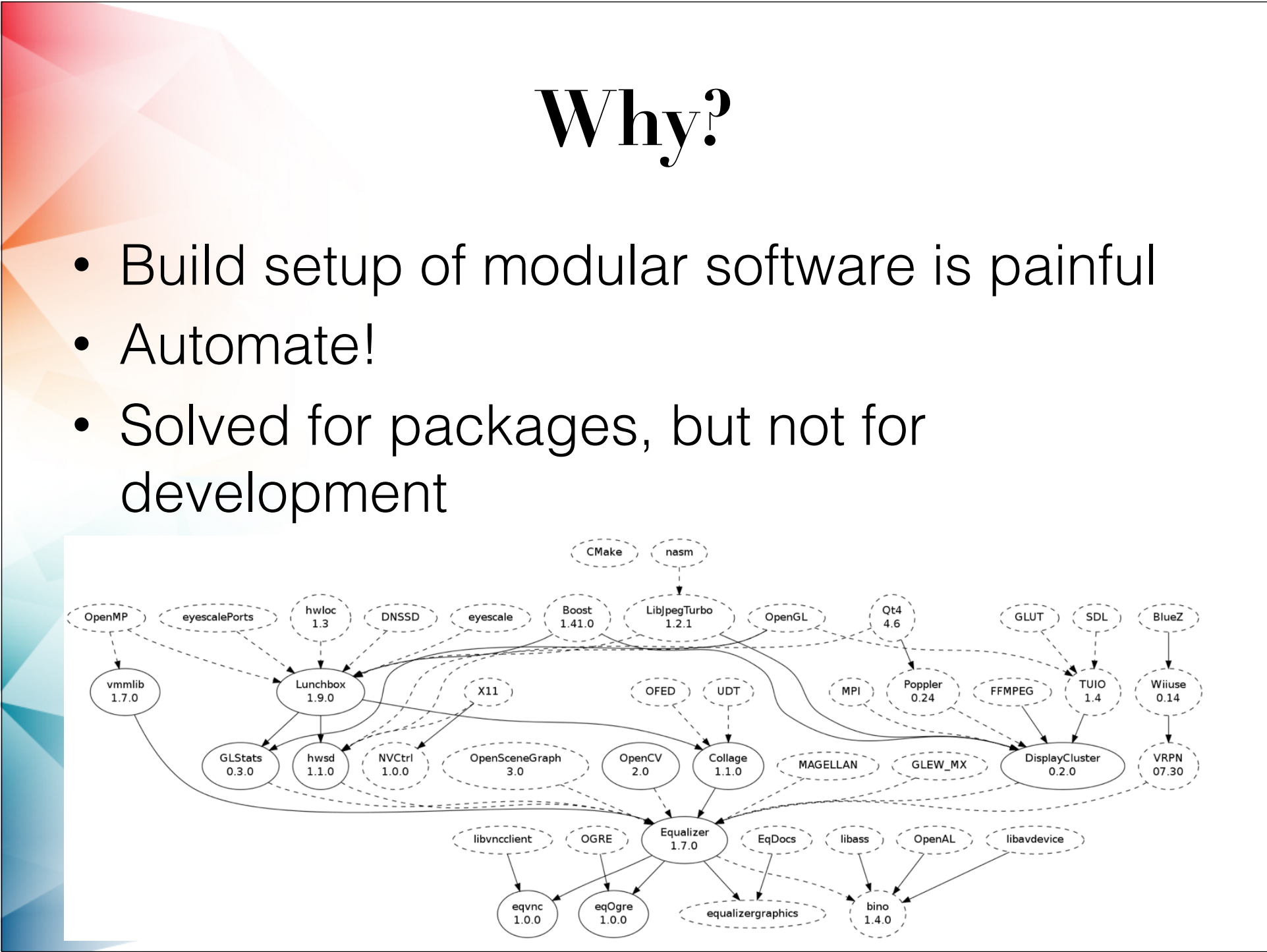


What is Buildyard?

- CMake-based build environment
- Facilitates build of multiple projects with dependencies
- Uses installed packages, svn or git source repositories
- Extensible through modular configurations

[illegible]

- # Why?
- Build setup of modular software is painful
 - Automate!
 - Solved for packages, but not for development
-
- The diagram illustrates a complex dependency graph for a software project. It shows a hierarchy of dependencies, with some packages depending on others. Solid arrows represent direct dependencies, while dashed arrows represent indirect or optional dependencies. The graph is organized into layers, with core dependencies at the top and more specific application components at the bottom.
- Key packages and their dependencies include:
- Core Dependencies (Top Layer):** CMake, nasm, Boost 1.41.0, LibjpegTurbo 1.2.1, OpenGL, Qt4 4.6, GLUT, SDL, BlueZ.
 - Intermediate Dependencies (Middle Layer):** OpenMP, eyescalePorts, hwloc 1.3, DNSSD, eyescale, Lunchbox 1.9.0, X11, OFED, UDT, MPI, Poppler 0.24, FFMPEG, TUJO 1.4, Wiusse 0.14, VRPN 07.30, DisplayCluster 0.2.0, MAGELLAN, GLEW_MX, Collage 1.1.0, OpenCV 2.0, OpenSceneGraph 3.0, NVCtrl 1.0.0, hwsd 1.1.0, GLStats 0.3.0, vmmllib 1.7.0.
 - Application Components (Bottom Layer):** libvncclient, OGRE, Equalizer 1.7.0, EqDocs, libass, OpenAL, libavdevice, eqvnc 1.0.0, eqOgre 1.0.0, equalizergraphics, bino 1.4.0.



How?

- Get Buildyard:

```
> git clone https://github.com/Eyescale/Buildyard.git
```
- Get a configuration folder in Buildyard:

```
> cd Buildyard  
> git clone https://github.com/BlueBrain/config.git config.bluebrain
```
- Configure and install system packages:

```
> make apt-get          # Ubuntu  
> make port-get         # Mac OS X, uses MacPorts
```
- Configure and build a project:

```
> make dash -j 9
```
- Work on a project:

```
> cd src/dash; vi ...; make -j 9
```

Give me more!

- Update Buildyard and configurations:
`> make update`
- Show the results of the last configuration:
`> make info`
- Reuse dependencies for project:
`include(FindPackages) in src/Project/CMakeLists.txt`
 - CMake/FindPackages.cmake is BY-generated
- Use an autoconf-based project:
`set(LIBJPEGTURBO_AUTOCONF ON) in config/LibJpegTurbo.cmake`

Give me more!

- Use a github user fork:

`set(DASH_USER_URL https://github.com/eile/dash.git)` in `config.local/`
`forks.cmake`

– remote "origin" points to eile, "root" to original

-

File System Layout

- Build/, Release/: Build directories where all generated files end up
- Build/[Project]: Per-project build directory
- Build/install: Installed project artefacts
- src/: All project sources
- src/[Project]: Per-project source directory

Show me the Magic!

- Uses standard ExternalProject.cmake
 - Chains projects together
 - Chains download->configure->build->install for each project
 - Make <project> takes long
 - Only to bootstrap
 - make in src/project does only build
- Configured using config.<org> folders

The Magic: config Folders

- One config folder:

```
> ls config.bluebrain/
```

```
dash.cmake codash.cmake depends.txt Livre.cmake README.md
```

- Depends.txt declares dependent configs:

```
config.eyescale https://github.com/Eyescale/config.git master
```

– Buildyard clones and parses these recursively

- Per-project configuration, e.g., dash:

```
set(DASH_PACKAGE_VERSION 1.1.0)
```

```
set(DASH_REPO_URL https://github.com/BlueBrain/dash.git)
```

```
set(DASH_DEPENDS bluebrain REQUIRED Lunchbox Boost)
```

```
set(DASH_BOOST_COMPONENTS serialization)
```

```
set(DASH_DEB_DEPENDS libboost-serialization-dev)
```

The Magic: project configs

- PACKAGE_VERSION: minimum needed
- REPO_URL: Source repository
- REPO_TAG: repo revision, default master
- DEPENDS: Dependencies
 - Can be system packages
 - Source is used as fallback, if configured
 - Missing REQUIRED dependencies will cause project to not be configured

The Magic: project configs

- BOOST_COMPONENTS: optional components for a dependency
 - Used for finding dependency
 - Forwarded to project source
- DEB_DEPENDS: used for apt-get target
 - Used to configure Travis CI
- PORT_DEPENDS: used for port-get target