Automated Testing in Game Development

A crucial but yet complex task

May 17, 2022

Chloé Porion (porion@kth.se)

1 The context

The video game industry is booming. Both the number of players and the valuation of the global market are expected to grow in the years to come. Video games are more and more complex and users' expectations are high, new games have to line up with the other products on the market, especially for high-quality games. Market laws are tough, and years of development as so as millions of dollars often raise high expectations but can also cause a huge loss of prestige and revenue for the studio in case of a buggy release or unsatisfactory user experience. For example, the game "Cyberpunk 2077" [2][7] released in December 2020 raised many critics for its poor quality resulting from a lack of testing. The results? A loss in market capitalization of the company of more than $2.1 billion in only 2 days.

Even if a game enters the software category, its development is much more complex, the game industry being at the edge of technical complexity. The game industry has specificities starting from the product's objective: software aims to provide a service while a game is made for entertainment [2]. Plus, games are made to be changed and completed with DLCs [1] depending on the audience's needs and behaviour.

2 Challenges and reasons of using automated tests

Reluctance to automated testing is quite common in the game industry due to the very nature of game development [2]. Agility and adaptation to the players' feedback and designers' creative desires are essential and often viewed as incompatible with automated testing. The game design is built to change, even core mechanics can be revisioned, making documentation and tests obsolete. Non-determinism, ensuing from AIs, randomness, and even multi-threading, is another argument game developers can raise against automated testing. So as economic and time costs. In addition to appearing incomplete, as the scope of the game might appear too large to explore all paths in the tests, the tests might be buggy and are non-reusable leading the developers to consider that spending much time and effort in building them is not worth. Finally, the fun-factor is a key cause for teams not to reconsider their testing process; testers assess the game entertainment which is very human-centred and hard to automate.

While automated testing appears for some as a brake on development, others see it as a way of ensuring quick release, meaning a bug fix or feedback within a week. Games need regular content updates to line up with the new standards, concept called the Game as a Service model [1]. With the lifespan of games getting longer - GTAV [2] was released in 2013 and is still regularly updated - automated testing allows developers to focus on the latest features, without worrying about breaking the existing game. Low-level testing is often neglected for the sake of gameplay manual testing [2] but the latter is often redundant and reproducing some bugs can be challenging, leaving some unsolved or unidentified. Regularly manually checking basic features is time-consuming while a simple test scenario can be built and automatically run on all builds. Plus, manual testing is not scalable; testing and retesting all configurations and interactions would require hundred of hours of manual tests in the most complex games [5][1]. The test suite is layered to reflect the complexity of a game, each layer representing a different kind of test.

[1] Downloadable content (DLC) is additional content created for an already released video game.

[2] Grand Theft Auto V is a 2013 action-adventure game developed by Rockstar North and published by Rockstar Games.

3 Types of tests

3.1 Unit testing and asset audit checks

The first and easiest group of tests to use is unit tests. Those low-level tests ensure the functions do what is expected. They are both easy to write and debug, they are a basic but still strong barrier to bugs. Asset audit tests, checking the internal state of the assets of the game, are also basic pieces of automated game testing. "Sea of Thieves" contains for example around 80,000 asset audit checks [8][1], representing around 80% of all automated tests of the game.

3.2 Integration testing

But testing single objects doesn't guarantee the whole will work. Integration tests have to be built. They cover a whole feature or action and imply communication between units. In "Sea of Thieves", the integration tests consist of an empty Unreal 3D map that is populated with only the assets needed to test the action [8][1]. For example, to test the action of spinning the wheel of the ship, an empty map with only the player and the wheel is created, see Figure 1 arrow 1.

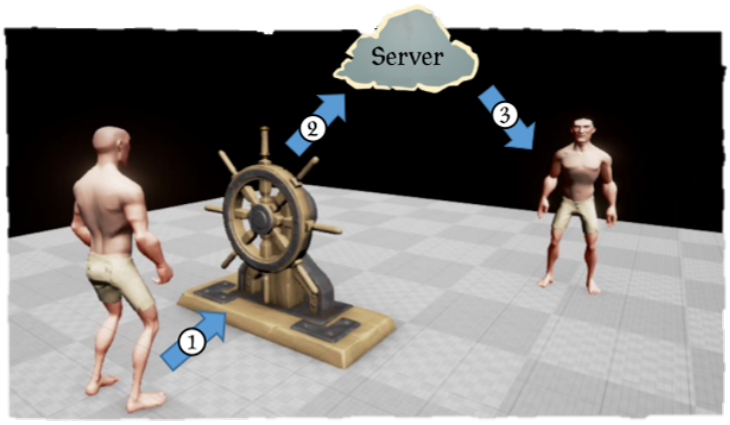

Figure 1: Sea of Thieves / Interactions between players and the wheel [8]

The scenarios are created using Unreal blueprints which is a node-based visual scripting system within the Unreal engine. Code could have been used but visual scripts make it easier to read, write and maintain (Figure 2).

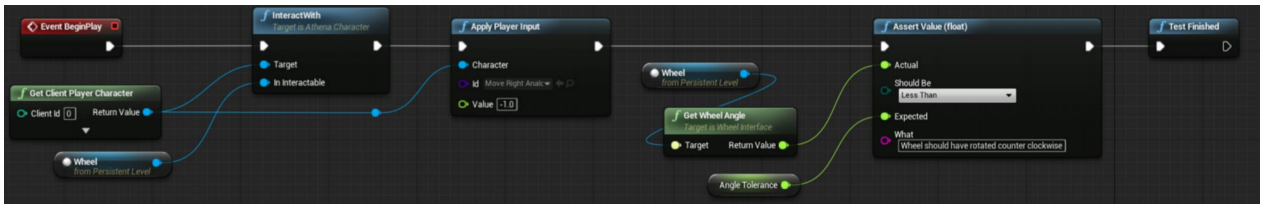

Figure 2: Sea of Thieves / Blueprint for testing the player-wheel interaction [8]

For online and multiplayer games, the integration through the network should also be checked. Integration tests can be slightly modified to add another player to the scenario and check that the actions of the first

player are transmitted to the second through the server connection (see Figures 1, arrows 2 and 3 and Figure 3).

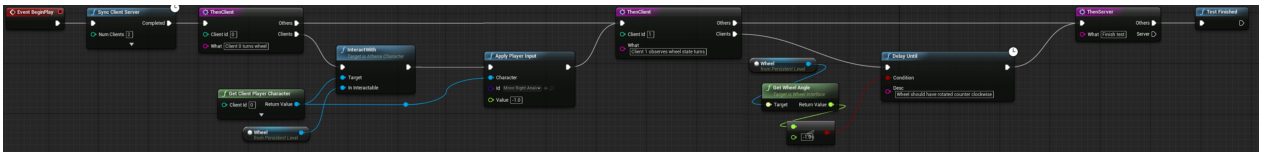

Figure 3: Sea of Thieves / Blueprint for testing the players-wheel network interaction [8]

As integration tests didn't scale well and were costly in run time (20s for an integration test VS 0.1 for a unit test), Rare came up with a middle ground test between unit and integration, "Actor tests" [3][8][1]. They do not require a complete build of the game, reducing the number of dependencies. The downside of actor tests is the use of functions like "tick" (to go to next time instant) that wouldn't normally be called because handled by the engine's update loop. The creation of this new type of test resulted in a huge reduction of run time and they constitute the major part (70%) of all tests (except assets audit checks)4.

Even if violating the general rule of testing that only one thing should be tested, the Sea of Thieves' development team decided to combine features related integration tests. For example, all three attacks an AI skeleton has were tested in a raw so the map, assets, and network connections only had to be created once.

3.3 Performance and load testing

Monitoring and testing the software under different circumstances and configurations (hardware platforms for example) is crucial to guarantee a good user experience and is even more important when building an online multiplayer game. Performance testing is about running longer tests to gather data and detect any spikes in memory use, frame rate, load times, ... [8].
The tests are run at specific locations in the map: the player is teleported at the location, wait a bit, and the performance data is captured [4]. The data is then analysed to identify any abnormal spike.

Apart from local performance metrics, editors of multiplayer online games have to ensure their infrastructures are robust enough to handle all the potential clients. Load testing is a particular performance test that aims to simulate traffic to test the infrastructure against a large set of connections and requests. The Rocket League's [4] development team used load testing to assess their infrastructures when working on making the game a free-to-play (F2P) [5] [6]. The expected increase in the number of players was around 3 to 5 times the audience of the time. Among the existing frameworks (Apache JMeter, K6, ...), the team chose Locust a Python open-source solution that allows one to simulate millions of simultaneous users and gather data. The system is configured (number of users, spawn rate, etc) using a Python file.

3.4 Screenshot testing

Accessing the rendering output is one challenge of game testing and often requires human testers to access quality and find messy textures or any other visual bugs. Yet, automated testing can be used to access visuals through a comparison of the latest build's screenshot with a stock picture. This is referred to as screenshot testing [8][1]. This doesn't make game testers obsolete but whether delay the manual checks to about every one or two weeks.

[3] An Actor is any Unreal engine object that can be placed into a level, such as a Camera, static mesh, or player start location. From Unreal Engine documentation

[4] Rocket League is a vehicular soccer video game developed and published by Psyonix.

[5] Online games that can be played without a fee.

3.5 Gameplay testing

Gameplay testing, also called end-to-end testing, is done by manual testers. It is the primary testing technique used by game developers today and many studios rely on testers to find bugs or any other abnormality but also give feedback on the actual game. They are black-box tests using exclusively the user interface to perform tests. Manual testing doesn't scale well so DICE, the studio producing Battlefield [5], decided to invest in building AIs for testing. Those auto-players would behave like real players, having access to the same environments and interactions but being guided by objectives telling them where to go and how to act. The AIs are programmed using visual scripting to guarantee easier adoption and upgrade process. The auto-players generate data that is then analysed using a 3D telemetry viewer. Specific tests can also be performed, for example, to test the map integrity i.e. find all spots where the player could get stuck. Auto-players offer both reliability, as the scenarios can be replayed, and scalability.

3.6 A real-game test suite

The final composition of the test suite for "Sea of Thieves" (see Figure 4) illustrates the optimisation that the development team performed turning integration to actor tests.

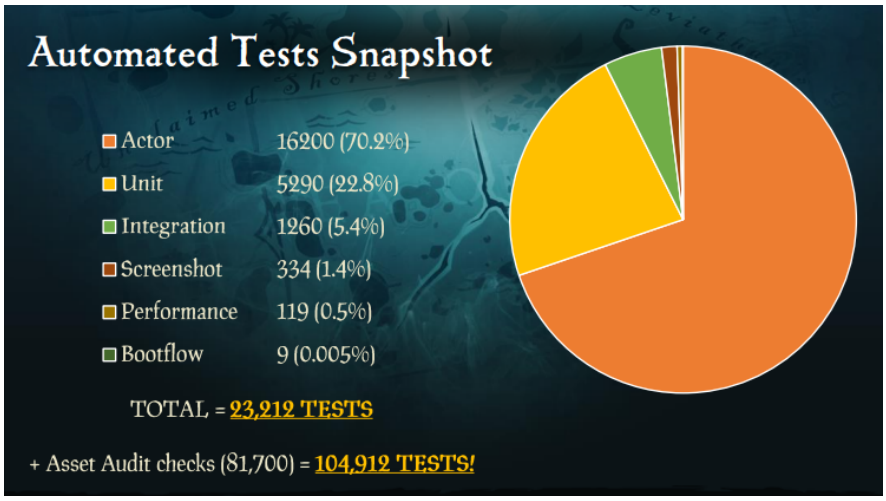

Figure 4: Sea of Thieves / Automated test suite [8]

4 Integration in the CI/CD pipeline

Once built those tests can be integrated into a CI/CD pipeline. Continuous integration software like TeamCity for "Sea of Thieves" [1] or Compass for "Call of Duty" [4] are used. The latter is a build server configured through Python code specially built for testing and profiling Call of Duty games. The tests are automated and a visual interface allows the team to easily get an overview of the results over the last builds and get details for each build like screenshots (see Figure 5).

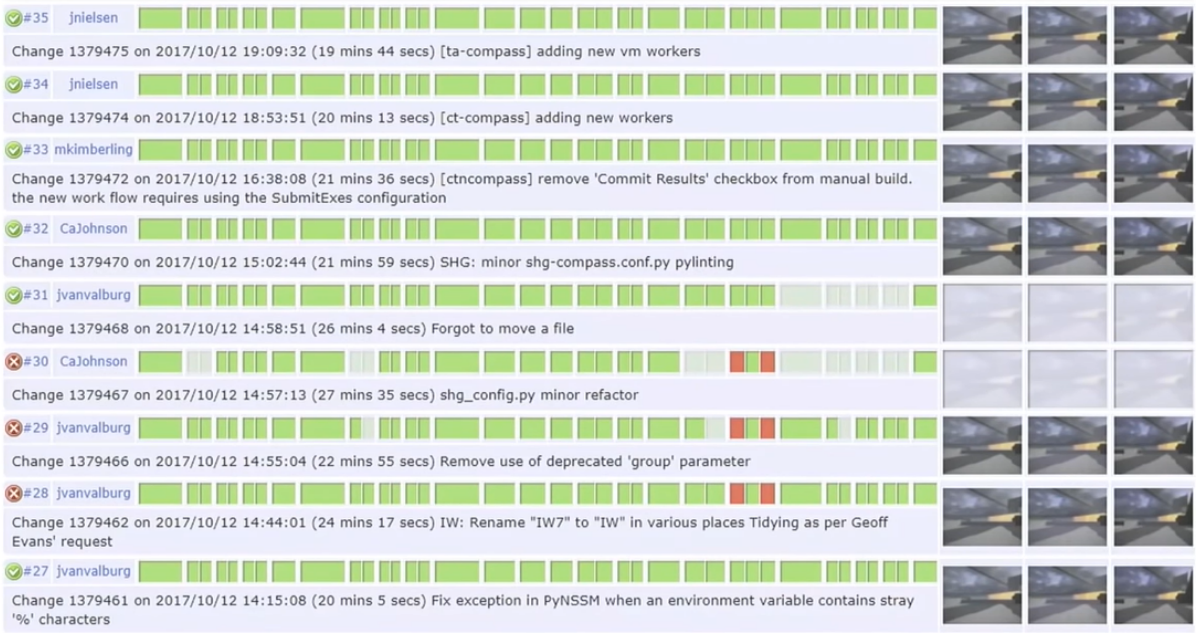

Figure 5: Call of Duty / Overview of the Compass CI server's latest builds [4]

The server also includes data analyses and graphs for performance metrics. All points on the graphs are linked to a commit (see Figure 6) to easily find the source of any performance deterioration.

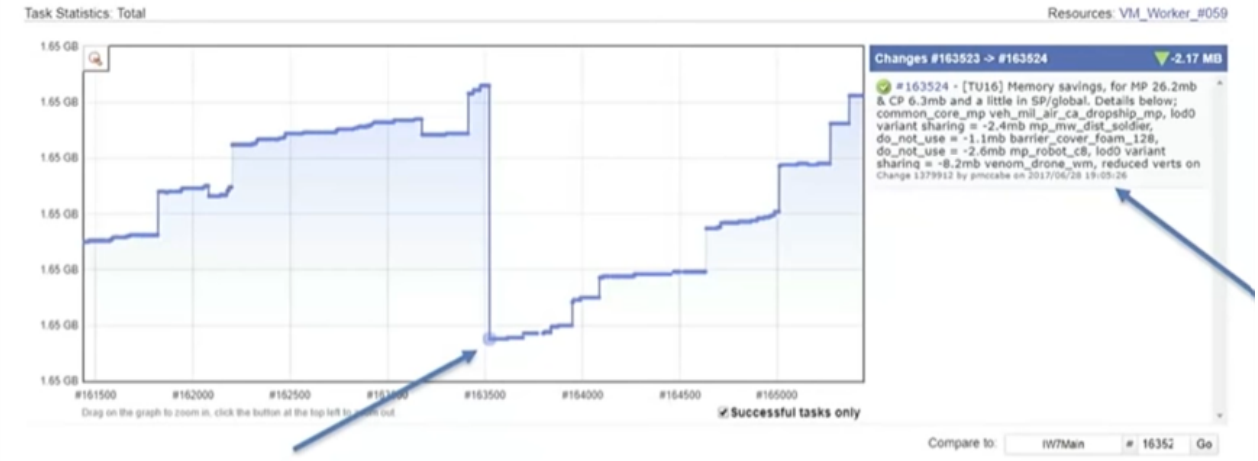

Figure 6: Call of Duty / Compass graph showing memory usage for each commit [4]

Specific CI measures are taken in game development. To avoid pausing the whole team for a context-specific issue (network issue, leaking state from previous tests, etc), both Sea of Thieves [8] and Call of Duty [4] teams used the auto-retry feature. All failing tests were run a second time to ensure the error comes from an actual bug.

As already highlighted by the actor tests, the DevOps techniques have to be adapted to the game industry but the contrary is also true. Development teams have to reconsider their organisation to fully take advantage

of automated testing. In Rare, the all submit process was overhauled to get as close as possible to a constantly bug-free build, allowing in theory a deployment at any time. The developers had to deliver code with a set of related tests and complete a pre-commit where all tests impacted by the new code are run. The higher restriction was that developers were allowed to submit only if the build was green, prioritizing bug fixes to new code. The testing process evolution at Rare can be seen in Figure 7.

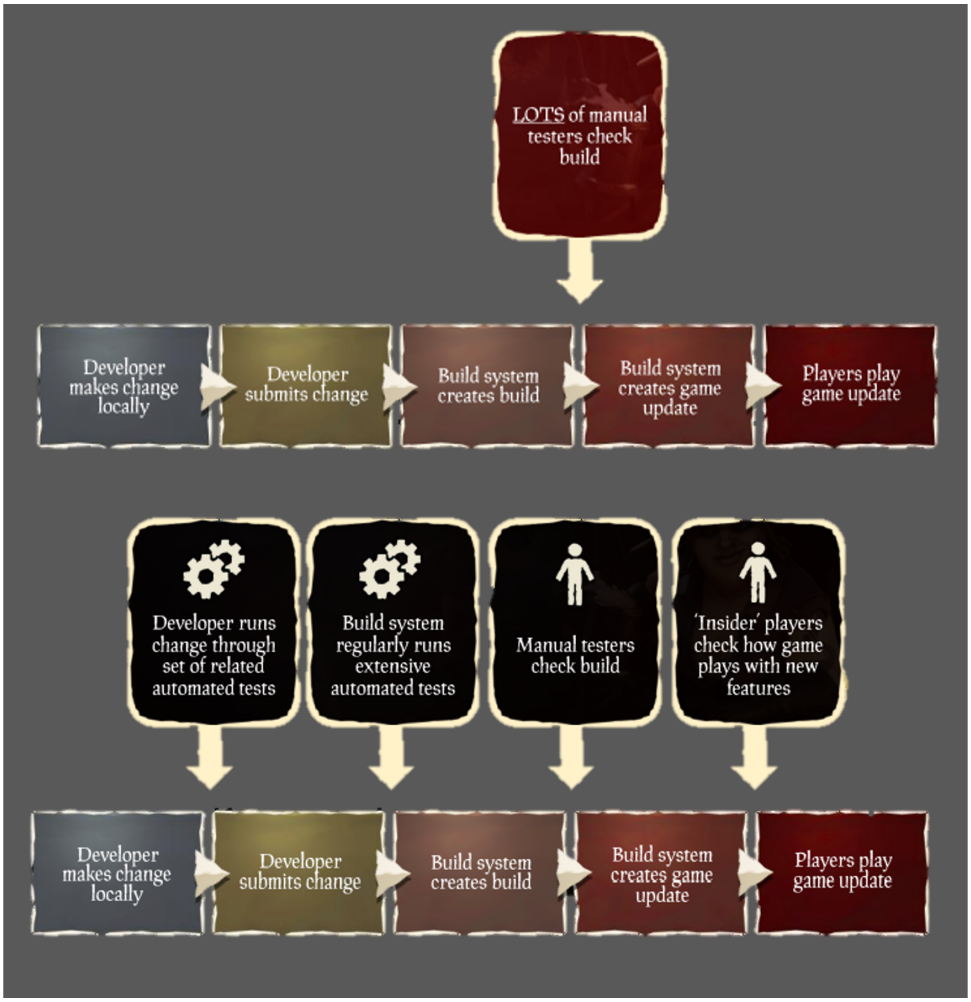

Figure 7: Sea of Thieves / Testing process evolution (top: before, bottom: after) [8]

5 Benefits & Limits of Automated Testing

Time is central when considering switching to automated testing. If it appears for some as a waste of time building a complex non-reusable system that can stop the whole development team in case a bug is detected, others like Jessica Baker from Rare [1] acknowledge that developing new features might take more time but it is time saved for later. Indeed, the commit that led to the bug is easily identified and developers do not have to look back at 3-month-old code. Solving bugs as they appear also reduce the crunch (see Figure 8), this period before release when the team experiences high-pressure while having to solve the last bugs before the game release. This is also the result of development policies prioritizing bug fixes to feature work, preventing

the number of bugs to mount up too much [8].

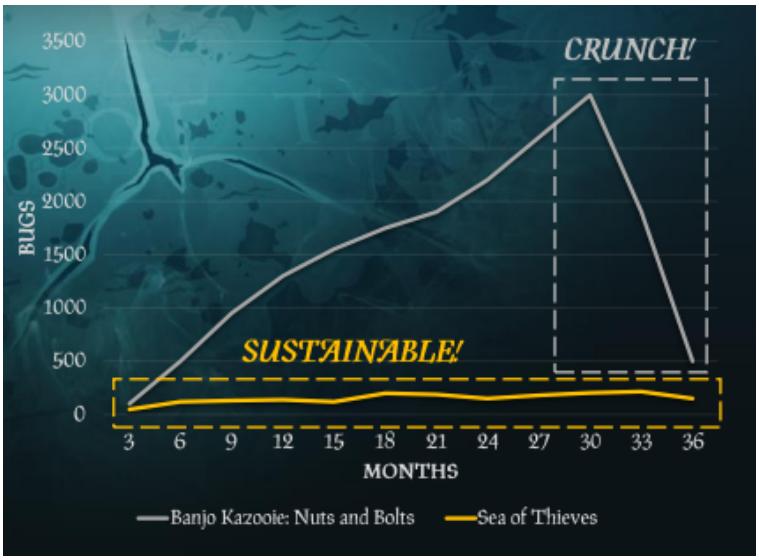

Figure 8: Sea of Thieves / Evolution of the number of bugs detected for two Rare projects [8]

Automated testing allows the development team to more precisely and quickly verify the build compared to a manual-check-only system. The time needed to verify the build was reduced from 10 to 1.5days at Rare between Sea of Thieves and their previous project. But, automated testing is not yet self-sufficient nor one-size-fits-all. Manual checks are still essential for detecting audio and visual bugs and assessing the game experience. Playtesters can influence the game such as in the last "God of War" game [6] released in 2018 [3]. Some playtesters didn't understand why a character would turn aggressive after the loss of her son, so to clarify the story, the developers added a dialogue after the cinematic to explain what the loss of a child could represent. Automated testing is a way to take care of routine tests so manual testers don't have to and can focus on areas not yet covered by automated tests, like graphical and audio rendering, gameplay, user experience, and exploratory testing. So to make it short: Automated testing is more scalable, more reliable, and runs faster than manual testing but still better to have a combination of both.

References

[1] Jessica Baker. *Automated Testing at Scale in Sea of Thieves — Unreal Fest Europe 2019 — Unreal Engine*. URL: `https://www.youtube.com/watch?v=KmaGxprTUfI`. (accessed: 2022-05-15).

[2] Y Guéhéneuc C. Politowski F. Petrillo. "A Survey of Video Game Testing". In: (March 2021). DOI: `https://doi.org/10.48550/arXiv.2103.06431`.

[3] Jeet Shroff - GDC 2019 Ed Dearien Kevin Keeker. *Playtesting 'God of War'*. URL: `https://www.youtube.com/watch?v=Zr4u5Kf_CT4`. (accessed: 2022-05-15).

[4] GDC 2018 Jan van Valburg. *Automated Testing and Profiling for Call of Duty*. URL: `https://www.youtube.com/watch?v=8d0wzyiikXM`. (accessed: 2022-05-15).

[6] God of War is an action-adventure game.

[5] GDC 2019 Jonas Gillberg. *AI for Testing: The Development of Bots that Play Battlefield V*. URL: https://www.youtube.com/watch?v=s1JOSbUR6KE. (accessed: 2022-05-15).

[6] GDC 2021 Matthew Sanders. *'Rocket League': Scaling for Free to Play*. URL: https://www.youtube.com/watch?v=W52Lm505300. (accessed: 2022-05-15).

[7] Jon Richter. *Cyberpunk 2077's Bugs and Glitches Still Hold it Back*. URL: https://gamerant.com/cyberpunk-2077-bugs-glitches-reputation/. (accessed: 2022-05-15).

[8] GDC 2019 Robert Masella. *Automated Testing of Gameplay Features in 'Sea of Thieves'*. URL: https://www.youtube.com/watch?v=X673tOi8pU8. (accessed: 2022-05-15).