

CT60A4160 Fundamentals of Software Testing

Tests report for 0 A.D.

Alpha 25b: Yauna

Group members:

Serhat Altay - 000398424

Chuangzhang Chen - 000210010

Trieu Huynh Ba Nguyen - 000405980

Annalina Wheeler - 000409268

Yatian Zheng - 000213156

Table of content

Document history	2
Introduction	3
Point proposal	5
Testing strategy	7
Environment and test tools	8
Report of testing 0 A.D. development report	9
Existing tests	11
Code review	12
Documentation review	12
Test cases	14
Functional tests	14
Non-functional test	15
Other tests	17
Bug report	18
Risks report	22
Conclusion	24
Reference List	25

Document history

Version	Date	Creator	Latest updates
T1	07.10.2022	Annalina Wheeler	Template was created
T2	9.10.2022	Annalina Wheeler	Testing strategy -section was written
T3	13.10.2022	Serhat Altay	Risk report section was written
T4	14.10.2022	Chuanzhang Chen	Introduction section was written
T5	14.10.2022	Yatian Zheng	Bug report was written
T6	14.10.2022	All	Modified the document to make it more uniform
T7	16.10.2022	Annalina Wheeler	Added to the bug report & wrote the Report of testing the 0 A.D. development report
T8	16.10.2022	Yatian Zheng	Fault report was added
T9	17.10.2022	Serhat Altay	Code review section was written & bug report added
T10	17.10.2022	Yatian Zheng	Conclusion was written
T11	17.10.2022	Chuanzhang Chen Yatian Zheng	Documentation review was written
T12	17.10.2022	Chuanzhang Chen	Introduction & reference added
T13	17.10.2022	Trieu Huynh Ba Nguyen	Existing tests were written, added to bug report
T14	18.10.2022	All	Final revision of the project

Introduction

Our project's objective is to demonstrate what we have learned from the course CT60A4160 Fundamentals of software of testing, with testing open-source projects: 0 A.D. This is a free, open-source, historical Real Time Strategy (RTS) game currently under development by Wildfire Games, a global group of volunteer game developers (McElroy, 2010).

This is a game that is mainly made by C, and it supports Windows, OS X and Linux. 0 A.D. was developed by Wildfire Games, the project began in 2000 in conception and officially started in 2003 as a mod for Age of Empires, but it sought independence because of the restrictions (Tozzi, 2010). The game is still in development and has not yet been completed (Ridgewell, 2010), and the release date of the final version is still undetermined. However, the game's Alpha version can be downloaded from the official website.

In 2012, 0 A.D. received second place in the IndieDB Player's Choice Upcoming Indie Game of the Year competition. 0 A.D. has generally been well received. It was voted LinuxQuestions.org's "Open Source Game of the Year for 2013". Between 2010 and 2021, the game was downloaded from SourceForge.net over 1.3 million times.

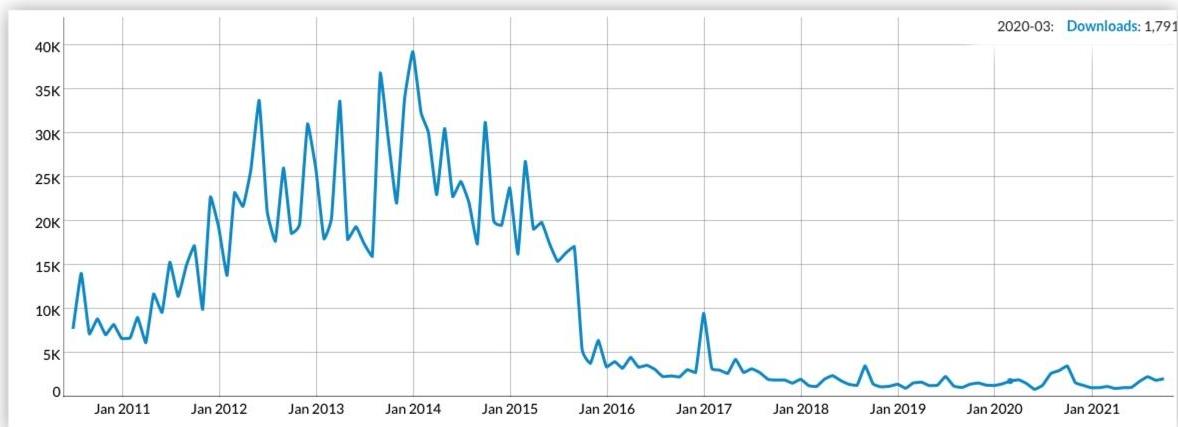


Figure 1: How much the game was downloaded from 2010 to 2021
(Available from: <https://sourceforge.net>)

0 A.D., like many real-time strategy games, contains elements such as building bases, training armies, fighting, and developing technology. The game's camps contain multiple civilizations, and each has its own unique army and buildings.



Figure 2: 0 A.D. interface

From a selection of 12 historical civilizations, players can select one camp (Knight, 2011). Players will initially have a major building resembling a "town center", a number of male warriors, and a few female characters, much to the Age of Empires series. Players can construct residences, barracks, docks, guard towers, and other structures with the materials they collect. The player starts off in the Village level of the game, which splits the civilization's technology into three phases. After that, the player can "upgrade" to the following civilization level through the town center if they have constructed enough structures and gathered enough resources. Players can advance to the Town stage, then the City stage, after completing the Town stage. Players will be able to train more sophisticated troops, create more sophisticated technologies, and construct more sophisticated structures if they move to a new stage.



Figure 3: "Town center"

Point proposal

Feature	Maximum points	Proposed
Well-written report	3	3
Well-written report of testing another report	1	1
Software installation process has been carried out as test case	0.5	0.5
Software installation process has been carried out with at least 3 different devices	0.5	0.5
Software has been tested with several different environments (at least 4) to test whether the minimum system requirements are solid	0.5	0.5
Software has been tested with several different devices to test, for example, whether the user interface scales to different sizes	1	1
Test software for accessibility	1	1
Design 10 functional test cases of your choice and test them	1	1
Design 10 non-functional test cases of your choice and test them	1	1
Code inspection to a part of source code (minimum 100 lines of code)	1	1
Run existing (unit) tests to check the state of software (and the tests)	1.5	1.5
Found a fault and reported it to developers	2	2

Documentation review	1	1
Found bugs and created a bug report table	1	1

Testing strategy

Due to the relatively long history of the game being open to the public from 2009-2010, we will assume that the game is able to function without major disruptions, so most of the software testing carried out during this project will focus on acceptance testing. The entirety of the testing will be carried out by new users, some of which have prior experience in playing similar games as well as users who are completely new to video games. 0 A.D. is an open-source video game and still remains under development, so we are expecting it to be of relatively high quality, but with the potential occurrence of some bugs.

As is common when testing large-scale video games in particular, most of the software testing throughout this project is carried out using the black-box testing approach. However, we will also use some white-box testing as we aim to inspect some of the source code and comment on its structure, readability as well as its resource-efficiency.

The testing process begins when the game is first installed. The first step is to verify, using a black-box testing approach, that 0 A.D. can be installed on each of the test devices and is able to run in their respective environments. After successful installation, functionality testing will be carried out to test whether the game operates according to its specifications, i.e, whether the functionality of basic elements works as it is supposed to. This will be done by following the tutorial documentation and playing the game according to its instructions to see if the main functions, such as training new units, constructing buildings, and attacking opponents can be followed through without the occurrence of bugs or other issues. In addition, functional testing will be used to test in case of audio-visual problems, graphics issues, as well as potential problems with using the chatbox.

Compatibility testing will also be done to ensure that the game meets the baseline requirements of the software and graphics in different environments. Additionally, we will test the game on various devices in order to see whether the user interface is scalable to different screen sizes. However, the compatibility testing is limited to computers as 0 A.D. cannot be installed or played on mobile devices or tablets as such.

The game being tested is quite large in size, meaning that a vast majority of the testing will be conducted as ad hoc testing. We will be playing the game without specific test cases or planning in order to conduct random testing of features that might otherwise be missed by the planned test cases. This may include testing for non-functional requirements, such as accessibility, the storyline of the game, as well as overall user experience.

Environment and test tools

Test environment 1: Windows laptop, ThinkBook 14 G2 ARE, 1920 × 1080

Test environment 2: Apple Mac, MacBook Pro (13-inch, M1, 2020), 2560 × 1600

Test environment 3: TV screen, NEC, X551UHD, 3840 × 2160

Test environment 4: Windows laptop, Lenovo IdeaPad 5 Pro 14ITL6 (14 inches), 2880 x 1800

Test environment 5: Monitor display, Samsung S24F356FHU, 1920 × 1080

Test environment 6: Windows laptop, Lenovo IdeaPad 5 Pro 16ACH6 (16 inches), 2560x1600

Test environment 7: Monitor, Lenovo L24q-35 2K QHD (24 inches), 2560 x 1440

Test environment 8: Ubuntu laptop, Dell Inspiron 15 3501, 1920 x 1080

Test environment 9: Linux Mint laptop, Dell Inspiron 15 3501, 1920 x 1080

The test environment and tools for this project were selected according to the environments available to us for personal use as well as the tools provided by the university. The devices and environments were specifically chosen so that we would be allowed the widest possible range of environments to test. However, limitations to scope of the project remains, since testing was carried out only on the most popular platforms in our possession. In addition, the game is not developed for use on mobile platforms, meaning that it cannot be installed or played on such devices, like phones or tablets.

Report of testing 0 A.D. development report

A development report is an essential part of the documentation of a software project. The report aims to communicate to the stakeholders what changes have been made to the software program, and how such changes have been implemented in the design. The changes made will then be tested to ensure that the new implementations work as expected and that no new bugs or faults have emerged from the changes made. This differs from a regular software testing report in that this section will specifically narrow the scope down to, and focus on, the newest features and improvements mentioned in the development report.

The development report being tested in this section is the latest available 0 A.D. development report, written between the months of September 2019 and May 2020. It was posted on the 0 A.D. community website on June 4th, 2020. The objective of the test report summary is to test the improvements and features mentioned in the development report, as well as create an overview of the development report and the findings made based on the testing done. The testing will be conducted using a black-box approach, meaning that the improvements tested are mostly related to gameplay, performance and user experience. This results in a limitation in testing as some of the improvements in the development report include fixes to the structure of the source code as well as other features that can only be found by inspecting the code itself. Moreover, the testing is also limited as the testing is conducted using the Windows operating system, and some of the improvements include fixes for operation system-specific features, such as for MacOS.

The areas of the development report tested are mainly concerned with both graphical and non-graphical improvements to the software, impacting performance. These include modifications made to the map editor, and visual updates to various graphical aspects, such as trees, plants and animals. In addition, sound has been added to weaponry and animals, and improvements have been made to previous issues with balancing affecting gameplay. Improvements have also been made with regard to internationalization and translation work, with the aim of 17 languages reaching over 90% translation.

Each of these aspects was inspected and tested. In terms of the visual updates, differences can be found in helmets, shields, and other such paraphernalia, which appear slightly more detailed as compared to the previous versions. Additionally, graphical improvements were also made to animals, plants and trees which can also be seen when comparing the new version with older ones. Sounds were also added to weaponry and animals. However, testing these features found that most of the animals, such as pigs, cows and lions were able to make sounds upon clicking on them, however, others, such as blue wildebeests and gazelles were not able to

do the same. In addition, it seems that many of the languages seem to be near-complete as stated in the development report. However, some of these translations when playing the game in Finnish, for example, contain confusing wording used on some of the buttons.

Overall, the 0 A.D. development report was tested using a black-box testing approach. The testing process contained some limitations as the testing device used operates in a Windows environment, meaning that some of the operating system-specific fixes could not be tested. However, although minor oversights and contradictions between the development report and the test results were found, most of the main improvements functioned and worked to improve the user experience as expected.

The report can be found at: [0 A.D.](#)

Existing tests

0 A.D. had a comprehensive testing collection included in the source code. The developers have created their own unit tests, based on what they deemed needed to be tested the most. Although these tests were extensive, it could not be said that the testing strategy for the game was complete. As with most open-source applications, testing was done only when it was considered necessary and on features that affect the core functionalities of the game.

Upon further analysis, it was found that the CxxTest framework was employed for unit testing of C++ components. There are more than 200 of these tests; some of the prominent ones are mentioned here. The file ComponentTest.h contains the class ComponentTestHelper, which could be used to test a single component, while the MockTerrain class generates simple testing terrain implementation with a constant height of 50. Furthermore, CLogger.h has TestLogger and TestStdoutLogger, which are helper classes and deal with log outputs. These tests create a simple gameplay scenario, initialize a map, place units and buildings, and in the end checks if these functions work correctly.

On top of that, the source code also includes a collection of tests for JavaScript components. These tests aim to verify that the graphical and interactive aspects of the game are functional. They make sure units' actions and players' commands work as intended. For example, the file test_Builder.js checks for the appearance and tasks of a builder unit.

Because 0 A.D. is an open-source game, developers and enthusiasts could create their own tests with ease. The internal code of the game was designed in such a way that contributors not only can help to develop the software but also can write additional testing cases to check the integrity of the product. The documentation of the game includes an in-depth guide on how to make new tests.

In conclusion, the existing tests available in the source code of 0 A.D. are exhaustive. They have successfully validated the features and functions of the game. Although there remained some deficits in the testing system, we could say that the developer community has achieved its goal of building an entire free yet complete strategy game.

Link to the source code of the game: [GitHub](#)

Code review

The overall source code of the game is written by various developers using mostly C/C++ (87.7%), JavaScript (6%), Lua (2.6%), HTML (1.2%), and Python (1.0%) programming languages. The specific source code part we chose to analyze is written in C++ programming language and consists of 565 lines of code. The code is used for removing GUI dependency from Canvas2d, and the file of the code is named as ProfileViewer. It uses many external libraries and includes several header files to work with. Most of these header files are inherited from graphics-, lib-, maths-, and ps- parent files in the source code and are used in different classes and functions.

After testing the code using different static analysis tools (CLion IDE is mostly used) for the C++ programming language, the entire code seems to be working without any visible faults detected by the IDE tools. The code appears to be aligned with its purpose and works with existing environmental requirements, and the integration with other parts of the source code is not interrupted by any errors. The contents are also up-to-date and maintained by the developer to suit the new requirements in the external libraries (the last edit day is 05/10/2022). In contrast, the code is written in a way that does not follow any particular coding standard. Additionally, the order of the functions/classes is not in a logical structure. The names of the variables, functions, and classes are not ideally giving a clear picture of the intention of their goals. The instructions describing the function's purpose (function or method docstring) are often created by short descriptions or sometimes even using only a few words, such as "Handle input" in line 258 or "Render" in line 145 inside a one-line comment. Most of the parameters are not described, and their connections with the function's logic are unclear. Furthermore, some of the if-else statements are written without giving an outsider developer a perspective of the overall implementation.

To summarize, the code is missing a specific documentation standard for describing the insides of the code and its functions. This missing specification will be a problem if this code needs to be changed or maintained by a different developer than the code's original creator, making it harder to extend. More detailed descriptions and the logical formation of the code will ease the compatibility of two or more different developers' code implementation if needed in the future. Any incompatibility in future cases will be prevented with a specific documentation standard, and testing will be easier to point out the exact location of the possible errors with the aid of standardized code descriptions for the overall content of the file.

Link to the analyzed part of the source code: [GitHub](#)

Documentation review

Documentation is well-written and there is a lot of documentation for developers. It is constantly being updated and new versions have been released on a regular basis. We chose the document "Build Instructions" as an example. It mentions the application's system requirements and introduction, dependencies, configuration and operation in Windows, Linux and macOS environments, etc. The instructions are clear and easy to understand. Most developers should be able to comprehend the content of the document and get information from it quickly.

The game's extensive documentation makes it straightforward to learn for first-time gamers. Documentation in the form of instructions describes each control as well as the unit's purpose in detail. For game tutorials, in order to help novice players comprehend the overall layout of the game and the purpose of each unit, it provides a step-by-step guide. Each part of the game has its own tutorial so that players can intuitively and visually understand the features and gameplay of 0 A.D. Although the manual is comprehensive and precise, it is outdated because it was last edited in 2018.

Link to the documentation of the game: [Wildfire Games](#)

Test cases

Functional tests

ID	Tester	Feature being tested	Testing approach and type	Test result	Notes
F-1	A.W.	Exiting the program	Black-box test	Pass/Fail	The program automatically opens to a full-screen view which makes closing the window a challenge. The window can be closed using Alt+Enter. However, there is no documentation for this.
F-2	A.W.	Opening the program	Black-box test	Pass/Fail	When opening the software program, it displays a program error message, and refuses to load. The program must be restarted at least twice before it loads.
F-3	T.H.B.N	Training units	Black-box test	Fail	The game randomly stops being able to train new units
F-4	A.W.	Prolonged pause and continuation of the game after	Black-box test	Pass	Leaving the game on a prolonged pause (30 min)
F-5	Y.Z.	Saving and loading games	Black-box test	Pass	Game progress can be saved and loading the game will restore the progress at the time of saving
F-6	S.A	Creating and selecting horse units	Black-box test	Fail	When you create horse units, they are not graphically structured well, making it hard to select individual units. Horses are joined together. Thus individual horse units are hard to choose and give orders on the battlefield. https://ibb.co/CtFx8NH
F-7	Y.Z.	Creating soldiers	Black-box test	Pass	Different kinds of soldiers can be created when there is enough meat.
F-8	C.C.	Construction of buildings	Black-box test	Pass	The user can command units to build Civic Center, StoreHouse, House, Corral, etc.

F-9	C.C.	Winning scenario	Black-box test	Pass	Game can enter the victory screen normally
F-10	C.C.	Failure	Black-box test	Pass	Game can enter the failure screen normally
F-11	S.A	Reading instructions	Black-box test	Pass/Fail	The instruction text is so small and not visible enough. https://ibb.co/TWBCKnn
F-12	S.A	Attack order to the structures	Black-box test	Fail	Attacking soldiers do not damage/demolish any buildings even if they are attacking and there is no defense.

Non-functional test

ID	Tester	Feature being tested	Testing approach and type	Test result	Notes
N-1	A.W.	Typing non-Latin script into the chatbox	Black-box test	Fail	Typing Chinese or Korean characters to the chat box displays question marks.
N-2	A.W.	Using cardinal directions to navigate the game	Black-box test	Pass/Fail	The map moves as the player moves. However, no cardinal directions are indicated anywhere in the interface, which makes navigation challenging in the case that the tutorial directs the user to move towards, e.g, Southeast.
N-3	T.H.B.N	Graphics at long distances rendering while moving	Black-box test	Fail	Graphics on Linux cannot render properly when moving the camera.
N-4	A.W.	Accuracy of translation into Finnish	Black-box test	Pass	The translation of the game into Finnish is complete and fairly accurate.
N-5	T.H.B.N	Accuracy of translations into other available languages	Black-box test	Fail	Incomplete translations for languages, such as Swedish or Ukrainian

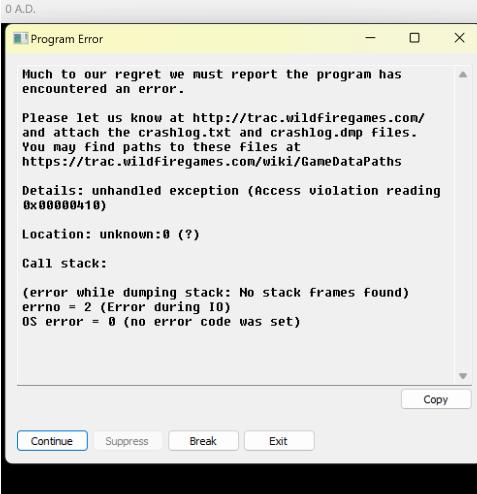
N-6	Y.Z.	Auto-save the game when closing the program unexpectedly	Black-box test	Fail	Unable to find the game in Load Game that was exited due to unexpectedly program closure
N-7	T.H.B.N	RAM usage under 512MB (minimum system requirements)	Black-box test	Fail	RAM usage was approximately 620MB at the home screen and rose to around 1GB during gameplay
N-8	T.H.B.N	Capability to handle a large number of units	Black-box test	Pass	2 players with 1000 units in total
N-9	All Windows testers	Run the game from the installer	Black-box test	Fail	The game cannot be played using the installer.
N-10	All Windows testers	Run the game from the shortcut	Black-box test	Pass	The game can be started from the shortcut
N-11	T.H.B.N	Run the game from the Linux terminal	Black-box test	Pass	The game successfully ran by executing 0ad in the terminal
N-12	C.C.	Music and system sounds	Black-box test	Pass	Game sound can be played normally
N-13	C.C.	multiplayer gameplay	Black-box test	Pass	With host IP, gaForme can be played by multiplayer
N-14	C.C.	Run the game from the Mac terminal	Black-box test	Pass	The game successfully ran in Mac terminal
N-15	Y.Z.	Unit sound effects	Black-box test	Pass	Different units have different sound effects when executing commands.
N-16	Y.Z.	Solutions when the command fails to complete	Black-box test	Fail	E.g no hint and no action to solve the problem when a soldier who is surrounded by a fence cannot go outside the fence as ordered by the player. https://ibb.co/HG64snc

Other tests

ID	Tester	Feature being tested	Testing approach and type	Test result	Notes
O-1	S.A	Automated fishing	Black-box test	Pass/Fail	The fish boat does not have an automatic fishing option https://ibb.co/98m3mm
O-2	S.A	Place soldiers to a specific direction	Black-box test	Fail	When soldiers move from point A to B, you cannot face them in a specific direction. You can only select their formation
O-3	S.A	Building names are not visible if not clicked	Black-box test	Fail	There is no building name when the mouse is on the top of the structures. Players need to click on the building in order to see the building's name https://ibb.co/BPDmXJv
O-4	A.W	Accessibility test: use screen reader to play the game	Black-box test	Fail	There is no way to enable a screen reader for the game tutorial

Bug report

Date and time	14.10.2022
Description	<p>Terms cannot be displayed when the player clicks on the terms of feedback. And some error descriptions are displayed at the top of the screen.</p> 
Test case ID	C-1
Test steps	Click the button “Terms” in the lower right of the screen
Input	Click on the terms
Expected result	Terms are displayed
Achieved result	A few lines of error descriptions are displayed at the top of the screen.
Other notes	<p>The bug happened in the test environment 1</p> <p>Reported to developers</p> <div style="border: 1px solid #ccc; padding: 10px;"> <p>Tia Posted 2 minutes ago</p> <p>I've been having issues with the display of feedback terms on my laptop. Everytime I click on the terms it doesn't show up and displays some error lines at the top of the screen. This happens occasionally on other people's laptops. I am willing to provide more information if needed.</p> <p>Processor: AMD Ryzen 5 4600U with Radeon Graphics</p> <p>Screenshot: https://ibb.co/mcGxj8d</p> <p>Thanks!</p> </div>
Bug severity	Minor

Date and time	16.10.2022
Description	The interface displays a “program error” upon starting the application.
Test case ID	C-2
Test steps	When the user opens the game, the interface displays a “program error”.
Input	Open the application
Expected result	Application will open to the main user interface (starting display)
Achieved result	User interface displays a black screen and “Program error”.
Other notes	<p>The bug happened in the test environment 4</p> <p>Screenshot of error:</p> 
Bug severity	Critical

Date and time	17.10.2022
Description	<p>Attacking soldiers do not damage/demolish any buildings even if they are attacking and there is no defense. The attack functionality is not working.</p> 
Test case ID	C-3
Test steps	The game was played for 2 hours and tested for other functionalities. The bug seems to have happened after playing three different scenarios in a single-player mood with different nationalities in a row.
Input	Attack order
Expected result	Damage/demolish the buildings by lowering the health bar
Achieved result	No changes in buildings' health bar
Other notes	<p>The bug happened in the test environment 6</p> <p>Video shot of the error: https://lut-my.sharepoint.com/:v/g/personal/serhat_altay_student_lutfi/EY1dj5fLT1GiRwKLX94CeEBUWOZHx0K4UHyfk0B9qS43A?e=efOS1Q</p>
Bug severity	Moderate

Date and time	17.10.2022
Description	A distinct red line that cannot be removed emerges on the map when OS graphics have finished generating it. 
Test case ID	C-4
Test steps	When the user opens the game, a distinct red line emerges.
Input	Open the game
Expected result	The map has been rendered and is ready for normal viewing by the user (without any interference)
Achieved result	A distinct red line that cannot be removed emerges on the map
Other notes	The bug happened in the test environment 2. It will significantly impact macOS system users' gaming experiences.
Bug severity	Critical

Date and time	12.10.2022
Description	Far away terrain fails to generate, black patches appear instead.
Test case ID	C-5
Test steps	Use the cursor to move around the map
Input	Open the game
Expected result	Terrain (grass, forests, mountains, etc.) appears
Achieved result	Random black flickering patches appear
Other notes	This occurred on test environment 7 and 8. Fixed by installing NVIDIA proprietary driver instead of open-source Linux driver.
Bug severity	Minor

Risks report

0 A.D. is an open-source game that has been available to the public since 2009 and has been expanded ever since. It is evolved to have various modes with different gameplay scenarios. The game has been tested with different techniques to examine the desired output and the user-experience level. It has been proved that the game is ready to be in production as it is capable of meeting user requirements in many different environments. Ease of usability, reliability, and availability measures is fulfilled in most of the target production environments available in the market. The gameplay performance is enough to reach the expected output scenario, and the user experience is mainly fulfilled as the overall game scenarios are applicable. As noted in our test cases, the game has a few issues and bugs. However, these issues usually do not prevent the software from being ready for production, as fault tolerance is minimal in most environments, and the acceptability of the game is not affected severely.

The software itself has much to improve, especially in the stability part for different environments. The gameplay is not affected by most external factors; however, some discovered bugs are affecting the user-experience part of the game. The game complexity requires so much stability that software should be more cautious about it. This stability demand of the software can be applied by implementing better testing approaches, such as automation testing, to meet the new environmental requirements and continuous user-experience expectations. Optimizing the most efficient gameplay requirements can be fulfilled by frequently testing the game functions while integrating new software features without compromising the old ones. Since the overall software is entirely based on the open-source idea, the game can be further tested by actively reviewing user reports with dynamic communication channels.

Our testing cases are implemented with various approaches, primarily applying the black-box testing method. The direct focus of our testing implementation was the individual units of the gameplay as well as the overall gaming experience. Most of our test cases tried to mimic the actual gaming scenarios in the best way possible.

We have found many minor issues but also a few critical performance faults in some testing environments. Explored issues and faults are analyzed to assess how much of the gameplay's speed, efficiency, accuracy, and performance is impacted. Thus, our test implementation methods can be considered a well-implemented approach, as we discovered errors and bugs in the gameplay. In contrast, our testing approaches could be more precise by testing more functionality of the individual units using different approaches. Other approaches, such as the white box testing method, could inspect broader and more intricate parts of the software, making the bug discovery resolved quickly by pinpointing the exact location of the software where the discovered bug occurred.

Conclusion

In our project, we focused on acceptance testing of game 0 A.D. We downloaded and installed 0 A.D. in different environments (e.g. Windows, MacOS and Linux) for testing. Testing was done in terms of compatibility, functionality and non-functionality. Black box testing was widely used in our project. Overall, 0 A.D. is a relatively mature game that has been tested by others and has good gameplay and user experience. It could adapt to different environments and screen sizes. Most of the features worked as expected, such as building buildings, creating and training units in different scenarios, etc. However, there is room for improvement. We found some functions that didn't pass the test, such as the inability to type non-Latin text. Moreover, some bugs remained, such as "Program Error" during launch. We reported some of them to the developers to help them improve their game.

Our project has some shortcomings due to limited time and devices. We did not test in a diverse range of environments and did a small number of white-box tests. For more in-depth testing, we should analyze the source code of the game at a more professional level, so that white-box testing could be carried out on a wider scale and try to fix persisting errors. We also believe that it is necessary to do (multiple) regression tests.

Reference List

1. Stan, 2020. *0 A.D. Development Report: September 2019 - May 2020* [online]. 0 A.D. [viewed 15.10.2022]. Available from: <https://play0ad.com/0-a-d-development-report-september-2019-may-2020/>
2. Auvray, Nicolas, 2022. *Git mirror of the 0 A.D. source code* [online]. 0 A.D. [viewed 16.10.2022]. Available from: <https://github.com/0ad/0ad>
3. McElroy, Justin, 2010. *The Joystiq Indie Pitch: 0 A.D.* [online]. 0 A.D. [viewed 17.10.2022]. Available from: <https://play0ad.com/game-info/project-overview/>
4. Tozzi, Christopher, 2010. *RTS Game 0 A.D. Needs You!* [online]. The Var Guy. [viewed 17.10.2022]. Available from: <https://web.archive.org/web/20120330061628/http://www.thevar guy.com/2010/03/23/rts-game-0-a-d-needs-you/>
5. Ridgwellian, Ian, 2010. *An interview with Wildfire Games* [online]. Geek Haven. [viewed 17.10.2022]. Available from: <https://web.archive.org/web/20110628140107/http://geek-haven.com/2011/interview-wildfire-game/>
6. Henley, Dave, 2012. *IOTY Players Choice Upcoming 2012 feature* [online] [viewed 17.10.2022]. Available from: <https://www.indiedb.com/groups/2012-indie-of-the-year-awards/features/ioty-players-choice-upcoming-2012>
7. IGN Editors, 2011. *0 A.D. – PC – IGN* [online]. IGN [viewed 17.10.2022]. Available from: <https://www.ign.com/pc>
8. Knight, John, 2011. *0 A.D.—Stunning Real-Time Strategy Game* [online] Linus Journal [viewed 17.10.2022]. Available from: <http://www.linuxjournal.com/article/10887>