|  |  |  |
| --- | --- | --- |
| **Bug** | **Cause** | **Steps to avoid reproduction** |
| Careless implementation of bit manipulation routines caused bugs. Specific: - subtracting of 1 after the inversion of the bits meant the subtraction was having the wrong effect. | - Lack of thought before implementation of the methods  - Lack of rereading of the new code to understand the code and data flow.  - Lack of thinking about ways that the code could break. | - Spend more time thinking before implementing any code. No matter how simple.  - Make it a note to reread every line of code before moving on to the next task at hand.  - Write test cases first will aid in defining a clear interface and thinking about the potential inputs that will break the code.  - Step through all of the code at least once in order to understand all of the data flow and what is happening in the code. |
| A hard coded value remained inside of the code causing unexpected behaviour. | - Placing a hard coded value inside of a script and not going back to it on the same day. | - Avoid placing hard coded values in code. - If it must be done, write it down somewhere in order to not forget. Also, attempt to return to it by the end of the day as the more time passes, the less likely you are to remember it. |
| Object is null when being accessed in Unity | Not enough consideration was made in determining where an object might have become or is null. | - Take time to think about how object references might become null or might have been null. Assure that that case is thought of and covered. |
| Character would phase through damaging obstacles returning a false contact normal. | Since OnTriggerEnter2D doesn’t provide a contact I was using the vector between both transforms. However, instead of doing that, I’m now doing the vector between the obstacles transform and the centre of the collider. | - Consider what would happen if the time steps were large or huge movements in between frames and determine what would give you the most accurate results. - Don’t require directions from a trigger enter, only use non-triggers instead. |
| Character would jitter quickly between two values in response to another character | This was due to using a single threshold as the catalyst to make the other character move out of that threshold. However, once out of the threshold, the character would immediately attempt to reenter it, thus causing jitter. | - Use ranges instead of a single threshold value when reaching the threshold can cause the dependant value to jitter in and out of the threshold. |
| Due to an incorrect assumption about the designers use of the tools, a method did not have the desired effect. | This was due to incorrect naming of the method and incorrect assumption of what the designer would do. | * Have a clear understanding of the designer’s goal. * Don’t assume that the designer will know the meaning of a vague method name. * Add generality in order to support the possible uses of the designer. |