Building a refactoring tool

With what?

- A compilation database for our project.
- Clang’s AST matchers.
- Clang’s libTooling.
- Clang’s libFormat.

Why?

- r194288 “If a linkonce_odr dtor/ctor is identical to another one, just rauw”.
- Implicit instantiations of class templates are linkonce_odr.
- Some of our code was using a declaration of templates, but it happened to link against an implicit instantiation.

Why?

- “using namespace common::base;” …
- common::base defines many things like Coord, Colour, etc…
- So does the rest of our code …
- Hence we get ambiguous name lookups in “void function(Colour c);”
- Need to remove the using directive.

Why?

- Can’t grep for it…
- Try removing it and fixing the compilation errors…
- An 8 hour day later, >700 files updated and the build is still broken…
- Turns out to be O(10,000) files to update!

Code! (v0)

Let’s:

- build a compilation database.
- write a program that uses it to open a C++ file and build up an AST in memory.

Code! (v1)

Let’s:

- try matching decls with AST matchers!
  - reference guide to AST matchers
- try issuing a replacement to edit the code!

Code! (v2)

Let’s:

- match DeclRefExpr’s.
- match TagType’s.
- try it on more than one file!

Code! (v3)

Let’s:

- make it not qualify names inside namespace common { namespace base { … } }

Code! (v4)

Let’s:

- make it not qualify already explicitly qualified names

Code! (v5)

Add debugging statements and refactor...

Code! (v6)

Let’s:

- skip matches inside templates

Code! (v7)

Let’s:

- respect explicit using declarations

Code! (v8)

Let’s:

- respect namespace aliases

Works well

- It’s easy to match pieces of the AST.
- Integration with build systems.
- Reformatting of changed lines for us!

Ideas for improvement

- Multiple times, we had to find the “best” name.
  - Why not make that an API?
- Making the AST follow the standard is critical.
  - We have to fix ElaboratedType. Any others?