Unit Testing Requires!

Bad Grammar Intentional,
Sentiment Real!

A Presentation by Cliff Green

Oct 3, 2024, GDG Albuquerque

(Containing pics of Balloon Fiesta, unit testing, gliders, cats, and other gratuitous subjects)

Cover photo by Cliff Green, Kolibri balloon piloted by John Wahl, Balloon Fiesta 2023

Theme is Framed Pastel Template by PresentationGO, https://www.presentationgo.com

Overview

- The case for unit testing
- A quick look at a related presentation on the C++ unit testing Catch library
- An open source unit test CI / CD demonstration
- Wrap up, final thoughts about unit testing
- Gratuitous photos that will pop up

Photo by Cliff Green, Moriarty Airport 2024

Why Unit Tests?

My wife Julie's first glider ride, Jan 1, 2024, Mark Hawkins piloting

Unit Testing Has Many Pros

- Subtle bugs are found quicker and easier with unit tests
- Unit test code is executed again and again; CI (continuous integration) means automated tests are run any time changes happen in a repository
- Designing test code from the perspective of the user of the code clarifies and improves the interfaces (classes, functions, etc)
- Design by contract is reinforced with unit test code - invariants and preconditions (and postconditions when needed) are explicitly tested

Automated Unit Tests

- Having automated unit tests enables regression testing, where breakages anywhere in the software package are immediately found
- Continuous integration (CI) means that any changes ready for integration immediately invoke builds and running regression tests
- GitHub provides CI (and other) services (demonstration coming up); there are similar online (and internal) software services available
- The cost of writing unit tests is paid back every time an automated test is run (versus manual testing)

White vs Black vs Grey Box Testing

- Black box - internal functionality or design is not known (or not considered), test inputs and outputs
- White box - internal design is known and tests are written to verify correctness
- Grey box - mixture of black and white box testing
- Unit testing is typically white box, while system testing is typically black and / or grey (but there are no hard rules)

Typical Other Ways of Testing Code

- Log / print messages scattered throughout code (clutters runtime environment!)
- Running code in a debugger (slow and laborious!)
- Manually running the executable or system and “watching what happens” (also laborious!)
- All of the above may still be needed at times (although I hardly ever use a debugger), but are slow and repetitive and rarely cover more than a few basic data cases or edge conditions

Manual Testing

- Run the system / app and observe … "Looks good, didn't crash, result appears correct, let's ship it!"
- Rarely does manual testing cover all needed use cases
- Manual testing many times doesn't cover edge or error cases, only "happy path"
- Manual testing takes time, is tedious, and requires doing the "same thing" over and over

Unit Testing Has Some Cons

- It takes time to design and write unit tests; production code takes longer until it is ready (but tends to have less bugs)
- Unit tests are only as good as the unit test design and code - sometimes the test code is buggy, not the production code
- Unit testing often times cannot cover all functionality, depending on the system and overall complexity
- Build scripts / CMake files have to be created for the unit tests, not just the production code

Assertions are the Heat Engine of Unit Testing

Photo by Cliff Green, mead by Tim Beck

A tasty bottle of mead waiting to be consumed (mead produced by GDG and ABQ SSC icon Tim)

Require and Check Assertions (Catch2 Lib)

- `REQUIRE(expression)` - fail and exit test case if evaluates false
- `CHECK(expression)` - same as `REQUIRE`, but continues execution
  - `REQUIRE_FALSE` and `CHECK_FALSE` simpler than negating expression
  - A thrown exception of any type is a failure
- `REQUIRE_NOTHROW(expression)` (and corresponding `CHECK_NOTHROW`) asserts no exception is thrown
- `REQUIRE_THROWS(expression)` (and corresponding `CHECK_THROWS`) expects an exception (of any type) is thrown
- `REQUIRE_THROWS_AS(expression, exception_type)` expects an exception of the specified type to be thrown (similar for `CHECK_THROWS_AS`)

Gratuitous Photo

My son snuggling with his cat Stoney

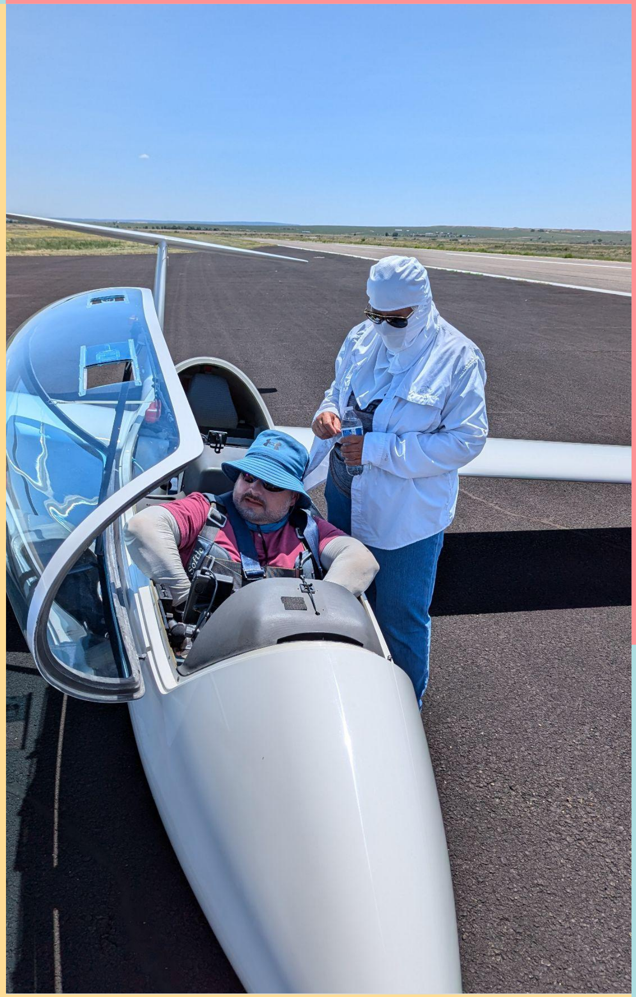
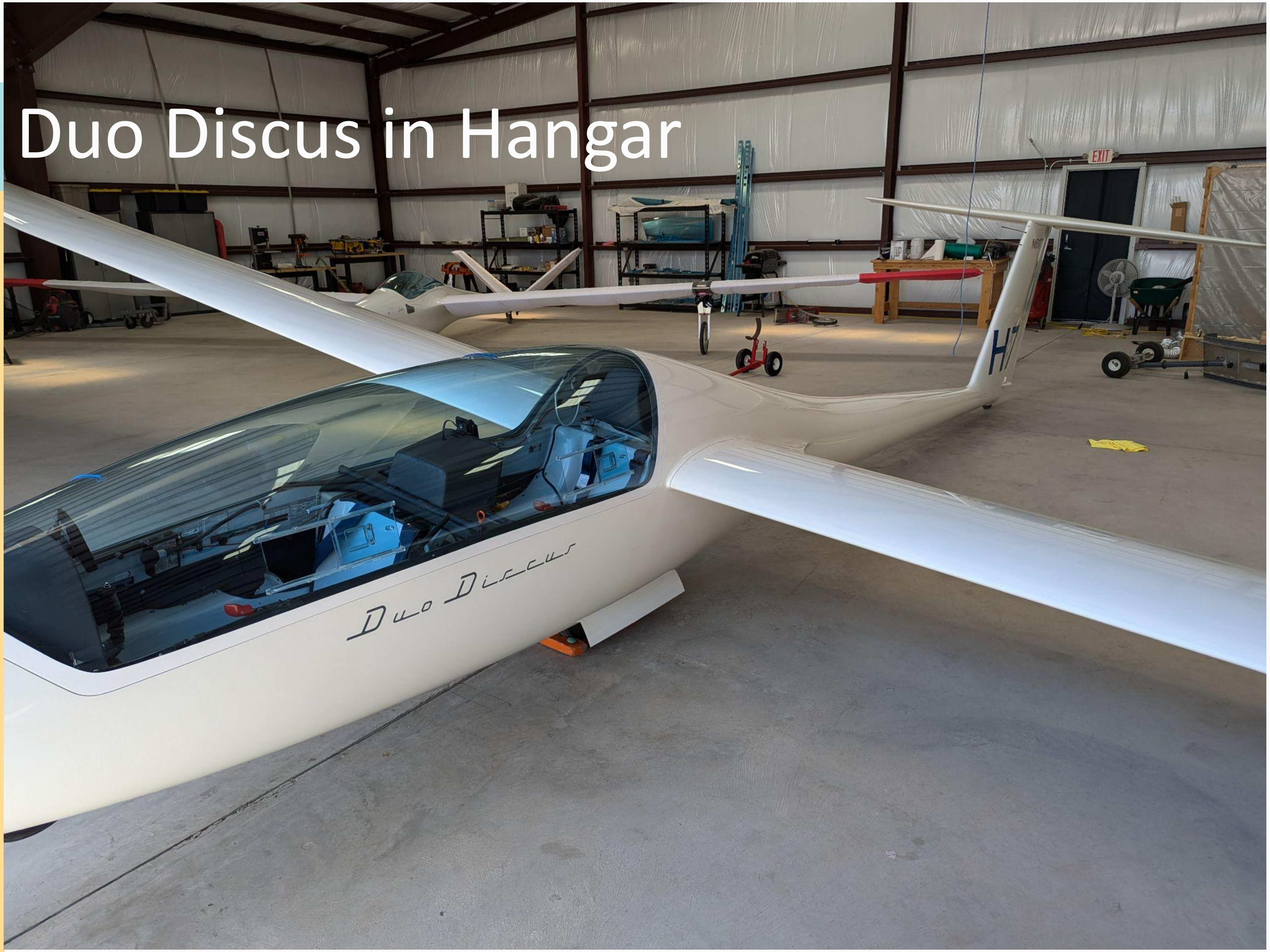


Photo by Cliff Green, Moriarty, NM

Couldn’t Resist One (Two?) More Gratuitous Photo(s)

Second glider (Duo Discus) for my aviation company (Merlin Aviation), Whitney (company co-owner and primary operator) piloting, Connie (other co-owner) assisting

Duo Discus in Hangar



Thank You!

- Two Essential Online Tools (mentioned in all of my C++ presentations):
  - CPP Reference page: https://en.cppreference.com/w/- have it ready, learn to read / use it
  - Compiler Explorer: https://godbolt.org/ - compile and run your code on multiple compilers, analyze the assembler output
- Question and discussion time, then demonstration (or maybe demonstration then questions and discussion)