Improving the user experience in Thales' navy warfare game

Graduation Report

STUDENT

Iain Harrison [467824]

Creative Media & Games Technologies

Saxion University of Applied Sciences

467824@student.saxion.nl

COMPANY SUPERVISOR

Weibe Huynh

Thales

wiebehuynh@gmail.com

GRADUATION COACH

Daniel Valente de Macedo

Saxion University of Applied Sciences

d.valentedemacedo@saxion.nl

Contents

Abstract

Thales is a large navy defence company who are currently developing a game in Unity to educate and entice potential new employees and game job fairs. However, Thales' game currently has very pressing UX issues that take away from the player enjoyment of the game. The purpose of this paper is to address these fundamental issues by developing systems and tools that will enable future developers to readily maintain a better level of UX. This study focuses on the development of Tools for the creation of animated asynchronous scene transitions, as well as UI feedback and navigation enhancements. This study used A/B testing to determine the influence of these UX changes on the user's perception of the game, as well as the essential research to determine how to construct these systems in the most pleasurable way for the user. The user experience is everything; even if a game or application has the best mechanics, no one would want to play it if the user experience is poor.

Introduction

Company

The client company for this research study was "Thales." Thales is a multi-billion-dollar navy defence corporation with over 80,000 employees all over the world. Thales develops naval radar equipment and systems for detecting and identifying enemy munitions and vessels. This research was carried out from the company's subsidiary in Hengelo.

Thales is a worldwide organisation focused on the development of products and systems used for naval and aerospace defence. This research project is taking place at the Thales Netherlands B.V. subsidiary located in Hengelo. The purpose of this research is to further develop an interactive navy warfare game meant to inspire people to seek employment at Thales, said game has been in development for 4 years so far starting development in in 2018.

The game's purpose is to provide the user with an enjoyable method to learn about the company and how complex radar systems work during wartime.

Thales desires this application to educate people about the complexity of naval warfare in a fun and enjoyable way. The application should allow its users to compete in navel battles with one another over a networked connection in real time. Most importantly the game should be fun to play and challenging to master.

Assignment

The goal of this study is to make the user experience more enjoyable when using the app. Prior to the intervention of this study, the programme had a very poor user experience, with key difficulties such as limited or non-existent player feedback and fully static frozen loading screens, causing player bewilderment and irritation when using it.

Blake Ross, Co-creator of Mozilla Firefox argues "the next big thing is the one that makes the last big thing usable" (Friis Dam, 2017). As this project has been under student development for 4 years many developers have focused all their attention on the implementation of their big system or feature, resulting in a large amount of neglect for arguably the most important part of any game or application. UX or user experience encompasses a vast range of systems and features focused on improving the user's enjoyment of the application. A game could have the most well-developed complex features but if the game has poor UX people won't want to play it, for this reason this

research assignment is focused on the development of the otherwise neglected UX within Thales' naval warfare game.

Objectives

This study will track the creation of systems and features aimed at improving the user's experience while using the app, as well as the creation of relevant tooling that will allow future developers to easily run these systems and features.

The user

The Naval warfare game needs to educate its users on the use of radar systems during excursions, as well as weaponry during battle. The player should be able to learn how to identify specific traits that are present on unidentified vessels and use that information to determine what type of vessel it is and if it is hostile. In addition, the player should be able to learn what type of countermeasures can be deployed to counteract specific types of incoming projectiles. These systems should be based on how naval warfare takes place in reality and how real-world ships handle these kinds of issues.

Thales' target audience consists of potential employees who may not be familiar with much, if any, relevant information on naval combat. These individuals could have any level of gaming experience and as such need the game to easily present itself with intuitive design.

Problem and analysis

Problem statement

The user experience in Thales' navy warfare game has been neglected for a long time, the game lacks player input feedback and does not display any visual information to the user during scene transitions. This results in an unsatisfying and unprofessional experience playing the game.

Research goals

Through careful communication with the company client and investors as well as other employees working on Thales' naval warfare game these research goals were selected. If this paper can successfully achieve these goals the problem statement shall be resolved.

	Description	Priority
1.	The user feels that their inputs have satisfying responses when navigating the UI	Must have
2.	The user can quickly navigate the UI to start the desired game mode with ease	Must have
3.	The application contains transition effect(s) present during asynchronous scene transitions	Must have
4.	The transition effect(s) present in the application is visually satisfying for the user and inform the user that the application is currently in progress	Should have
5.	The scene transition effects system contains ample tooling to enable future developers to create new transitions and easily use transitions in the game when needed	Should have
6.	Custom UI objects containing input feedbacks are easily accessible and enable developers to quickly make responsive UI	Should have

7.	The systems that involve tooling such as the scene transitions and input feedback have been documented, so that future developers can understand how to use them quickly	Could have

Main question

How to improve Thales game's user experience by developing tooling that enables future developers to create responsive feedbacks, such UI behaviour and scene transitions, improving the player's experience during job fairs, allowing Thales to better present themselves to potential employees within the general public.

Sub questions

Q1	How can transition effects and displaying loading information improve the user experience during scene transitions?
Source	Both primary and secondary data collection will be used. Secondary to collect information on the reasons why and how transition effects are used. Primary data will be collected to understand the effects of this research.
Type	Quantitative research will be used as there is not a clear solution to this issue.
Approach	A combination of A/B testing and desk research.
Results	A/B testing will provide information on the validity and necessity of transition effects, desk research will enable the development of quality transition effects.

Q2	What can be done to improve the user experience whilst navigating through the user interfaces present in the application?
Source	Secondary data will be collected as user experience in relation to interface is already a well-researched subject.
Type	Qualitative data will be collected so that the user interface can experience more attention to detail in its design.
Approach	Desk research.
Results	The attained desk research can identify what is needed to improve the quality of the UX whilst navigating the UI.

Q3	What can be done to enhance the players overall enjoyment of the application whilst enabling them to feel as if their inputs have responsive and rewarding feedbacks?
Source	Both primary and secondary research will be conducted to understand how feedback can become more rewarding, as well as the impact this feedback has on the user experience.
Type	Qualitative research into advanced UX/UI solutions will be conducted.
Approach	A combination of A/B testing and desk research.
Results	The goal of the desk research is to develop responsive and rewarding feedbacks, A/B testing will be used to test its validity.

Current state of the project

State of the UX prior to this research

Observing the condition of the UX in Thales's naval warfare game before this research's intervention, the user would be faced with a UI devoid of any pleasant input feedback. After clicking a UI element,

a few seconds pass before the UI display abruptly switches to the next interface. As a result, the application is depicted as an unpleasant experience.

Watch full walkthrough video HERE

Fig. 1

After opening the application, the user is greeted by a scene with one of the game's boats sitting idle upon some slightly animated waves. Excluding the water, the scene is void of any movement. Interacting with an element of the UI that progress the user towards to play session like "Start Host" will start loading and displaying the lobby menu. Figure 1 shows that not all elements of the menu screen are displayed at the same time, resulting in the feeling of dissatisfaction and unprofessionalism. All the UI buttons displayed on the start screen and lobby menu are relatively small in comparison to the available screen size, this isn't a major issue until accounting for the varied length of text displayed in the buttons (see Fig. 2). People naturally tend to process larger visual imagery faster up until a certain size.

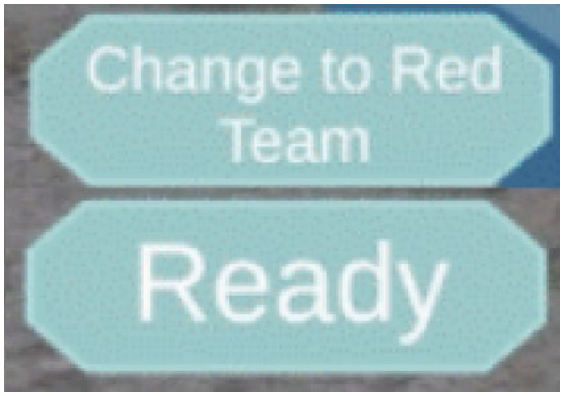


Fig. 2

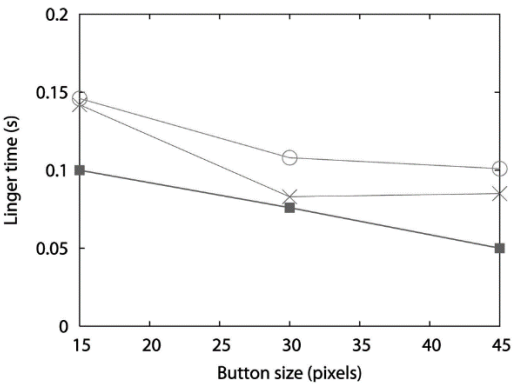
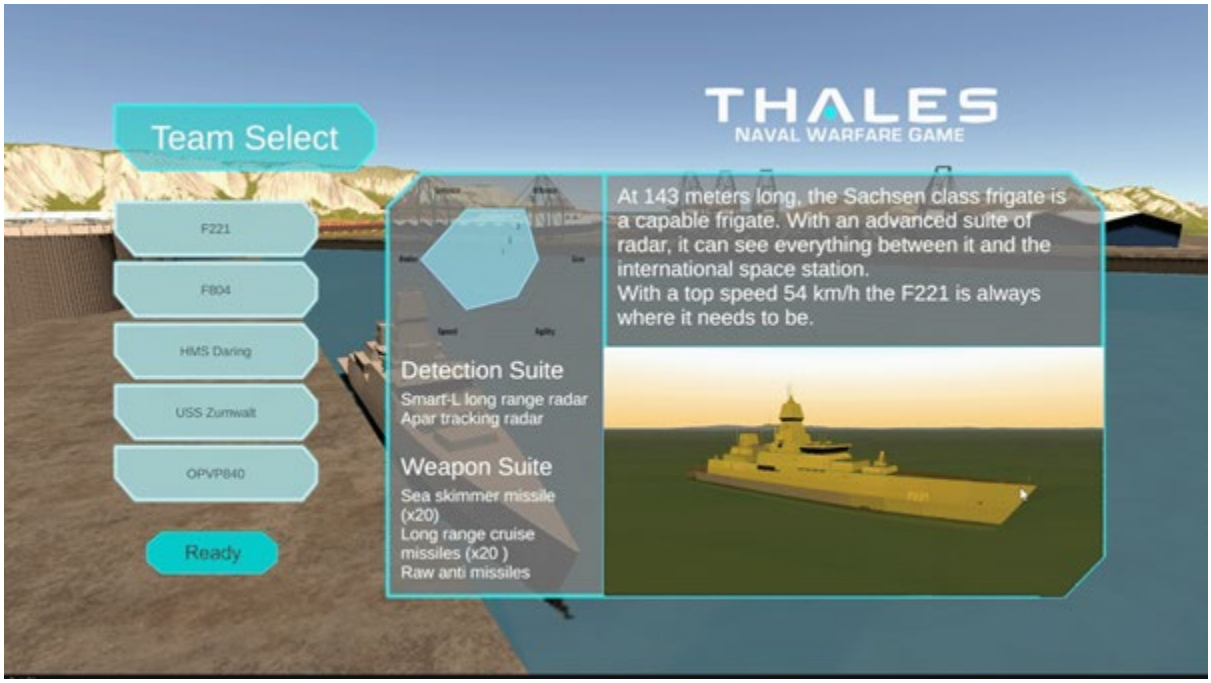


Fig. 3

Figure 3 (Alaniz, 2020) depicts the amount of time a user spends lingering before interacting with a button in a simple application with nine square buttons. The information presented in the graph show that larger buttons produce faster inputs and responsiveness from the user, however in this example effectiveness stagnates after a size of 30 pixels. The UI present in Thales' naval warfare game inhibits only a small section of the screen in the bottom left, therefore a reasonable assumption would be simple enlarging the size of the buttons could improve the UX of the start and lobby menus.



Watch full walkthrough video HERE

Fig. 4

After the user sucessfuly navigates the UI and selects their ship the game attempts to move to the game scene, however before loading in the user first needs to wait in front of a completely static screen. This screen displays the same visual imagrey in the background of the start and lobby screens (as can be seen in Fig. 4 above).

Age	Simple gameplay—the game is easy to learn and play	Interesting storyline—the characters and story are exciting	Fast Performance—the game loads quickly and performance is speedy	I can play the game when I am not connected to the internet
18-25	2.47	2.87	2.92	2.51
26-35	2.65	2.83	2.91	2.63
36-45	2.66	2.76	2.92	2.64
46-60	2.77	2.60	2.80	2.59
Over 60	2.80	2.32	2.78	2.48
Global	**2.68**	**2.67**	**2.86**	**2.58**

Fig. 5

The researcher documented an avarge wait of 50 seconds on their local machine. Figure 5 above displays the results to the question "How important are each of the following when playing a video

game?" This market research into online gaming in 2019 asked participants to rate on a scale of 0 to 4, with 4 representing the highest level of importance, the data in figure reveals that "fast performance" or short load times are more significant to the average user than a games storyline, intuitive gameplay, and whether or not an internet connection is required (Limelight, 2019). In regard to games a waiting time of 50 seconds is perceived as a large expanse of time.

Watch full walkthrough video HERE

Fig. 6

Following the static loading screen, the user is bombarded by three walls of text information that acts as a tutorial of how to play the game. These information panels present the game as an overwhelming and complicated experience, although it is necessary to convey this important information to the player this isn't the optimal way to do so. What makes this experience much worse for the player is the fact that the user doesn't have the choice to move to the next slide, instead each panel in only visible for 5 seconds before moving to the next panel or closing the tutorial. The information present in these 3 tutorial panels is not available anywhere else inside of the application, meaning if the user was unable to comprehend everything within those 15 seconds they wouldn't get another chance.

Preliminary testing

Reason for preliminary testing

This research conducted a preliminary testing phase to grasp an understanding of state of the application prior to intervention. This is being done so that the researcher can conduct a set of A/B testing. The first test, test "A" is the control that is the preliminary test. After the user experience has been improved and the system implemented the second set, test "B" will take place so that the effectiveness of the systems can be validated.

Testing goals

How poorly do test participants view the 50 to 60 second scene transition with no representation that anything is happening?
How useful do the test participants believe the in-game tutorial is?
Do test participant enjoy playing the game and navigating the user interface?

Participants

A total of 20 students from Saxion University participated in the first round of A/B testing. The participants went into the test with no prior knowledge of the game. Participants' ages ranged from early twenties to early thirties. During the playtest, participants were not offered any assistance.

Requirements and scope of the test

Each tester requires the use of a modern computer with a stable internet connection and a local install of Steam (2022). Testers are not to be informed about the game prior to the test and shouldn't receive any assistance from the researcher during the test.

The aim of the research is to test with at least 15 people, with each tester taking no more than 15 minutes to experience the application before being presented with a questionnaire.

Results

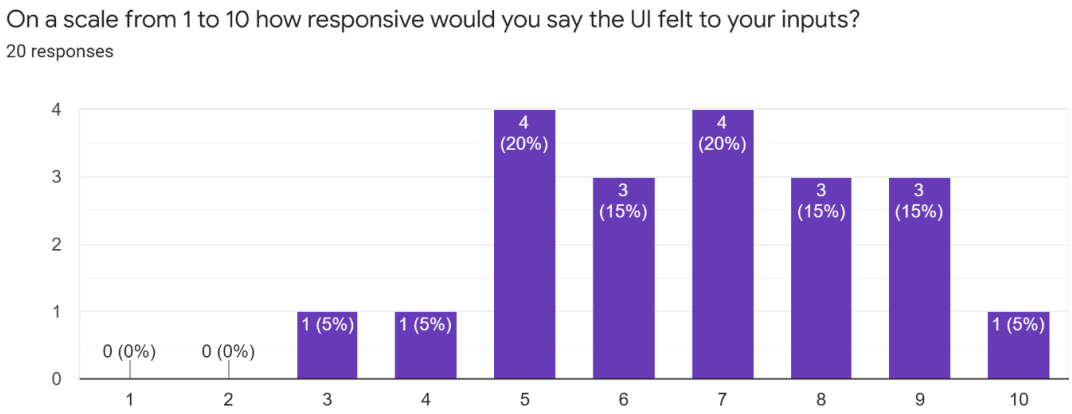


Fig. 7

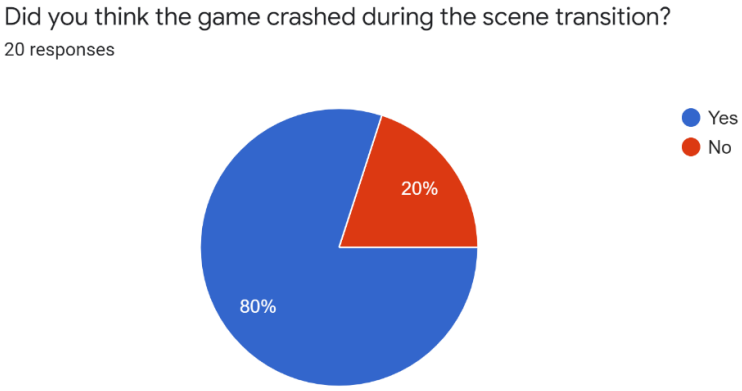

Did you think the game crashed during the scene transition?
20 responses
Yes
No
20%
80%

Fig. 8

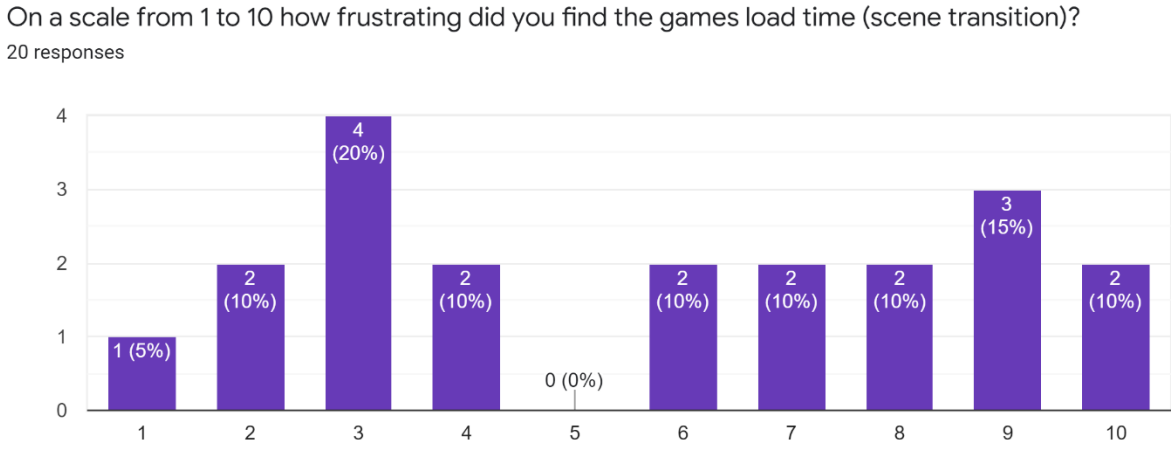

On a scale from 1 to 10 how frustrating did you find the games load time (scene transition)?
20 responses
4
3
2
1
0
1 (5%)
2 (10%)
4 (20%)
2 (10%)
0 (0%)
2 (10%)
2 (10%)
2 (10%)
3 (15%)
2 (10%)
1
2
3
4
5
6
7
8
9
10

Fig. 9

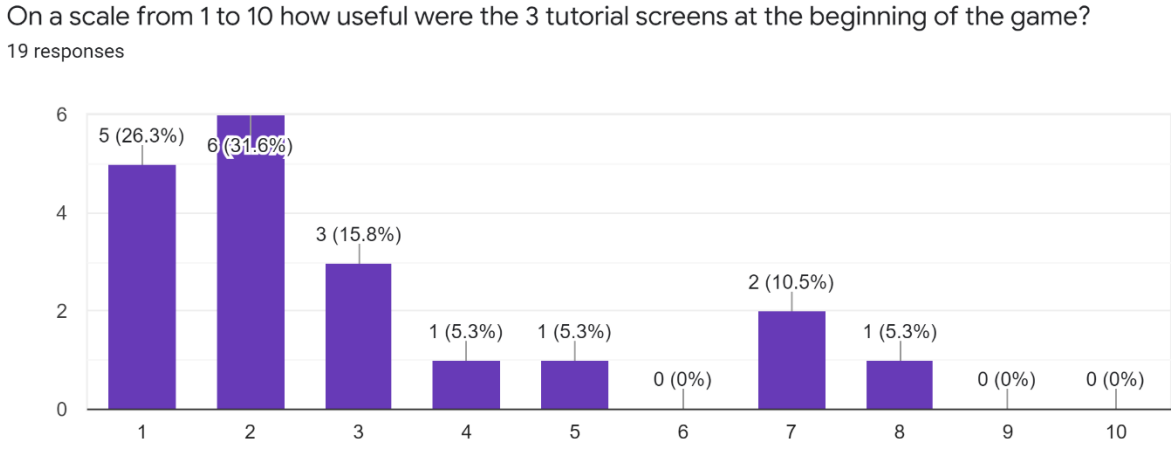

On a scale from 1 to 10 how useful were the 3 tutorial screens at the beginning of the game?
19 responses
6
4
2
0
5 (26.3%)
6 (31.6%)
3 (15.8%)
1 (5.3%)
1 (5.3%)
0 (0%)
2 (10.5%)
1 (5.3%)
0 (0%)
0 (0%)
1
2
3
4
5
6
7
8
9
10

Fig. 10

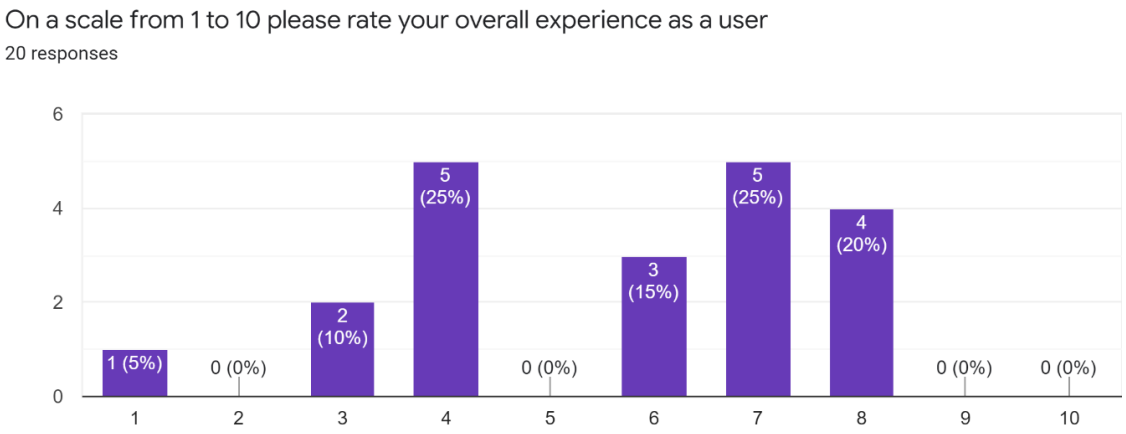


Fig. 11

Reflection of result data

Observing the collected data provided by the preliminary research emphasizes the UX issues present in the project. Most notable is that Figure 8 shows that 16/20 participants believed that the application had crashed during the playtest. This was due to the combination of the long waiting time and the lack of any movement or indication of progress present in the application during the transition.

The collected data in figure 10 clearly depicts that that the 3 tutorial screens were of no use to the users.

Figure 7 shows that the participants think that the user interface is responsive to their inputs, however implementation of dynamic feedbacks should enhance the responsiveness further.

Key take away points of interest

80% of participants believed that the application had stopped responding during the playtest, this clearly indicates that the scene transition to the main game scene is problem area for the game that desperately needs intervention.

4/20 people did not believe that the application crashed during the scene transition, 9/20 people rated the game 7 or higher out of 10 for the overall user experience. Meaning that at least 5 participants both believed that the game crashed and was at least a 7/10 for user experience.

Scope

Deliverables

This project requires the development of systems and tooling targeting developers working on Thales' naval warfare game. The developed systems should enable future developers to easily implement transition effects and dynamic UI buttons.

Inclusions

The scene transition system should allow for the implementation of a variety of different animated transitions. The system should not only focus on the development of a specific type of transition and should instead grant future developers creative freedom when using said system.

The custom developed buttons should not be static, some form of animated interaction is required.

Exclusions

The focus of this project is to enable future developers to use the implemented systems if they so choose, as such animated effects to support the user experience cannot be expected to be fully realized.

Assumptions

The largest transition in game lasts around a minute due to very poor implementation of a pathfinding system provided by a previous employee, although this research aims to improve this area of the application the possibility of resolving this unrelated issue is not likely.

Constraints

The researcher experienced a myriad of setbacks due to a variety of reasons resulting in the subject of this paper to change multiple times, as well as periods of time where working towards the completion of this thesis was obstructed. Therefore, a time constraint was unwillingly applied to the development of these systems.

The research aims to polish Thales' game as much as possible to grant the user the smoothest experience possible whilst playing the game. The game has been in development for over 4 years and this research cannot be expected to solve all UX issues in the project.

Researching user experience

What roles are needed to develop quality UX/UI

When it comes to User Experience (UX) and User Interface (UI) design, there is no one-size-fits-all approach (Mittal, 2018). However, it is critical to comprehend the phases and roles involved in the design process (Mittal, 2018). A game user researcher (GUR), a user experience designer (UX), a user interface designer (UI) makes up a design team collaborating on a project. However, it is occasionally necessary for one person to fill all of these tasks, which is why it is critical to grasp the differences between them (Mittal, 2018).

The role of a User Interface (UI) designer is to incorporate a visual hierarchy into a design so that users can follow it. It is common for UI designers to iterate ideas many times before creating a high-fidelity design. To further support the UX design, UI designers use visuals imagery and animations to better express and deliver information to the user.

A User Experience (UX) designer oversees designing the user's experience when playing the application. The goal of UX designers is to create a product that is simple to use and comprehend whilst also being enjoyable to interact with. A UX designer's primary purpose is to prioritise the user's needs when exploring design solutions (Mittal, 2018).

Professional playtests are conducted by games user researchers (GUR) to guarantee that games are understandable and enjoyable. Making games that people enjoy is dependent on the research gathered by game user researchers.

The importance of dynamic load screens over static load screens

A user's attention is never fully focused on the application, when the user has to wait for elements of an application or game to load their attention will drift. A waiting time of one second is already enough of a delay to be noticed by the user, however this delay is not long enough to disturb the user's train of thought. If the user must wait for ten seconds, they should be informed that the application or game is currently in a process that requires their patience (Card et al., 1991). The wait time greater then ten seconds also requires that the user be informed.

The purpose of a loading screen is not only to inform the user that the application or game is currently busy with a process, loading screens can also be used to redirect the focus of the users away from the waiting period. When a user notices and focuses on the wait time they become more aware of the delay in turn perceiving the wait time as longer than it is (Liikanen & Gomez, 2013) When a person's awareness is focused on a short wait time they often overestimate the delay, when focused on a long wait time a person often underestimates the wait time as shorter then it was (Vierordt, 1868/2020). To ensure that the user has the best experience possible in an application or game the developers should created loading screens that occupies their attention. When a user is distracted by an animation that can hold their attention, they are less likely to fixate on the delay.

Types of loading screens

Across applications, websites any other digital media a vast range of animated loading screens exist. Some types of loading screen are more frequently used then others, but all serve the same purpose to distract or otherwise entertain the user during load times.

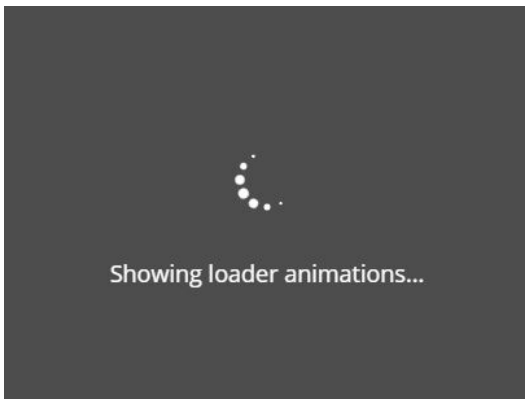


Fig. 12

A spinner is the first sort of loading screen. Spinners are typically shown as spherical centre aligned objects that spin 360' continuously. The preserved wait time is influenced by the spinner's speed. Even though the delay was the same, a spinner that completes a full 360' revolution in 1 second is thought to be faster than a spinner that completes a 360' rotation in 4 seconds, according to a 24 participant study by Soderstrom and colleagues (2018).

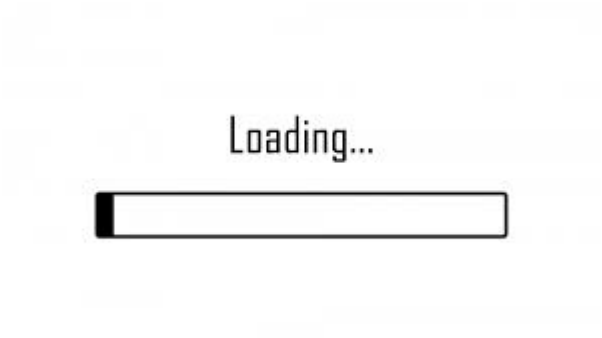


Fig. 13

The loading bar is perhaps the most popular type of loading screen. The most typical representation of a loading bar is a thick horizontal line that is gradually filled from left to right by another overlapping horizontal line. The length of a loading bar influences the preserved time in the same way that the speed of a spinner does. Shorter loading bars are thought to require less time (Kurusathianpong & Tangmanee, 2018).

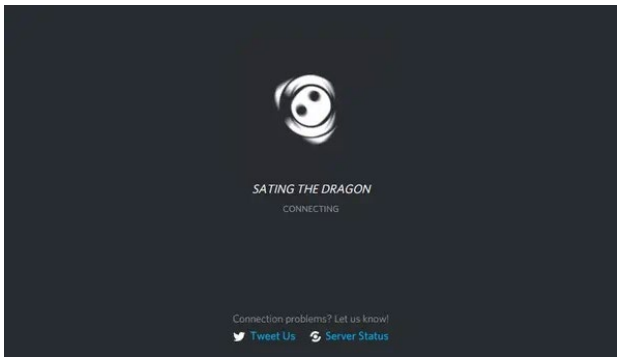


Fig. 14

Although spinners and loading bars are definitely the most commonly used forms of loading screens, they are not the only kind. Loading screens present the opportunity to give a program more personality and flare, many applications and games have created their own custom loading screens. See figure 14 as an example. Discord has created their own loading screen like a spinner but instead of continually spinning, it acts in bursts and segments itself with a dynamic animation. Spinner and loading bars are both very effective forms of loading screens, however the creative freedom of dynamic load screens often makes the experience more pleasurable for the user.

How the speed of an animated loading screen impacts the preserved waiting time

The speed that a spinner rotates impacts the observed time for the transition effect (Soderstrom et al., 2018). The spinner's rotation speed was examined for its effect on perceived performance. Out of the three rotation speeds evaluated, the one that rotated 360 degrees in one second was found to be the best. That was also the fastest spinning spinner, implying that faster loading animations are more likely to provide the impression of faster loading times.

Application navigation and UI

The goal of UX and UI designers is to create pleasant and rewarding experiences for users so that they will return to the application. User interface consistency is one of the most fundamental UI principles for developing enjoyable app experiences (de la Riva, 2021). Fortunately, Thales' naval

warfare game features a light blue and grey UI style that is consistent. Future development of the user interface must keep this style in mind, even during scene transitions.

Perhaps the most impactful that can be done to improve the perceived quality of an application is the implementation of game “feel” or “juice”. These ambiguous terms mean the same thing and that is responsive and rewarding use of visual imagery and audio (Jonasson & Purho, 2015). Using animation, sound effects and particles a developer can make the user feel as if their input have more weight. When a user perceives their action as impactful, they feel more rewarded, they feel as if they are achieving something or in some way. Its this output that their inputs invoke that that make the experience enjoyable, without ample output the user may feel as if they game is buggy or broken (Pereira, n.d.) see figure 8.

It is better for a user to experience something before they understand it. The preliminary study displayed these principles clearly, after users loaded into the main scene, they were first greeted with three 5-second-long tutorial screens. These screens attempted to explain how to play the game before the user even got to see what the game looked like.

Generating ideas

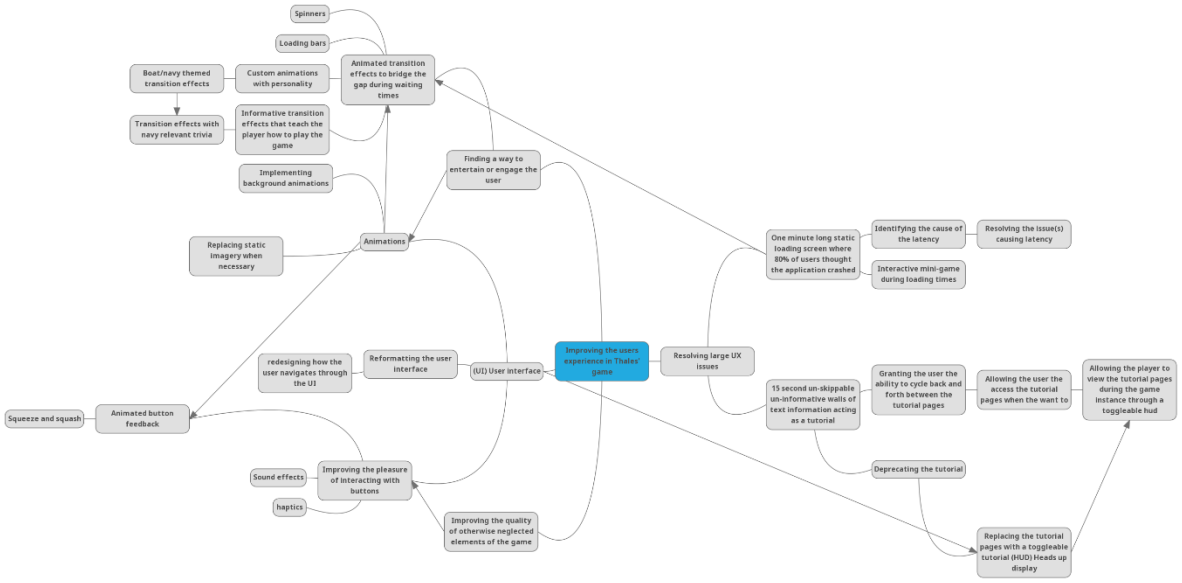


Fig. 15

How to improve the users experience during scene transitions

Prior to this study, the user would have had to wait for roughly 60 seconds during the loading screen, after which the user would be transported to the main game scene. For more than 30 years, people have been aware of response times (Card et al., 1991).

We know that a delay of 0.1 seconds is the upper limit for giving the user the impression that the system is reacting instantly.
A delay of 1 second is approximately the limit for the user's train of thought to remain undisturbed. When the delay is greater than 0.1 second but less than 1.0 second, no specific feedback is usually required.
If a delay would last 10 seconds the user's attention would start to wonder. Users will want to do other things while waiting for the computer to finish lengthy delays, therefore feedback indicating when the computer expects to be done should be provided.

As the delay present in the project is lengthy enough to require information regarding the expected duration of the transition, intervention is required. During the preliminary testing 80% of participants reported that they felt the application had crashed during the scene transition. This was due to the lack of any visual indication that the application was still running, expecting the user to wait for a minute without any form of feedback has proven to be catastrophic and detrimental to the user's experience.

Some form of animated transition is required, most games opt to use the classic loading bar to bridge the gaps during loading screens. Applications and websites often use Spinners, however custom animated loading screens can grant an application an outlet to express personality and creativity. For this reason, the development of a system that can house any form of transition effect is desired. The user experience during scene transitions can be improved by creating a system that allows anyone with access to the project files to quickly generate unique custom transition effects.

Fig. 16

The period of time where the user is waiting for the next scene to load is a potential outlet to convey information to the user. Scene transitions could be used to inform the player about game mechanics they might not know or bits of trivia related to the games theme. Figure 16 shows how the mobile game "Clash Royal" (2022) uses there loading screens to provide their players with tips on how to play the game effectively.

How to improve the users experience navigation and UI

The imagery present in the project prior to the involvement of this research was almost completely static, no movement could be seen on any scene until the user entered the main game scene and moved their ship.

Animations could be used to present the game in a more enjoyable way, animations could be used in the background behind the UI elements during the main menu and ship selection screens.

Animations can also be used to make UI elements like buttons more satisfying to interact with as a user (Jonasson & Purho, 2015).

The layout and general navigation of the project could be overhauled however looking at the preliminary testing an overhaul does not seem necessary. Navigating the UI is simple it just takes a while due to the minute long scene transition and 15 second inescapable tutorial.

Preliminary testing found that the 15-second automatic tutorial was not well received, with more than half of the test participants giving it a 2/10 or worse grade. Implementing a toggleable tutorial HUD that players can use to access this material at their own pace if they so desire is an easy solution to this problem.

Deliverable products for future use at Thales

The goal if this research is to enable future Thales naval warfare game developers the ability to easily implement a more satisfying version commonly used game features, buttons and scene transitions are the focus of this research. The idea is to create an easy-to-use system for creating scene transitions with template transitions already made so that future developers can express their artistic freedom. Custom buttons will also be created with more satisfying input feedback, these buttons should be as easily accessible as unity's default buttons so that future developers use them.

One of the research goals of this paper is to document the tooling and systems so that future developers can quickly learn how to use these features. The documentation will be laid out as a guid with actionable steps that clearly depict the process of creating and implementing an animated scene transition.

Refinement of ideas

SWOT

Using an animation instead of a static loading screen.

Strengths	Weaknesses
Distracting the user form the wait, reducing the perceived waiting time.	As the application is bust during scene transitions animations may buffer.
Opportunities	Threats
To enable the game express deeper levels personality to its users.	Adding animations to the loading screens could extend the duration of the loading screens.

Using a spinner type animated loading screen.

Strengths	Weaknesses
Easily identifiable as a waiting screen.	Lacks creativity and personality.
Opportunities	Threats
Fast spinners effectively reduce perceived waiting time study by Soderstrom and colleagues (2018)	After identifying the spinner as a loading screen users will fixate on the duration of the transition.

Using a Loading bar type animated loading screen.

Strengths	Weaknesses
Easily identifiable as a waiting screen.	Lacks creativity and personality.
Opportunities	Threats

Shorter loading bars reduce perceived waiting time. (Kurusathianpong & Tangmanee, 2018).	After identifying the loading bar as a loading screen users will fixate on the duration of the transition.

Using a custom animation type loading animated loading screen.

Strengths	Weaknesses
Custom animations are more enticing from a user's perspective, distracting them from the wait time.	Time and resource allocation to design and create transition effects.
Opportunities	Threats
An outlet to express creative freedom and give the game a unique personality/style.	Possible performance issues extending the wait time of scene transitions.

Displaying text information during scene transitions.

Strengths	Weaknesses
Outlet for sharing information with the user.	Reduces usable space for animations.
Opportunities	Threats
Thales could use this outlet to inform potential future employees about the work taking place at Thales	Transition times are not always equal, it is possible that the user will not have enough time to read the information.

Creating a system dynamic enough to enable future developers to make any type of loading screen they desire.

Strengths	Weaknesses
The ability to take advantage of the unique strengths associated with different forms of loading screens.	When given to many potential options making a choice becomes more challenging.
Opportunities	Threats
By creating a system that encourages creativity future developers with artistic skills	Reliant on developer creativity.

Creating custom UI buttons with more satisfying input responses.

Strengths	Weaknesses
Making the user feel as if their actions have more weight to them.	Buttons texture overlapping other UI elements during animation.
Opportunities	Threats
The user's overall perception of the application my increase if the UI is more satisfying to navigate.	Over stimulation.

Replacing the 15 second un-skippable tutorial pages with a togglable tutorial HUD.

Strengths	Weaknesses
Enables users to choose when to view the tutorial information.	Any information that needs to be conveyed to the user requires screen space.

Allows users to view and experience what the game is before being told how to play.	
Opportunities	Threats
Potential employees at job fairs can pick up the game and access the tutorial, even after a game session has already been loaded by another user.	Some users may find the “how to play” button annoying as it doesn’t add to the gameplay but is still present on their screens.

After reflection, a decision has been made to develop a custom button(s) with dynamic feedback, as well as an animated transition system with enough modularity to enable the creation of any kind of animated transition. The creation of a toggleable HUD to replace the tutorial.

Developing systems for improved user experience

Developing buttons with dynamic input feedback

Requirements
Buttons should be animated.
Buttons should make sound when interacted with.
Buttons should react when the user hovers over them.
Buttons should be easily implementable for future developers.

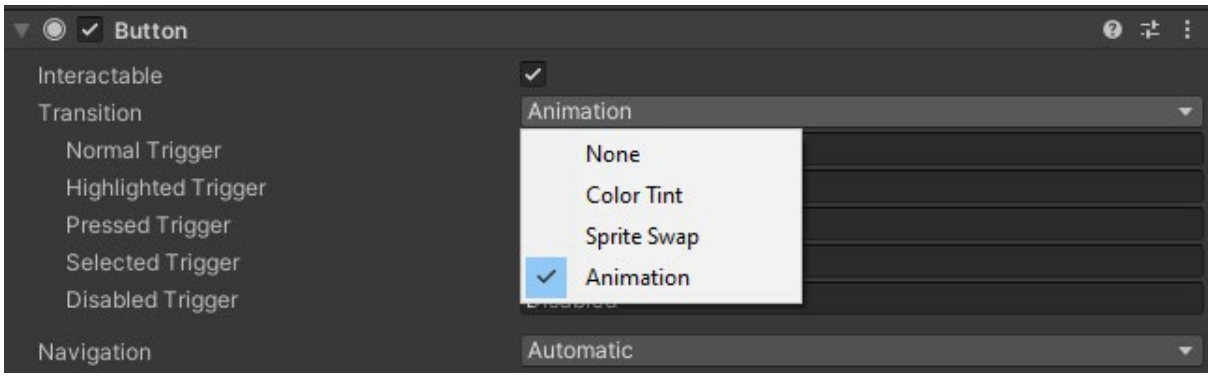


Fig. 17

Unity’s UI buttons have a few options for transitions, by default the “Color Tint” option is selected. Colour tint applies simple colour overlays to the UI buttons image component, switching from color tint to “Animation” and automatically generating an animation controller is the simplest way to animate buttons in Unity.

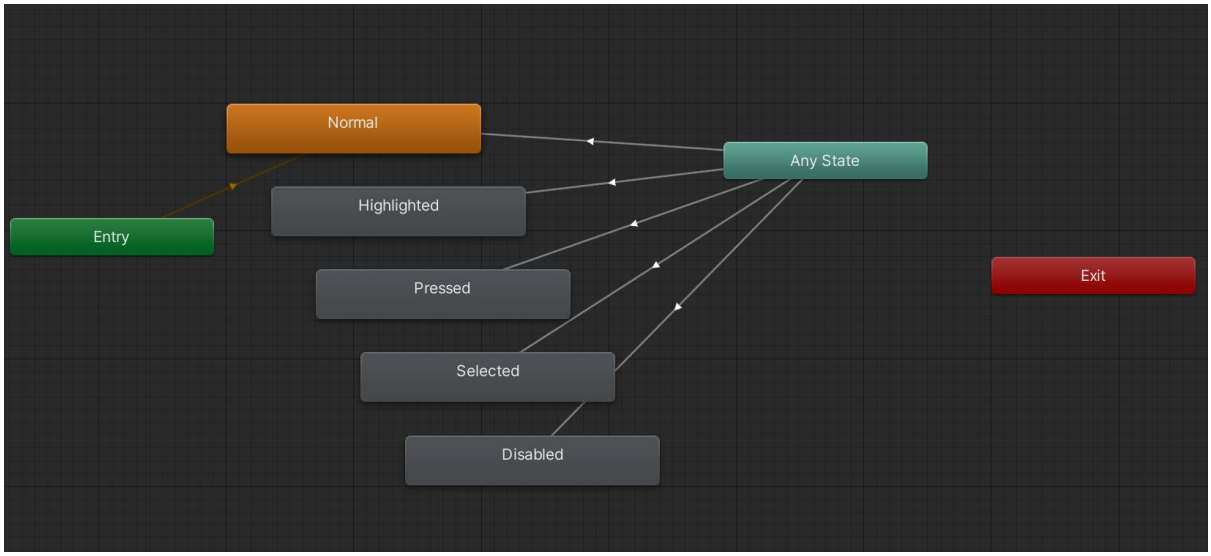


Fig. 18

Simply choose each state and set the desired scale for that state within the newly constructed animation controller. With a tween, Unity will automatically blend the states.

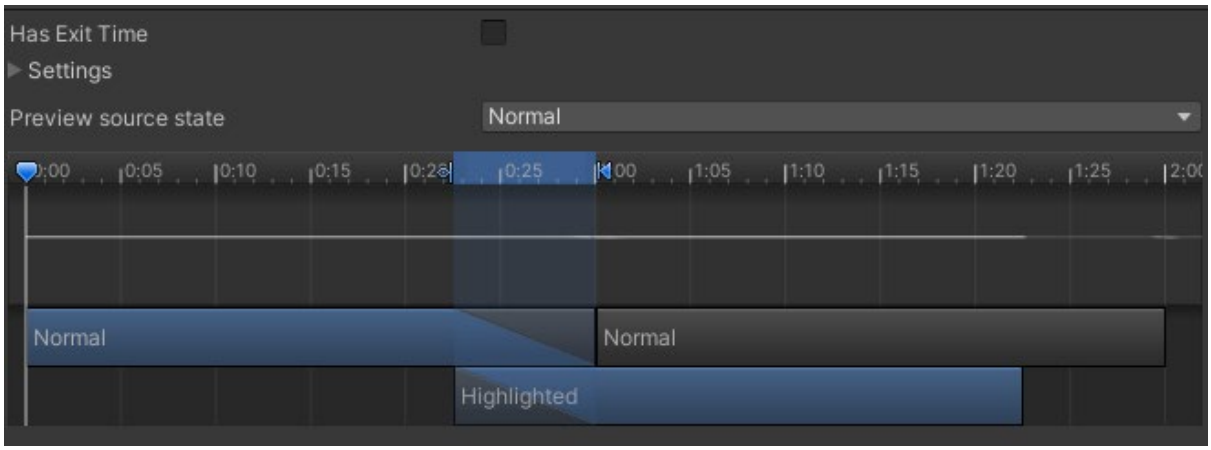


Fig. 19

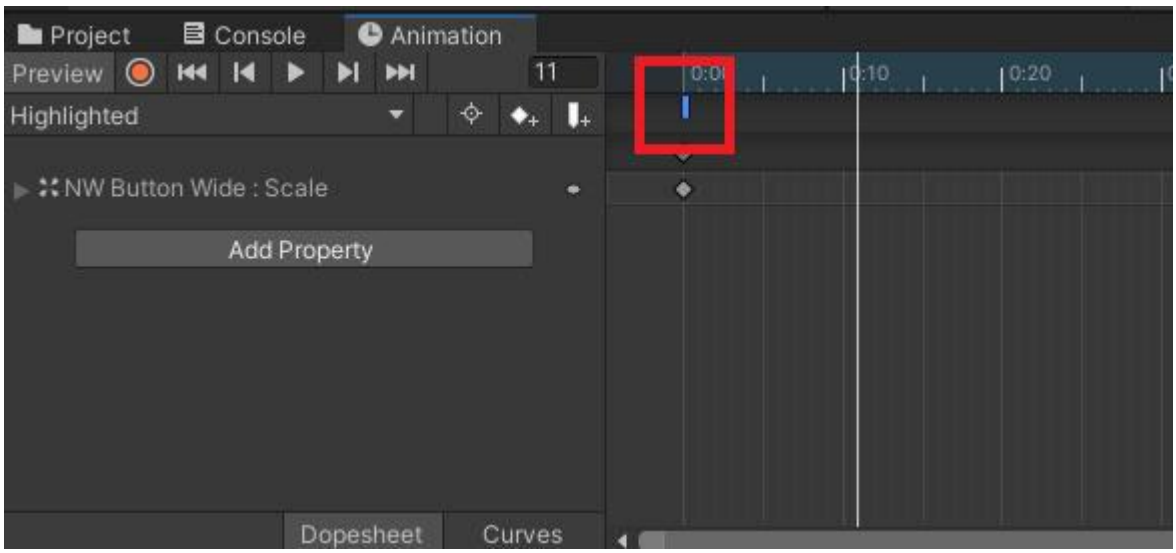
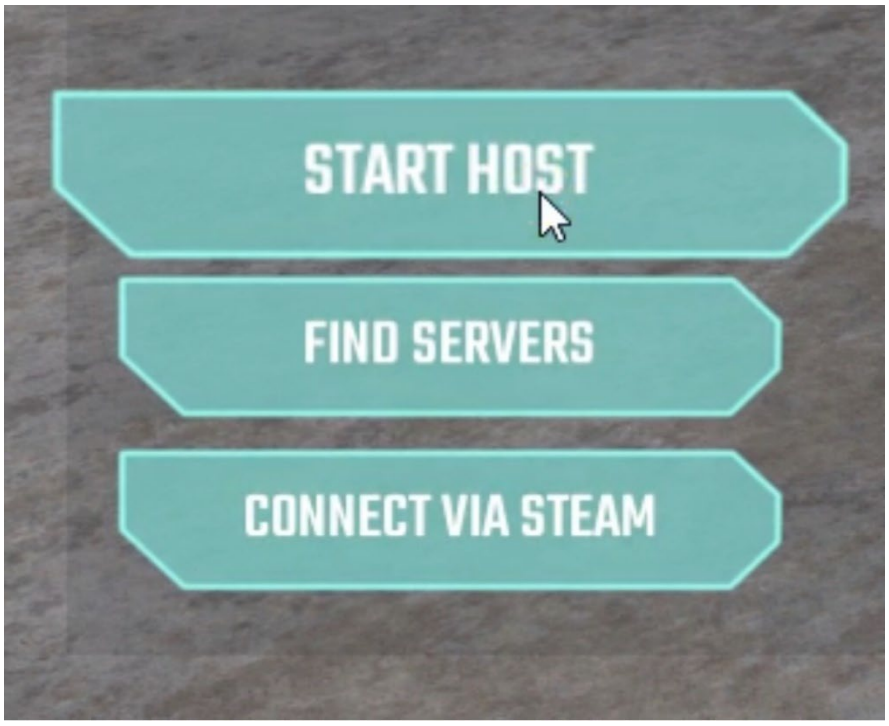


Fig. 20

To implement sound effects on hover and button press a function call is added to the beginning of the animation, this function call is responsible for playing the sound effect.



To view this animation follow THIS link

Fig. 21

After the completion of these steps a simple yet satisfying button is created, this button reacts when the mouse hovers over is as well as when the mouse interacts with it.

Enabling developers to easily access and use the custom buttons

The Unity game engine (2022) allows developer to expand upon and customize the engine through the use of Editor scripting (Unity – Editor, 2022) One example of the editor scripting attributes is Unitys “Menuitem” (Unity – MenuItem, 2022), the Menuitem can be used to expand upon The unity menus. The menuitem takes to arguments, the first is a string that is used to designate the desired path where the element should appear. To enable developers to access the custom buttons as easily as Unity’s default buttons an element will be added under the hierarchy menu, to achieve this the required path is “GameObject/Naval Warfare/UI/Buttons/Button”.

After menuitem has been created the functionality to instantiate the selected prefab implemented. This system should be easily expandable for future Thales navy warfare developers, therefore a system that requires manually referencing the asset path of each prefab wont work. Instead, a scriptable object (Unity – ScriptableObject, 2022) can be used to store references to multiple different prefabs. Unity’s AssetDatabase (Unity – AssetDatabase, 2022), can be used to load the Scriptable object. By doing this only one path reference is required to access all of the desired prefabs.

This implementation means that only one path reference is needed, however if said path reference would be unable to find the scriptable object this system would fail. To ensure that the object isn’t moved without changing the path reference Unity Editor.OnInspectorGUI (Unity – Editor

OnInspectorGUI, 2022) is used do display a message in the inspector informing future developers to change the path if they want to re organize the project files.

Using the scriptable object that contains the needed prefab references functionality can be added to the menuitems so that they can Instantiate their referenced prefabs.

Unity UI can only be viewed with a Unity canvas (Unity – Canvas, 2022) hence the developer should only be able to instantiate the UI prefabs through the menuitem if a canvas is active in the scene. To guarantee that there is a canvas in the scene the UI menuitem elements will only be visible if a canvas is the currently selected gameobject in the hierarchy. If a canvas gameobject is not selected a "Help Log" menuitem element will be visible in the hierarchy menu, selecting this option will log a warning informing the developer that they need to first select a canvas.

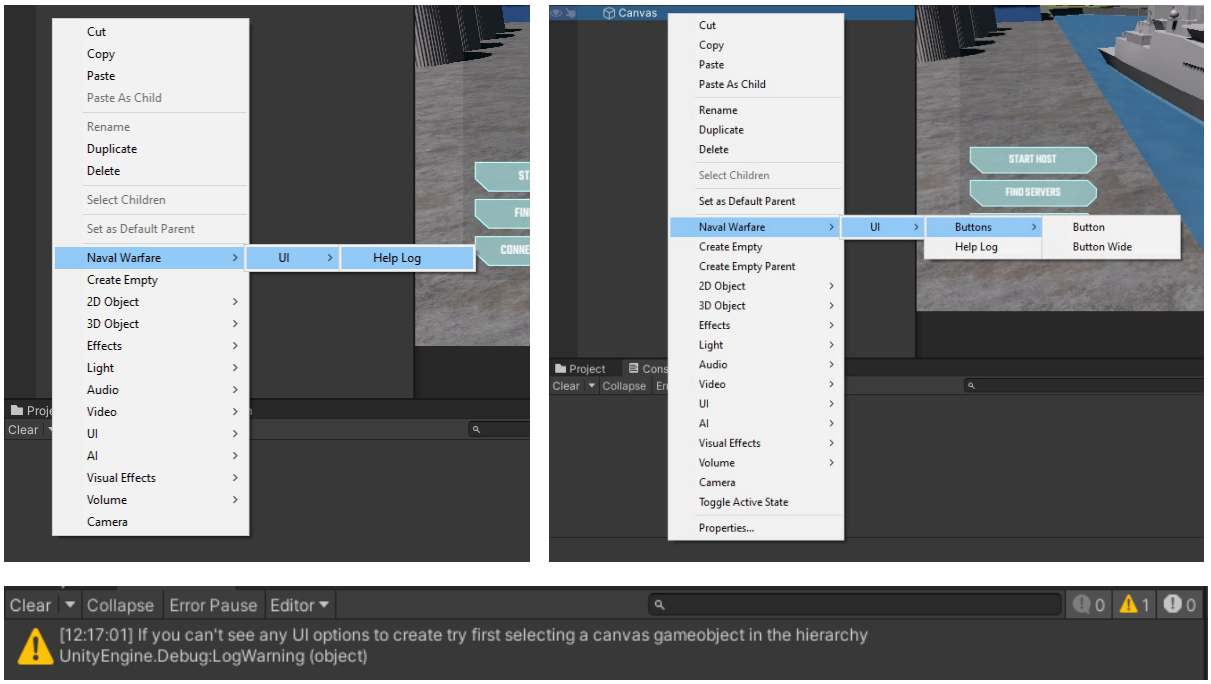


Fig. 22

Development of an asynchronous scene transition system

Requirements
Scene transitions should occur asynchronously.
The project should contain transition template(s).
Animated scene transitions should conclude automatically after the scene is loaded.

Thales' navy warfare game uses mirror (Mirror Networking, 2022) to handle the networking between users. Mirror networking handles scene transitions and networking communications between scenes, fortunately this means that all scene transitions present in the application happen asynchrony.

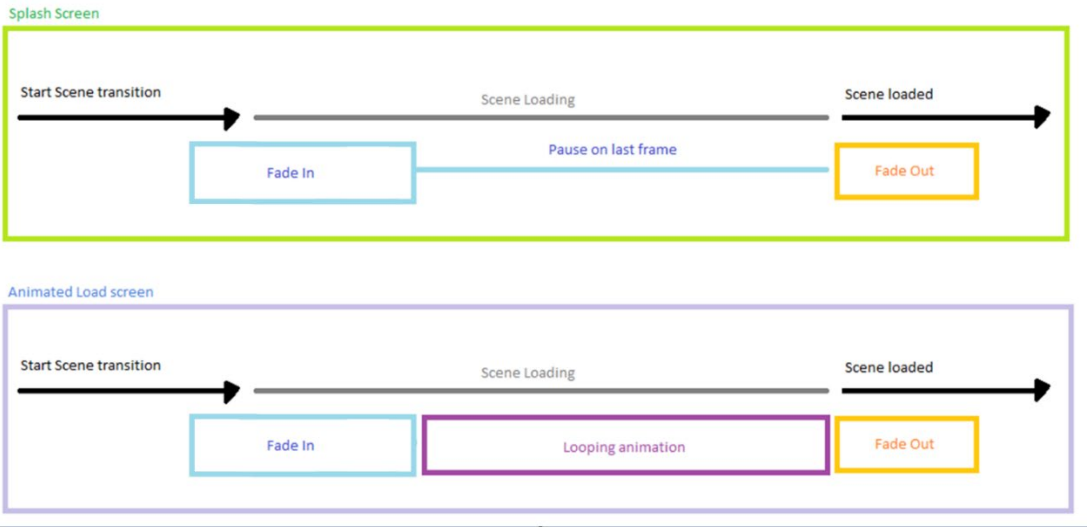


Fig. 23

```
public enum TransitionTypes
{
    SplashScreen, //an image fades in and out to blend the transition (used for short transitions)
    AnimatedLoadScreen //3 animations are played starting with a start animation then a looping mid animation, when scene loaded the 3rd closing animation is played (used for longer wait times to stimulate the user)
}
```

Fig. 24

Thales naval warfare game requires two different types of transitions, a “Splash screen” and an “Animated load screen”. Splash screens are used for short transitions between Small UI oriented scenes, the transition happens very quickly and is only used to blend the scene together. Splash screens start with a fade in animation that pauses on the final frame, when the target scene is loaded the splash screen will fade out. The animated load screen is designed to be used during longer scene transitions. Just like the splash screen it starts and ends with a fade in and out, however unlike splash screens the animated load screen continuously loops its second animation until the target scene has been loaded.

These two forms of scene transitions have been separated as they require different implementations. Although the implementations are different, they only differ slightly. The animated load screen and the splash screen use the same functionality for starting the transition and detecting when a scene has finished loading to end the transition. The animated load screen uses an animation controller to handle the animation loop after fade in.

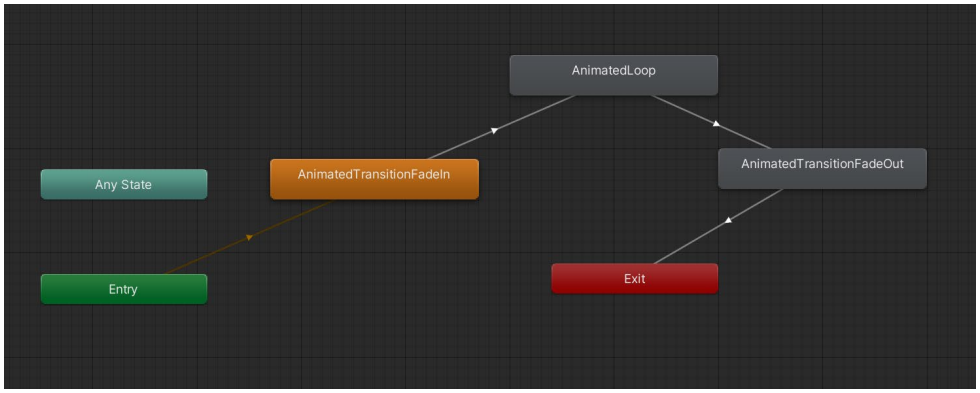


Fig. 25

As an ambiguous amount of scene transitions will be present in the application having an animation controller for each transition effect would be impractical and require a large amount of storage. To

remedy this issue an “Animator Override Controller” (Unity – AnimateOverrideController, 2022) can be used to mimic the functionality of the default animated load screen controller.

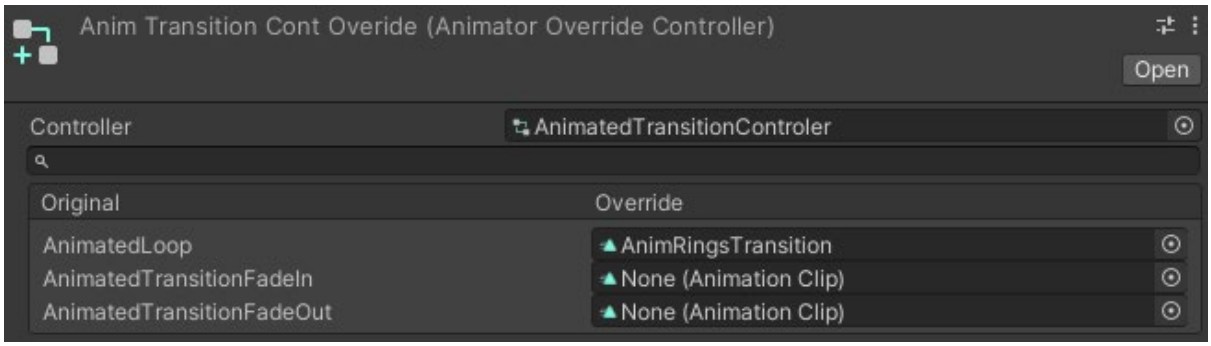


Fig. 26

Figure 26 shows that not all fields need to be overwritten, this means that a developer could quickly copy the template file and just make one singular animation to create a new scene transition.

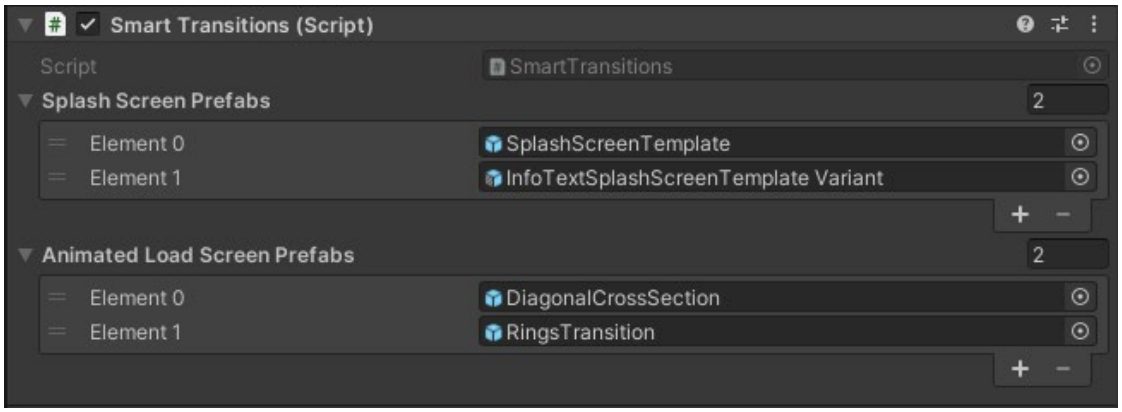


Fig. 27

Splash screens and animated load screens are referenced within the inspector. When starting a new transition, the developer can indicate the type of scene transition they want to play, as well as whether they want to choose a specific transition of that type or a random transition of that type.

Here are two example animated load screens [Transition One] [Transition Two]

Stating the scene transitions after a delay

A scene transition is divided into three animations for animated load screens and two animations for splash screens by the designed scene transition system. There is an animation at the beginning and ending of both sorts of scene transitions. The default templates start, and end animations are a fade in and out to allow compatibility with as many different scene transition designs as possible.

Future Thales naval warfare game developers that create their own scene transitions will need to expand upon the template file by adding their own image gameobjects (Unity – Image, 2022). The animations used for fading the transition in and out would require a reference to each of these newly implemented image gameobjects so that the alpha value of the image can be manipulated.

Fig. 28

A system that requires future developers to complete monotonous task such as manual referencing image gameobjects would not satisfy this papers research goal for ease of use tooling. The unity "Canvas Group" (Unity – CanvasGroup, 2022) component can act as a wrapper for all child image components, therefore the start and end fade animations can simply manipulate the canvas group component's alpha value and all newly added game objects will automatically be compatible with the animations.

Even though the template transitions use a fade in and out animation the animator override controller can override them with unique animations generated by future developers.

```
IEnumerator ExecuteAfterTime(float time)
{
    yield return new WaitForSeconds(time);

    Debug.Log("Animation wait time is now over");

    _manager.ServerChangeScene(AssetLoader.instance.mapData[0].mapScene);
}

0 references
private void ForScreenShotOnly()
{
    NWNetworkManager.singleton.gameObject.GetComponent<SmartTransitions>().StartTransition(SmartTransitions.TransitionTypes.AnimatedLoadScreen, -1);//-1 is used to pick a random animated transition
    StartCoroutine(ExecuteAfterTime(0.2f));
}
```

Fig. 29

The purpose of the first animation is to cover the screen in preparation for the transition. During scene change visual disturbance unpleasant for the user is commonly witnessed within the currently loaded scene. Visual disturbance could be anything from animated objects becoming static or shaders and lights no longer operating as intended. The visual disturbance starts on the frame that the asynchronous scene loading was invoked, this means that the first transition animation and the scene change should not happen at the same time. The animation needs time to fully cover the users screen, if the scene change was invoked at the same time as the animation the user would be able to see the visual disturbance for a moment before the animation covers the screen. Using a coroutine (Unity – Coroutines, 2022) it is posable to start the animation then wait a specified amount of time before invoking the scene change, this way the user won't witness any visual disturbances.

Detecting scene load completion and ending the scene transitions

Only one scene transition can be in affect at any given time. By subscribing to SceneManager.sceneLoaded its possible to detect when the asynchronous scene loading has finished, If a transition effect is currently active the result will trigger the ending animation to play. After the transition's ending animation has finished the transition effect gameobject will destroy itself.

```
void OnEnable()
{
    Debug.Log("OnEnable called");
    SceneManager.sceneLoaded += OnSceneLoaded;
}
```

Fig. 30

Decreasing the time required for the main game scene transition

The transition to the main game scene would take 50-60 seconds to complete, this lengthy delay is due to poor implementation of a pathfinding system created by a previous employee. Whenever the main game scene would be loaded the pathfinding system would generate a nav mesh consisting of thousands of nodes during run time. Remaking the pathfinding system in a way that didn't generate the nav mesh at run time and instead loaded a stored version is not within the scope of this research.

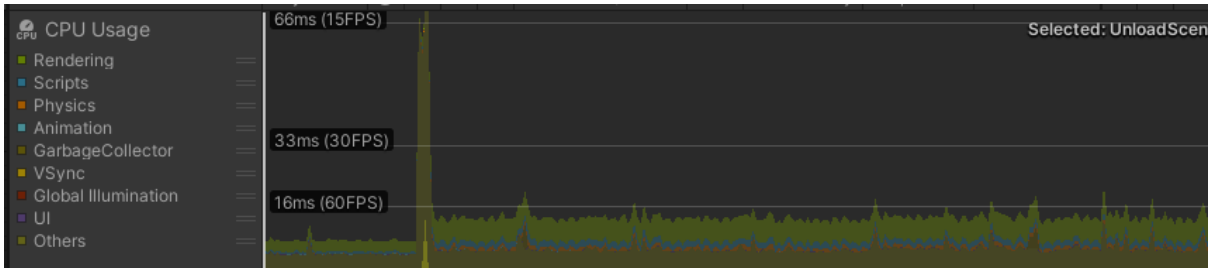


Fig. 31

When developing the animated transitions, the researcher became aware that the poor implementation did not just increase the time needed to load the scene, instead it was causing the application to become non-responsive. The frame rate would drop so low that the animated transition would freeze. The majority of the 50-60 second wait time was over the span of a few frames.

```
Thread calcTred;

Unity Message | 0 references
private void Start()
{
    navMeshReady = false;
    // Generate the Navmesh data
    NavMeshTriangulation navMeshRaw = NavMesh.CalculateTriangulation();

    ThreadStart threadStart = new ThreadStart(() => {
        ComputeNavMesh(navMeshRaw);
        navMeshReady = true;
    });
    calcTred = new Thread(threadStart);
    calcTred.Start();
}
```

Fig. 32

Something had to be done because the pathfinding system was freezing the transition system animations and causing a lengthy wait for the user. Remaking the pathfinding system was not within the scope of this study; therefore, the problematic pathfinding system functionality was moved to another thread (Microsoft – System.Threading namespace, 2022). Moving the pathfinding system away from the main thread saved a significant amount of time during the transition to the main scene. The wait time decreased from 50-60 seconds to 5-8 seconds. Moving the pathfinding system

off the main tread also resolved the issue of the transition animation freezing as the project was no longer hanging on a few frames.

Alternate tutorial implementation

The preliminary test revealed that the three cycling tutorial pages were an ineffective way to convey information to the user relevant to how to play the game. when communication with the other employees working on Thales' navy warfare game it was decided that a better implementation would be a toggleable HUD.

Fig. 33

Figure 33 shows how the tutorial HUD looks like in game, in the top left of the screen is the "how to play" button that enables and disables the heads-up display. The three old tutorial pages are still present in the project at the top of the toggleable HUD they can be open and closed freely.

Testing

Testing goals

Do test participant still believe that the application crashes during scene transitions?
How useful do the test participants believe the new in-game tutorial is?
Do test participant enjoy playing the game and navigating the user interface?
will the UI feel more responsive now with the implementation of button feedback?

Participants

To reduce bias as much as possible testing took place at Saxion University with a unique set of testers that did not take part in the first round of A/B testing. Testers once again entered the experience not knowing any prior information amount the game and would receive no guidance during the test. 18 participants took part in the “B” half of the A/B testing, Participants' ages ranged from early twenties to early thirties

Results

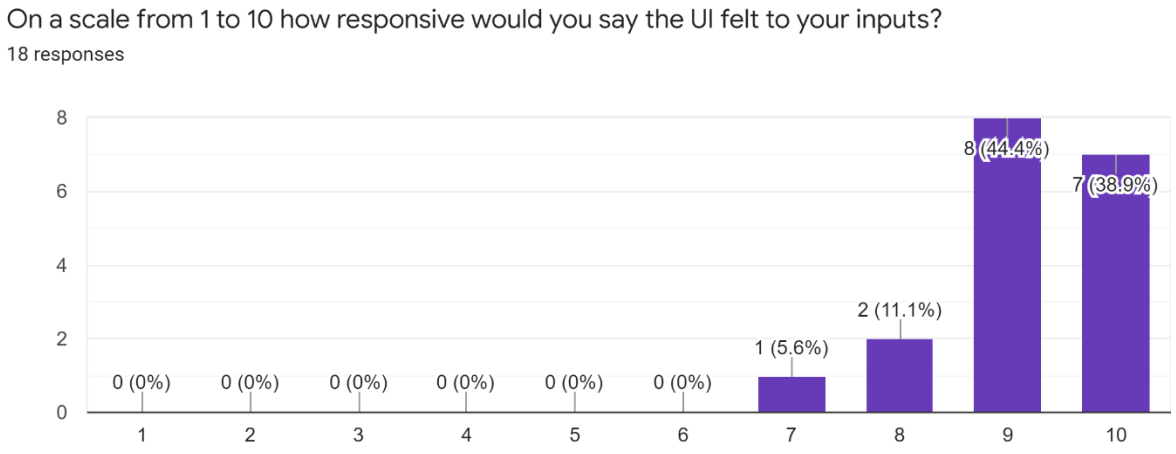


Fig. 34

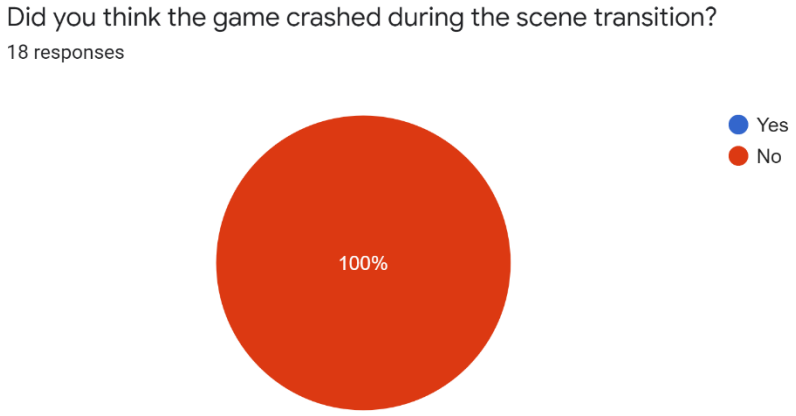


Fig. 35

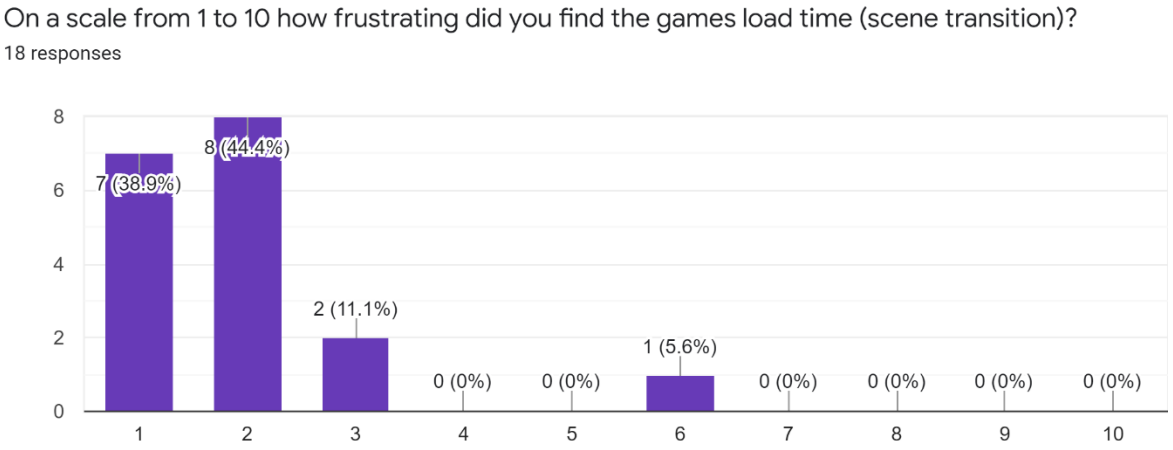


Fig. 36

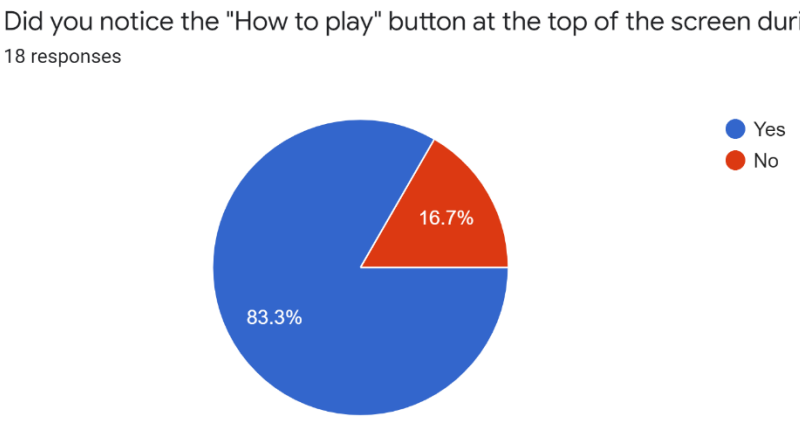


Fig. 37

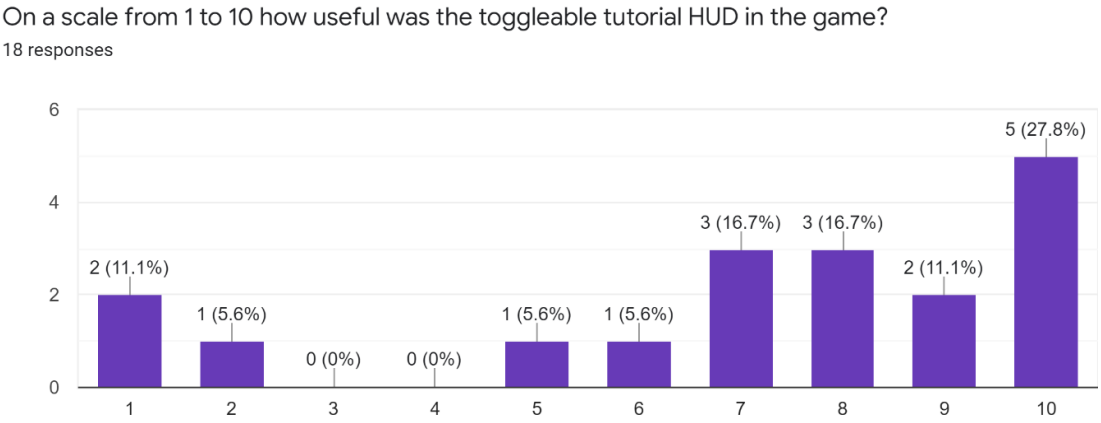


Fig. 38

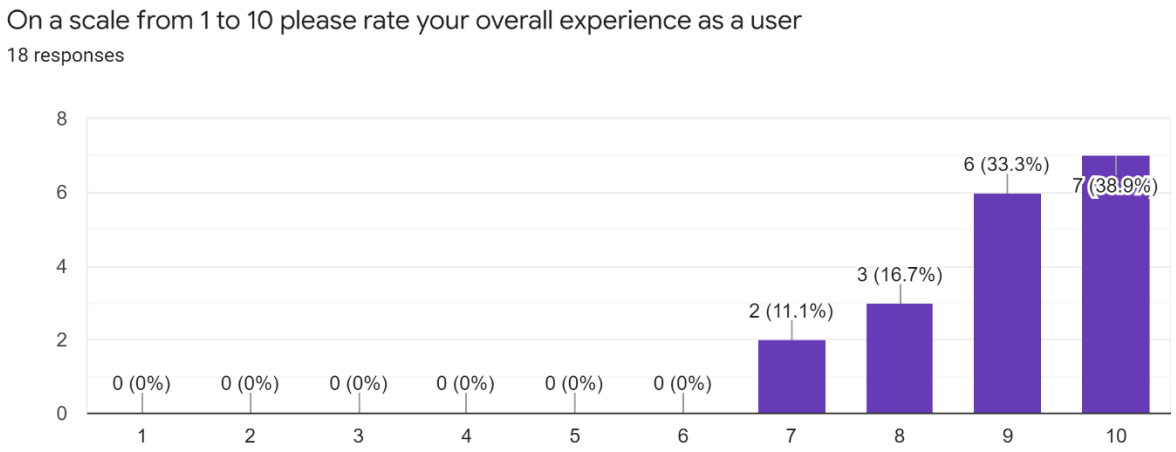


Fig. 39

Reflection of result data

Comparing the mean/average test results from the A/B testing				
Users rating for how responsive the UI inputs felt.	13.4	↑	16.5	+3.1
Percentage of users that thought the game crashed during the main scene transition.	80%	↓	0%	-80%
Users rating for how frustrated they were during the scene transitions.	11.4	↓	3.5	-7.9
Users rating for how useful they felt the in-game tutorial was.	5.7	↑	12.8	+7.1
The users rating for their overall experience as a user.	11.2	↑	16.2	+5

Fig. 40

Figure 40 above shows the comparison of the 2 rounds of A/B testing, test “A” the preliminary test is on the left-hand side and test “B” the after-implementation test is on the right of the indication arrows. The far right of Figure 40 displays the difference between the two means.

The data present in Figure 40 depicts that all target areas of this paper have succeeded in achieving improvement. The smallest change in rating average is for the responsiveness of the UI however, the “B” tests result for UI responsiveness is the highest statistic among all questions present in ether test. Perhaps the most impactful change between the two rounds of testing is the 80% drop assumed game crashes. The average overall experience of the user was reported to improve my 5 points further supporting the claim that this research was successful.

Conclusion

The goal of this research paper was to improve the user experience of Thales' navy warfare game by developing systems that addressed the game's most pressing UX issues. Preliminary testing prior this papers intervention suggested that the long static load screens were creating the most player dissatisfaction, however there were addition issues as well. Although the goal if this paper was to improve the user experience this research was not developed for the user, rather the systems this paper developed were for future Thales develops. Thales’ navy warfare game has been in

development for over four years, with the primary contributors being groups of interns and graduate students. As these interns and graduates have been using the progression of the game to develop their own systems for their academic studies the projects UX was largely neglected.

This paper mainly focused on the development of tooling for the creation of asynchronous scene transition effects, additionally this papers also worked to improve the user's satisfaction navigation the user interfaces. To improve the users experience navigating the UI dynamic buttons with responsive feedback were developed. Lastly this research worked to deprecate and replace an ineffective in game tutorial with an alternat interaction.

User experience is the most important part of any game or application, if a game isn't enjoyable to use no one will wish to use it. This research succeeded in improving the UX of Thales naval warfare game, in addition this research gave future develops the tools needed to continue best UX practises moving forward.

Never neglect the user's experience.

Sources

Alaniz, G. (2020). 3 Simple tips on UX button design. https://www.kalamuna.com/blog/3-simple-tips-ux-button-design#:~:text=Size,tactile%20sense%20for%20the%20user

Card, S., Robertson, G., & Mackinlay, J. (1991). The information visualizer, an information workspace. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '91). Association for Computing Machinery, New York, NY, USA, 181–186. https://doi.org/10.1145/108844.108874

Clash Royale (2022). https://clashroyale.com/

De la Riva, M. (2021). Why consistency is so incredibly important in UI design. https://careerfoundry.com/en/blog/ui-design/the-importance-of-consistency-in-ui-design/#:~:text=Consistency%20in%20UI%20design%20is,control%2C%20familiarity%2C%20and%20reliability

Friis Dam, R. (2017). What is the next big thing?. Interaction Design Foundation https://www.interaction-design.org/literature/article/what-is-the-next-big-thing#:~:text=You%20could%20always%20take%20Blake,you're%20stuck%20for%20ideas.

Jonasson, M. & Purho, P. (2015). Juice it or Lose it. https://www.gdcvault.com/play/1016487/Juice-It-or-Lose

Kurusathianpong, P., & Tangmanee, C. (2018). Comparison of Perceived Waiting Time Between Two Lengths of Progress Indicator and Two Styles of Graphics Animation With Perceived Uncertainty as a Covariate. 2018 Seventh ICT International Student Project Conference (ICT-ISPC), 1-6.

Liikanen, L. & Gomez, P. (2013). Designing interactive systems for the experience of time. In Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces (DPPI '13). Association for Computing Machinery, New York, NY, USA, 146–155. https://doi.org/10.1145/2513506.2513522

Limelight (2019). The state of online gaming – 2019. https://www.limelight.com/resources/white-paper/state-of-online-gaming-2019/

Microsoft (2022). System.Threading Namespace https://docs.microsoft.com/en-us/dotnet/api/system.threading?view=net-6.0

Mittel, A. (2018). UX vs UI – similarities and differences. https://blog.prototypr.io/ux-vs-ui-similarity-differences-837775584cd8

Mirror Networking (2022). Mirror Networking: Open source networking for Unity. https://mirror-networking.com/

Pereira, V. (no date). What's your input? https://www.packt.com/whats-your-input/

Söderström, U., Bååth, M., & Mejtoft, T. (2018). The Users' Time Perception: The effect of various animation speeds on loading screens. In Proceedings of the 36th European Conference on Cognitive Ergonomics (ECCE'18). Association for Computing Machinery, New York, NY, USA, Article 21, 1–4. https://doi.org/10.1145/3232078.3232092

Steam (2022). https://store.steampowered.com/

Unity (2022). https://unity.com/

Unity – AnimateOverrideController (2022). https://docs.unity3d.com/ScriptReference/AnimatorOverrideController.html

Unity – AssetDatabase (2022). https://docs.unity3d.com/ScriptReference/AssetDatabase.LoadAssetAtPath.html

Unity - Canvas (2022). https://docs.unity3d.com/ScriptReference/Canvas.html

Unity - CanvasGroup (2022). https://docs.unity3d.com/ScriptReference/CanvasGroup.html

Unity – Coroutines (2022). https://docs.unity3d.com/Manual/Coroutines.html

Unity – Editor (2022). https://docs.unity3d.com/ScriptReference/Editor.html

Unity - Editor OnInspectorGUI (2022). https://docs.unity3d.com/ScriptReference/Editor.OnInspectorGUI.html

Unity – Image (2022). https://docs.unity3d.com/2018.3/Documentation/ScriptReference/UI.Image.html

Unity - MenuItem (2022). https://docs.unity3d.com/ScriptReference/MenuItem.html

Unity – ScriptableObject (2022). https://docs.unity3d.com/ScriptReference/ScriptableObject.html

Vierordt, K. (1868/2020). Der Zeitsinn nach Versuchen, Frankfurt und Maim.