CS354P

DR SARAH ABRAHAM

BUG TESTING/BUG TRACKING

BUG TESTING

- An extremely important aspect of software development
- We tend to skim over it in classes, because there are so many (more fun) “fundamentals” to cover
  - But bug fixes are arguably the majority of what you’ll be doing at a job
- The “80/20” rule
  - 80% of the time will be spent on 20% of the product
  - To rephrase: building out a working prototype takes 20% of the time, polishing the product takes 80% of the time

WHAT ARE BUGS IN GAMES?

- A little different from other industries...
- Bugs can:
  - Prevent player progress
  - Be implementations that do not match the design specification
  - Impact player experience and immersion
- Bugs can be related to performance, visuals, audio, player input, game state, physics, AI, etc...

DEBUGGING REVISITED...

- When unexpected behaviors occur, how do you debug?
- Do you read log files? If so, how?
- Do you use break points? If so, how?
- Do you use print statements? If so, how?
- What other tools do you use? How do you use them?

APPROACHING BUG FIXES

- It's impossible to debug effectively through guess work
- Approach bugs methodically and with an open mind
  - How many times have you heard a programmer say "that's impossible" as the impossible thing happens in front of them?
- If you feel tired or frustrated, go do something else/sleep on it if possible
  - Sometimes fixing things just takes time and a clear head -- attempting to rush the process will just make the process take longer and you will be more unhappy

WORKING WITH A HYPOTHESIS

1. Understand the bug
   - What exactly makes it a bug? What is expected versus actual behavior? What other behaviors may or may not be related to the bug itself?
2. Understand the bug's reproducibility
   - When does the bug occur? What sequence of events causes it versus doesn't cause it? Take time to really understand when and how the bug manifests
3. Form a hypothesis for why the bug is occurring
   - Pick one thing at a time to consider! Choose the "easy to fix or test" hypotheses first and eliminate the "most likely" issues as soon as possible too
4. Isolate the variables as much as possible when testing your hypothesis
   - You must be methodical in how you approach this. Trying to test quickly is only going to obfuscate the issue and miss the problems

OTHER FORMS OF TESTING IN UNREAL?

UNREAL AUTOMATION TESTING

- Automation testing, broadly speaking, is the process of running tests automatically to check against an expected result
- In UE5, Automation Testing is the lowest level of this automated process
  - Primarily used in-engine and exists outside of UObject ecosystem
  - Not visible to Blueprints or Reflection System
  - Can create Simple and Complex tests by deriving from `FAutomationTestBase`

AUTOMATION MACROS AND FUNCTIONS

- `IMPLEMENT_SIMPLE_AUTOMATION_TEST` and `IMPLEMENT_COMPLEX_AUTOMATION_TEST`
- Both macros have three parameters:
  - TClass (name of test class)
  - PrettyName (name that appears in UE)
  - TFlags (flag values for testing)
- Must include RunTest function for both types of testing and GetTests function for complex testing

SIMPLE TESTS

- Used for single, atomic tests (i.e. unit tests)
- Unit tests examine individual sections of code in isolation
- Ideally considers smallest, testable part of the API and verifies they are working
- No consideration for larger functionality

USING SIMPLE TESTS

- Answers questions related to basic functionality
  - Does the map load?
  - Do actors spawn with the correct parameters?
  - Is the UI updating as expected?
  - Do state-changing functions change the state correctly?

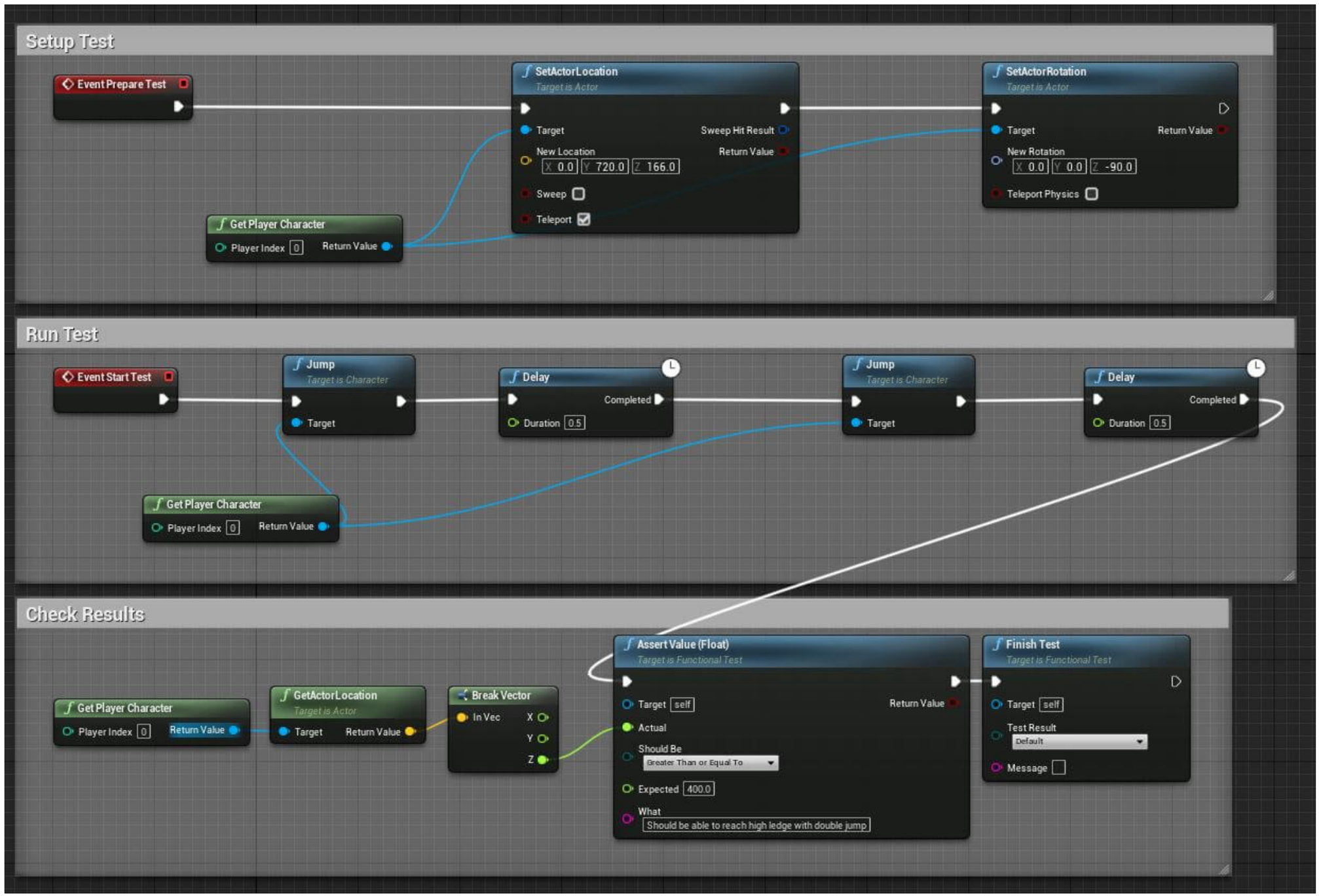
COMPLEX TESTS

- Used for testing the same code using multiple inputs
- Can check things like do all maps load or do all Blueprints compile
- Used for content stress tests to avoid system crashes and other issues arising from performance or memory considerations

FUNCTIONAL TESTING

- Tests larger features such as mechanics or actions and take system interactions into account
  - Takes the perspective of the user and “expected” behavior
  - Harder to reason about but better for considering the user’s experience
- Possible to implement these systems using Blueprint and the Functional Test plugin

EXAMPLE: TESTING DOUBLE JUMP



(https://kobiton.com/automation-testing/an-introduction-to-automated-testing-for-an-unreal-engine-project/)

AUTOMATION DRIVER

- Simulates player input to test functionality within the game
  - Currently limited to keyboard and mouse support
- Can be used on scene actors in addition to UI
- Uses dependency injection to pass dummy values into the input system
- Fairly advanced system to work with but can provide a way to build out both UI testing and bot testing

ADDITIONAL AUTOMATION SUITE FEATURES

- FBX Test Builder checks modifications to imported .fbx assets against expected results
  - e.g. unit testing for art assets
- Screenshot Comparison Tool provides an interface for diff comparisons between images based on build
  - Requires human eyes but simplifies the debugging of graphical issues

SMOKE TESTING

- Process of determining whether or not a build is stable based a simple set of tests
  - Most baseline level of expected functionality to continue working
- Unreal Smoke Tests are run *every* time the Editor, game, or commandlet starts
  - Only tests that run in under 1 sec should be marked as Smoke

TESTING LARGER INTERACTIONS

- Despite many levels of automation, games still rely heavily on manual testing to catch unexpected interactions
  - Tester time is cheap compared to programmer time (i.e. they are underpaid)
- Task is to catch corner cases automated systems miss

QUALITY ASSURANCE

- Quality Assurance (QA) assures product's quality is at acceptable, expected level for customers
- Feedback loop:
  - Design –> Develop –> Test
- Dedicated QA expedites process of tracking and correcting bugs and features
- Complementary role to designers and developers

IDEAL BUG REPORTS

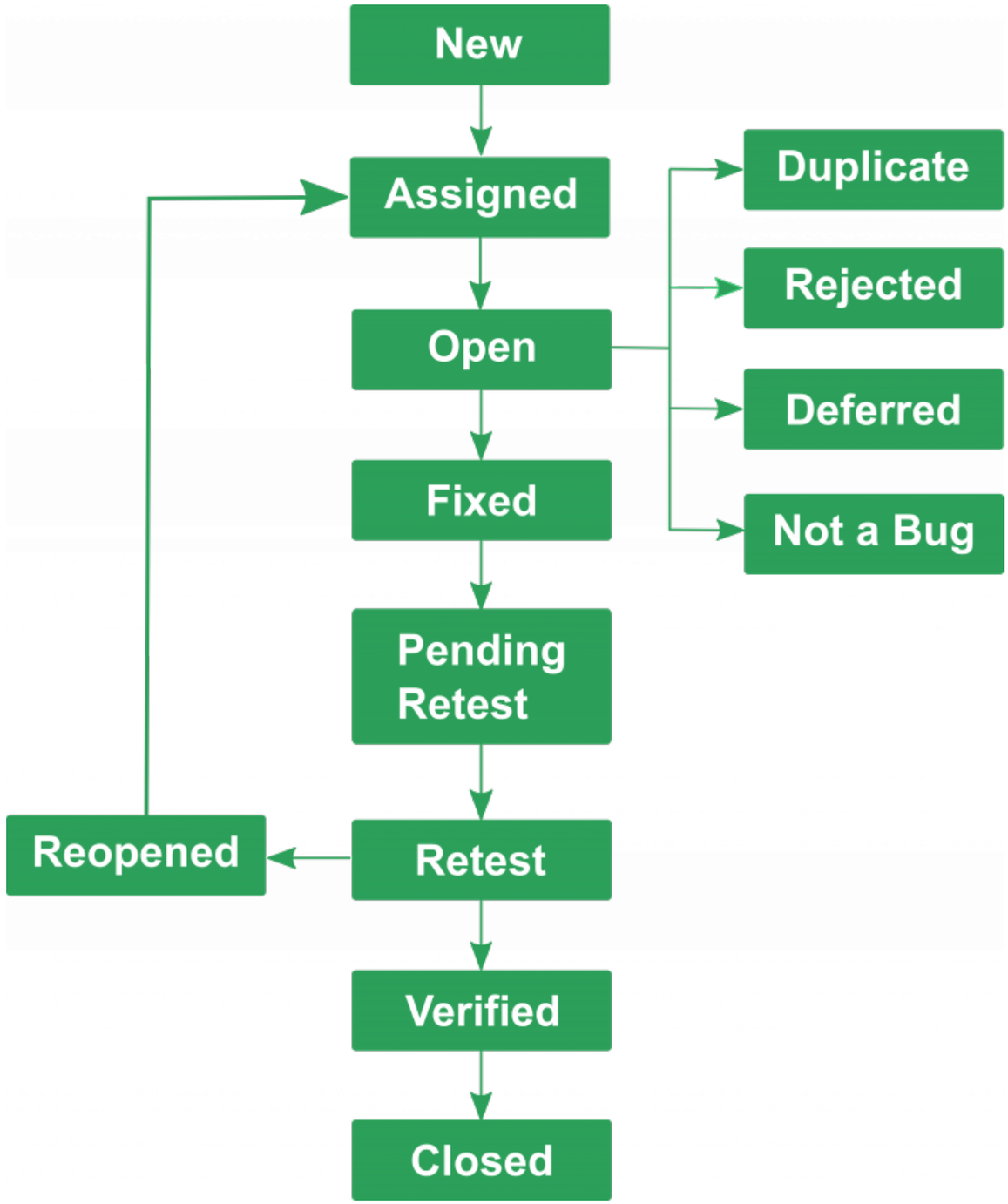
- Bug reports should have:
  - Descriptive title
  - Bug summary
  - Encountered behavior
  - Expected behavior
  - Steps to reproduce
  - Screenshots or video
- Other things to track are the product, platform, build, priority, severity, and status

CONVEYING BUG INFORMATION

- **Descriptive title** should explain the issue, including what system is involved and the specific error that occurred
  - "System crashed" is bad…
- **Bug summary** should describe how and when the bug was encountered
- **Encountered behavior** should be descriptive, accurate, and as tangible as possible
  - "Blood is too gloopy" is bad…
- **Expected behavior** should be accurate and specific
- **Steps to reproduce** should *be as detailed as possible*
  - May need to include hardware/periphery information, etc
- **Screenshots or video** of bug should be concise and include notations if possible

BUG LIFE CYCLE

- Efficient fixes require efficient bug tracking
- Easy to have duplicated bugs in a bug tracking database
- Can be difficult to reproduce issues
- Bug retesting must occur within the correct build to avoid “recurring” bugs



BUG TRIAGE

- Process of assessing bug severity and priority
- Bug severity determines how serious (i.e. game-breaking/profit-losing, etc) a bug is
- Bug priority determines how important it is to fix a bug
- Some examples:
  - What could be a high severity/high priority bug?
  - What could be a low severity/low priority bug?
  - What could be a high severity/low priority bug?
  - What could be a low severity/high priority bug?

CODE FREEZE AND ASSET FREEZE

- Testing is *always* more extensive than development
  - Good testing requires controlling the inherently chaotic development process as much as possible
    - Consistent build numbers and incremental changes, etc
- Code freeze/asset freeze says no new features or assets will be accepted into the game
  - Mostly just recommendations in your classes…but **hard stops** in industry!
- If your team lead gives you a code freeze/asset freeze deadline, you **cannot** make changes to that code after the deadline without permission
  - Essential to at some point move from the “feature-building” stage to the “bug-fixing” stage

FURTHER READING

- Testing and Test Driven Development in UE4 <https://benui.ca/unreal/unreal-testing-tdd/>
- Automated Testing in UE4 <https://kobiton.com/automation-testing/an-introduction-to-automated-testing-for-an-unreal-engine-project/>
- How to write a bug report <https://geteasyqa.com/qa/write-bug-report/>