Interview with Cristian Deri on 2023-05-16

1. **Can you describe the game testing process? What tools/techniques do you use?**

   There are three ways to test a game;

   Unit tests (UNIT) = Smaller tests. To tests a limited part, i.e. unit, of the game.

   Integration tests (INT) = Tests integrations of units and dependencies.

   End to end tests (E2E) = A test where you run the full game in the game engine.

   E2E-tests are very computationally expensive so ideally those kinds of tests should be run the least and be at the top of the testing pyramid. Now, however, it's mostly the other way around as most tests are E2E-tests. The QV/QA-team can independently run a E2E-test but the other tests needs to be initiated by the groups working on the game, as they require insight into the game build and process early on.

   Cristian and his team currently work exclusively on Battlefield, a sandbox FPS game which require a substantial amount of testing, and therefore rely very much on automated testing. They primarily work on E2E-tests using bots, but also assist on the other types of tests. As the tests are very computationally expensive they need a lot of computational power to run. Therefore, to be able to scale the testing capabilities depending on the current need, they've transitioned towards using cloud based solutions.

   In the tests they collect logs from the test software, game logs, screenshots and videos. To identify failure they look for specific events, such as a string in a log. Since all rules are specified in code they are not that intelligent, so it would be preferable to be able to detect patterns. Screenshots and videos are, at a basic level, processed with computer vision algorithms and then checked manually by a team member. Visual recordings are normally focused on areas that are, beforehand, identified as "bug prone".

2. **In which stages of the game development process do you test the games? Do the testing differ between testing round/stages?**

   Previously, game testing happened mostly towards the end of the game development process. Now, however, Cristian and his team are now getting involved earlier in the process. He call this a "shift left" strategy, where they incourage less expensive unit or integration test early on so all units and integrations works before running the more demanding E2E-tests.

3. **What do you know about the game before testing it? I.e. are they tested without prior knowledge or do you, for example, get a game specification beforehand to follow.**

   Now, with the "shift left"-strategy, they validate that key requirements are met before development starts and therefore get information beforehand. Before, it was more of a "black box"-testing.

4. **Have you used any automated tools or machine learning algorithms in your testing process? If so, what challenges have you faced?**

   In collaboration with SEED they've used machine learning to pilot helicopters, as it's very hard to do with bots. They've also used computer vision techniques to check textures and automatically raise bugs.

   The largest challenge to implement machine learning is that it requires a certain know-how to be accessible. There are currently no common practice so it can be hard for the developers to get started.

5. **What is a bug? Can there be difficulties to distinguish what is a bug and not?**

   A bug is a behavior in the game that is unexpected compared to the designed intent. If a bug impacts the game experience negatively it should be removed. Some bug can be regarded as shippable, which means no action is taken to remove them.

6. **Which types of bugs the are most common? Are there any bugs that are specific to certain game genres?**

   Cristian have worked primarily worked on FPX and action games.

   Common bugs for these types of games include:

   - Bugs associated with weapon damage. A weapon does the wrong amount of damage, do not damage the right person or problems with Time To Kill.
   - Visual glitches. This includes wrongly stretched textures, missing textures and problems with the size of models. It also include memory corruption that, for example, result in flashing artifacts.
   - Geometry issues. This include falling through the floor, being able to go through a wall or getting stuck on a part of geometry. It also includes problems with visual clipping, when you for example can see through a wall, and problems with doors.
   - Game Crashes.

7. **What are the more uncommon bugs or bugs that are hard to find?**

   - Persistence bugs, which is bugs related to actions that should be counted and added to a state/score and saved.
   - Physics bugs where action and reaction doesn't match.
   - Game play bugs, such as when a weapon or item is picked up but disappears.
   - Gadget functionalities, where a gadget doesn't work as expected.

- Network related problems such as scattered movement as a result of network disfunction.

8. **What types of bugs are hard to find with bots?**

   Problems with visual clipping, such as seeing through walls, are hard to catch. This, as everything then looks fine except that you can see something you are not suppose to see.

   Everything that needs to be validated by the UI or graphics is hard to find, as the computer vision algorithms that are used now can only detect more basic errors.

9. **Where in the game do you usually encounter bugs? Are there specific types of geometry that are more likely to contain bugs? For example, houses versus terrains.**

   Vegetation and terrains often causes performance problems. It's also more common to get bugs such as getting stuck or falling through the ground with this kind of geometry since the geometry itself is more complex than a flat surface. It's also very common with bugs in relation to water.

   In houses, with narrow spaces and corrides its common to get problems with visual clipping, such as seeing through walls or doors. Overall, doors involves a lot of issues.

10. **What elements should be included in a 3D game world to perform good tests?**

    Water, since it is rather complex and therefore usually causes bugs. This, as it both requires the character to transition between the states of walking/diving/swimming, and the sound to transition, which can cause multiple problems.

    Doors, as they are in general are a hugh problem. Both when it comes to visual clipping and the fact that they switch between the state of open/closed, depending on game circumstances.

11. **Are proper visual representations important to do the test? For example, the actual model of a table instead of just a block to represent a table.**

    It depends a bit on what type of game it is and the art-style of the game. It is needed to find low quality or wrongly stretched textures.

12. **Is it possible for us to get access to jira-ticket documentations of bugs? If not, what information do you usually include in the documentation.**

    LiU can't get access to the actual project documentation but can get an example of a jira-ticket, to see what information it normally include.

13. Anything you like to add, that we should keep in mind going forward?

The debuggability of the platform, with good recordings and logs, are very important to track bugs.

Notes written by Julia Hannu