Porting Cocos Creator web games to Steam

By Eele Roet
Amsterdam University of Applied Sciences, Game Development

Company
Wanted 5 games

Counselors
Robin Maree
Remco van Swieten

10-4-2024

Student
Name Eele Roet
Student number 500795948
Phone number +31 6 19885650

City
Haarlem

Date
10-4-2024

Institute
School Amsterdam University of Applied Sciences
Program Information and Communications Technology (HBO-ICT)
Field Game Development

Company
Name Wanted 5 Games
Address Stationsplein 116, 2011 LN Haarlem

Counselors
Thesis supervisor Frank Aldershof
Company supervisor Robin Maree

Assignment period
September 4th 2023 until February 2nd 2024

Abstract

This thesis will document how to build and release Cocos Creator projects on the Steam store. This topic is of value to Wanted 5 Games due to the recent release of the game Stratego Online in November 2023. Wanted 5 Games wants to reach a broader audience by porting the game to new platforms. This presents challenges since they do not have experience with uploading games to these platforms.

The main research question surrounding this topic is: "*What is the optimal platform to release Stratego Online on, and what is the best method for porting the current Cocos Creator project to this platform?*"

This thesis includes desk research into the game distribution platform that offers the most value to Wanted 5 Games regarding Stratego Online. The results of this research will then form the basis of the practical implementation in the later chapters. These later chapters will document a possible workflow for making and releasing builds during production of the game.

The desk research will set up platform metrics, collect data points from a range of platforms, and use data analysis to conclude that Steam is the platform best suited for Stratego Online.

The introduction of the field research, in the form of a practical implementation, describes the requirements of releasing a game on Steam. This only includes the minimum requirements set by Valve to release the game in the store. The most important requirements here are the need for a standalone PC application and implementation of Steams API.

The best method of porting Cocos Creator projects to Steam was found to be the Electron framework, which wraps the base JavaScript game in a standalone application. This conclusion was reached after benchmark testing the available options.

Finally, an implementation of the Steamwork.js module was used to create a setup for calling Steams API functionality from inside the Cocos Creator project.

The research has identified a low-risk and efficient approach for porting Stratego Online to Steam. The existing web build of Stratego can seamlessly be adapted for standalone PC use with Electron. This allows for its deployment onto the Steam platform. This approach minimizes development effort. The recommendation is that Wanted 5 Games continues this implementation using the knowledge in this thesis.

Glossary

Term	Definition
Cocos Creator	A game development engine and editor, supporting 2D and 3D games across multiple platforms through its support for JavaScript and TypeScript programming languages.
Steam	A digital distribution platform developed by Vale Corporation, used for purchasing, downloading, and managing video games and related content.
Steam store	The Steam store acts as the digital marketplace of the Steam platform. Users can browse the store to buy products.
Stratego Online	The latest online version of the board game Stratego, owned by Royal Jumbo BV and developed by Wanted 5 Games.
HTML 5 games	Browser-based video games developed using: HTML 5, CSS, and JavaScript, allowing them to be played directly in the browser without additional installation.
Steamworks API	Tools and services provided by Valve Corporation for game developers to integrate various features, such as multiplayer matchmaking and achievements, into their Steam-enabled games.
Web portals	Online platforms that host a variety of browser-based games, offering users access to a diverse selection of titles across different genres.
Steamworks	Tools and services that help game developers and publishers build their games and get the most out of game distribution through Steam.
JSB	JavaScript binding (JSB) is used by Cocos Creator to bridge the gap between JavaScript code (used for game logic) and native code (C++ or other languages used for performance-critical tasks).
Electron	An open-source framework that allows for the creation of cross-platform desktop applications using web technologies like HTML, CSS, and JavaScript.
Greenworks	A npm package that acts as an interface between JavaScript code and the Steamworks API. Currently this package is deprecated.
steamworks.js	A npm package that acts as an interface between JavaScript code and the Steamworks API.
Chromium	Chromium is an open-source web browser project maintained by Google, serving as the foundation for various web browsers such as Google Chrome, Microsoft Edge, and others. It provides the core engine for rendering web pages and executing web applications.

Introduction

This thesis will document how to build and release Cocos Creator projects on the Steam store. This topic is of value to Wanted 5 Games due to the recent release of Stratego Online in November 2023. Wanted 5 Games wants to reach a broader audience by porting the game to new platforms. This presents challenges since they do not have experience with uploading games to these platforms.

This thesis includes desk research into the game distribution platform that offers the most value to Wanted 5 Games regarding Stratego Online. The results of this research will then form the basis of the practical implementation in the later chapters. These later chapters will document a possible workflow for making and releasing builds during production of the game.

Reading Guide

The first chapter provides context to the assignment by providing information about the shareholders and surrounding circumstances.

The second chapter will set up platform metrics, collect data points from a range of platforms, and analyse what platform is best suited for Stratego Online. This will be achieved by performing data collection and product comparison.

The third chapter describes the requirements of releasing a game on Steam. This will only include the minimum requirement set by Valve to release the game in the store, not extra requirements for ensuring the success of a game on Steam.

The fourth chapter will determine the best methods to practically implement a porting workflow. These methods will be tested and compared using benchmark testing.

The fifth and final chapter will provide an example of implementing the requirements (from chapter three) that are not yet included in the workflow (from chapter four).

At the end of the thesis will be a recommendation to Wanted 5 Games about implementing a base Steam port workflow. Future recommended features are also included.

1 Context

1.1 Wanted 5 Games as a company

Wanted 5 Games is a game development studio based in Haarlem. They specialize in HTML5 games in order to reach the broadest audience with the natural benefits that web games offer. Being a smaller studio means that employees have a tight connection and are aligned on the goals of both the projects and the company. (*Wanted 5 Games - HTML5 Game Studio*, 2021)

Wanted 5 Games has grown by producing high quality web games in genres like Match 3. They have worked on some own IP and game concepts, but have also worked with brands like: Exploding kittens, Smiley and TikTok to realize existing concepts.

With design, art, and development, the company can take on the whole process of making games in-house.

1.2 Royal Jumbo BV and Stratego

Royal Jumbo BV is a leader in the realm of board games and puzzles. With a heritage dating back to 1853, Royal Jumbo BV has captivated audiences worldwide with a diverse portfolio of games and puzzles. (*About Jumbo - Jumbo*, 2022)

A flagship product in the lineup of games is Stratego, a classic strategy board game that has stood the test of time since its introduction in the 1940s. Described as a combination of: chess, checkers, and poker, Stratego is the ultimate battle of wits and strategy. With its rich history and enduring popularity, Stratego has become a beloved staple in the world of board gaming. (*Stratego - Jumbo*, 2024)

Jumbo has innovated with the Stratego IP over the years to keep players engaged and reach new audiences. Titles such as “Stratego Junior” and “Spies & Lies- A Stratego story” put a spin on the original to target different demographics. (*Stratego - Jumbo*, 2024)

Stratego Online is the newest online version of Stratego since its release in November 2023. This game has been developed by Wanted 5 Games and is currently still receiving updates. The game has been developed as an HTML 5 game in Cocos Creator. Mobile builds for IOS and Android have been added to the workflow from the start of the project to support 3 different platforms.

1.3 The assignment: Cause and value of porting the game to reach a bigger audience

Both Wanted 5 Games and Jumbo have stakes in the project and benefit from Stratego Online being successful. The monetization model in the game benefits from a large number of players that are willing to spend money inside the game.

Besides making new content to draw players in, they want to reach new player bases. Players on platforms that usually do not play web games.

The game is currently already released to two mobile platforms (App store and the Google Play Store) as well as on the play.stratego.com website. The game is performing well on the mobile platforms, drawing in hundreds of new players per day. This shows that an expansion in platforms is beneficial to the size of the player base.

Since Wanted 5 Games is a smaller studio with limited resources, an important part of the assignment is that the porting process should be relatively low maintenance. Parts of the process that can be automated should be automated. This results in the best method of porting being one that adds value with a low amount of additional effort.

That is why the assignment at the center of this thesis is “Efficiently port Stratego Online to a beneficial platform that is new to Wanted 5 Games.”

1.4 Problem definition

There are a couple of reasons that this assignment requires thorough research. Firstly, an informed decision needs to be made about what platform would be the optimal choice for expanding the player base.
In this case, optimal is defined as a platform that provides the game with a large number of dedicated players that fit the target demographic of Stratego Online. The target demographic will be further defined in chapter 2.

New platforms can also possibly bring new problems that the team at Wanted 5 games does not have the knowledge to fix at the moment. These problems might involve:

- The game running on unfamiliar hardware.
- A lack of knowledge about the optimal platforms to release games on.
- Platforms might require platform-specific features to be implemented.
- Building the game to the correct format to launch on a specific platform.

These issues also play a role in integrating a new platform into the current Continuous Integration (CI) that is set up for making and uploading builds. Knowing the limitations and requirements of the platforms being used is essential for setting up a smooth CI.

Experience and knowledge about the points above are crucial to choose a new, optimal platform to add to the current workflow. That is what this thesis aims to offer Wanted 5 Games.

1.4.1 Research question

To keep the research focused, a main research question should be defined. This should include the assignment and when the question is answered, the unknown information in the problem definition should be clarified. Thus, the question is phrased as follows:

“*What is the optimal platform to release Stratego Online on, and what is the best method for porting the current Cocos Creator project to this platform?*”

1.4.2 Sub-questions

To answer the main research question in a coherent manner, we define sub-questions. These sub-questions divide the research into relevant parts. Starting with a broad analysis of the available platforms, then looking into the platform specific requirements of the selected platform, and finishing with research about the best methods to port the current Cocos Creator project to the selected platform.

The sub-questions read as follows:

“*What platform offers Stratego Online dedicated users while offering realistic porting options?*”

“*What are the required components when porting a game to Steam?*”

“*What are the options to port a Cocos Creator project to Steam?*”

“*How to integrate the ‘Steamworks API’ and ‘Steam Store page’ platform requirements?*”

It is clear from these questions that the conclusion of the first sub-question is that Steam is the optimal platform for Stratego Online. However, it should be noted that these sub-questions were

defined more generally at the start of the research. Only after concluding the first sub-question were terms like: “the optimal platform” or “the selected platform” replaced by “Steam” for clarity.

1.5 Methodology

Most of the research in this thesis was conducted with the guide of “The ICT Research Methods” by Bonestroo et al. (2018). This collection of research methods for a framework that supports solid research. The framework has been used in a number of other thesis papers in the field of ICT, like the thesis papers of Van Leeuwen (2011), and Van Turnhout et al. (2018).

Below, the reader can find the research methods that were used during the literature and field research, and the purpose those methods serve within the research. Bold phrases refer to the research methods from Bonestroo et al. (2018).

Clarifying focus and scope of the research

To gain a baseline understanding of the assignment,
interviews were conducted with stakeholders about the problems that need to be solved. This information was used to write the context in the first chapter.

Solidifying the literary research (gathering context for the problem)

More specific knowledge of the context surrounding the porting of HTML5 games is crucial for setting up literary research.
Available product analysis was performed of the available games distribution platforms to create an overview of the possible destinations for Stratego Online.

Brainstorm sessions then resulted in proposed metrics to measure platform value, setting up the possibility for objective comparison.
Interviews with Wanted 5 Games employees were conducted to align the metrics with the company values.

Performing the literary research

Data points of all metrics were then collected for each platform, creating a dataset enabling for **data analytics** to support the choice of platform.

Solidifying the field research (gather context for the problem)

The result of the literary research then formed the basis of the field research. The goal of this research was to gain concrete knowledge about how the port should function, and how this can be achieved. First, the current porting workflows within Wanted 5 Games were
observed. This included processes like making builds for mobile devices.

HTML5 porting Workflows of other developers were also analysed during the
community research in order to find recommendations of developers outside the company.

Validating the result of the field research

Before starting the technical implementation, a plan for validating the result should be devised. **Document analysis** of system requirements resulted in an overview of what Steam requires builds of games on their platform to adhere to. The result of the field research can then be compared to these requirements to validate the success of the final implementation.

Choosing a fitting technology

Expanding on the observations made to solidify the field research, a better understanding of the requirements of the final system was needed to decide on the best methods of porting.

User requirements were **explored** that the ports should adhere to. These requirements were set up with the CTO of Wanted 5 Games.

Available product analysis could then plot out promising, existing porting solutions that could be implemented.

Prototyping minimal ports using these solutions then offered the choice of comparison in the form of **benchmark testing** to conclude the most promising options to support the choice of technology.

Realising all platform requirements

To expand the minimal port discussed earlier into the fully capable Steam port, final platform requirements needed to be implemented.

Community research helped to find resources and offered solutions to issues during implementation. **Prototyping** then further expanded the porting system to include the final requirements.
Finally, **benchmark testing** validated the resulting system against platform requirements that were set at the “Validating the result of the field research” step.

---

2 The most advantageous platform for Stratego Online

This chapter contains a broad overview of the landscape of gaming distribution platforms.

Looking into a good sample of popular distribution platforms on different devices, data will be collected on a variety of metrics.

This data can then be used in data analysis to assess what platform offers Wanted 5 Games the most value for any particular game, which in this case will be the platform that best suits Stratego Online.

This will be researched in five sections. The first section looks into the available platforms. The second section will define metrics to measure the value of each platform. The third section will gather data about the selected platforms. Then, the fourth section explains the comparison of the data. Finally, in the fifth section will be the conclusions on what platforms offer the most value to Stratego Online.

2.1 Available platforms

Starting off with the list of 3 platforms Wanted 5 Games currently ship new Stratego builds to:

- Web, on the official play.stratego.com website
- Mobile, on both the iOS App Store and Android play store

Wanted 5 Games has had experience developing for these platforms in the past. “Cook & Match: Sara’s adventure” and “love tester” were both games that had web and mobile platforms in their workflow. This was especially useful for Cook & Match, considering the popularity of “match 3” games on both HTML 5 games sites and on mobile devices.

For Stratego Online they would like to reach an even broader audience of dedicated players. That is why this chapter will also explore platforms they do not have experience with, on devices Wanted 5 Games have not released games on yet.

To find suitable platforms, exploratory research was performed around the most popular game distributors per device. Focussing on the most popular gaming devices:

- PC
- Web (can be both on PC or mobile)
- Mobile
- Consoles

For each class of devices, the biggest platforms for those devices were noted. While this list is not exhaustive, it is comprehensive enough to form a strong foundation for data collection and analysis. The chosen platforms reflect market leaders, providing plenty of data for making informed decisions regarding the optimal destination for Stratego Online.

PC (Blancaflor & Miguel, 2022)	Steam Epic Games GOG.com
Mobile (Blancaflor & Miguel, 2022)	App store Play store

Web (Blancaflor & Miguel, 2022)	Kongregate Newgrounds Armor Games Poki Miniclip CrazyGames SPIL Games portals (Azerion) Admeen (Azerion)
Console (Obias, 2024)	PlayStation Store Xbox Store Nintendo eShop

In the next section, the process of collecting relevant data for each platform will be described. This will consist of: looking at user demographics, and technical considerations. Like this, a confident, strategic decision can be made about the platforms that align with Stratego Online.

2.2 Platform aspects that offer value to Wanted 5 Games

To do a good analysis of the value the different platforms offer to Stratego, more information about the platforms needs to be gathered. To keep this information the same for all platforms, metrics have been set up that can be measured and recorded.

2.2.1 Proposed metrics

These metrics describe different features that the platforms offer. Information about the users of each platform and monetary information such as revenue shares between developers and the platform.

1. Size of monthly user base. (Added after interviews)
2. Global presence/international reach.
3. Users' age.
4. Users' location. (Removed after interviews)
5. User engagement and retention.
6. Users' preference for game genre. (Removed after interviews)
7. Other games similar to ours should perform well on the platform.
8. Use of pre- and mid-roll ads on the platform.
9. Freedom to use our own payment system.
10. Percentages on the revenue share deal.
11. Possibilities of our game being featured on homepages and in favorites.
12. The channels that users on the platform find the game through.
13. The amount of effort to be expended to build a version of the game that can be hosted on the platform.
14. The risk of platform specific bugs.
15. Users can see what games that their friends are playing.
16. Users can log into games using the platform's user account.

This list was first set up following a brainstorm session that included logically writing down what metrics were assumed to offer value. Then interviews were conducted with multiple people from different disciplines at Wanted 5 Games to verify the value of the metrics.

2.2.2 Validation interviews

During the validation of the metrics interviews were conducted with:

- Jord, CEO at Wanted 5 Games. Jord knows a lot about the business side of game development. His interview will be providing information about the monetary and marketing aspects of the metrics.
- Weikang, Lead Developer at Wanted 5 Games. Weikang has great technical skills and foresight. He has also done a number of web portal SDK implementation, so his interview will be interesting regarding the development effort and costs.
- Charlie, my co-developer in the Stratego team. Charlie has a big and varied skill set for making games, so he will be asked for his general thought on the metrics.

The three were asked for short interviews during the workday where we went through the metrics that were proposed. The thoughts behind them were explained and questions were asked about how much sense the current metrics made and if any important metrics were missing. The paraphrased responses can be read below.

Jord:
The list of metrics looked good, but the most influential value measure was missing: the platform's reach. The amount of players on the platform is essential when assessing the value of releasing to that platform, and this metric was promptly added to the list.

Weikang:
Mostly experienced in releasing for web and mobile platforms.
Biggest issues were for IOS on both web and mobile, IOS had weird OS specific issues like rendering and audio bugs that would only occur on those devices.
The standard flow of implementing a specific web SDK is not a big development cost or effort. It requires a lot of testing to see whether the implementation is correct, but this is not a development heavy task.

Charlie:
When assessing value in a new platform, Charlie would look at games on the platform that are similar to the game we are making. It is smart to see if there are a lot of other games with similar gameplay or within the same genre. Then it's important to review how well those games are performing.

This feedback sketched a good overview of any missing metrics and what metrics are valued by Wanted 5 games.
Another point that was brought up during the interviews was that 2 metrics appeared superfluous, metrics number 4 and 6. These were in my opinion too similar to numbers 2 and 7. The interviewees agreed, and they were removed from the list.

2.2.3 Final list of metrics

This resulted in a list of 14 testable metrics that will provide an idea of the value that each platform offers. The 14 aspects are divided into 7 categories to improve visibility.

Platform Reach	1. Size of monthly user base. 2. Global presence/international reach.
Audience Demographics	3. Users' age. 4. User engagement and retention.

	5. Other games similar to ours should perform well on the platform.
Revenue Share and Monetization	6. Use of pre- and mid-roll ads on the platform. 7. Freedom to use our own payment system. 8. Percentages on the revenue share deal.
Marketing and Promotion support	9. Possibilities of our game being featured on homepages and in favorites.
Distribution Channels	10. The channels that users on the platform find the game through.
Development effort and costs	11. The amount of effort to be expended to build a version of the game that can be hosted on the platform. 12. The risk of platform specific bugs.
Community and Social Features	13. Users can see what games that their friends are playing. 14. Users can log into games using the platform's user account.

2.3 Data collection

2.3.1 Acquiring data from trusted sources.

Since there are a lot of different metrics, multiple data collection methods were used to gain the information in this chapter.

- Using a direct source.
- Employee correspondence.
- Statista.com.
- Similarweb extension.
- Web articles without direct sources.
- My best judgement.

During the research it was attempted to always provide information from the most credible source possible. The value of a source was judged through "Trickle down reliability". This concept is based on the phenomenon where data naturally becomes more and more unreliable as it is passed down.

As the information changes hands it is guaranteed to get distorted in some way. This is why a direct source is always the most reliable, and human recollection is generally the least reliable.

With this concept in mind there was always an attempt at finding a direct source of information (e.g., the official website run by the organization in question), then some trusted information aggregator like statista.com, then online news articles. Only when no other credible sources were found, was personal judgement used to make observations.

Sources are cited directly along the data in Appendix A. If no source is present, the data was gathered by personal observations. This data has the lowest guarantee of being accurate, but will still provide a general impression of those metrics to come to a conclusion at the end of the chapter.

2.3.2 Data overview

The raw data collected is available in Appendix A.

Notable results:

While collecting the data, a couple of interesting data point popped up. These are described below in no particular order.

It was assumed for turn based strategy games to be popular on Epic games, but the genre was not promoted on the front page and popular games within the genre were not present on the platform.

The big web portals get a lot more monthly visits than expected, with the largest one averaging around 100 million visits per month, and the small platforms still around 3 million visits per month.

There is a big difference in revenue share percentages between platforms. The industry standard has been a 30/70 split for platform and developer, respectively. Epic games and the Play store are now offering 12/88 and 15/85 respectively. (Sinclair, 2018)

The difference in development costs for different ports is very big. According to the lead developer at Wanted 5 Games, web ports take about a week of implementing and testing. Console ports may require completely rewriting the game in a new engine for new hardware.

2.4 What platforms offer the most value?

The analysis was intentionally split into 2 parts: general data collection and this part, the specific analysis of the best platform for Stratego.
This way the data collected in the first part will be valuable for Wanted 5 Games until the data will be outdated.

This approach also makes the second part, analysis of the best platform for a specific game, easier. Now only the ideal metric values results need to be specified and these can be compared with the data to find the best fitting platform.

But what are the desirable metrics for Stratego Online?

There are 14 metrics that describe the most favorable conditions on the ideal platform. Overall, the port should be low in development costs, accessible and visible to users, and be as profitable as possible.

For 4 of these metrics It is important to have a good idea of the target audience for our game, these are metrics: 3, 4, 5 and 10. The Creative Director was asked for information about the average Stratego player that we want to reach. This resulted in a description of players that have a certain type of engagement with the genre that Stratego offers: the strategy board game genre. Players with this type of engagement enjoy long games, and have long retention. They are eager to climb the ranks and compete.

This research resulted in the data that describes the hypothetical, optimal platform to launch Stratego Online on. These optimal results are available in Appendix B.

Now the two components that are needed to compare all the platforms are known, and the best platform for Stratego Online can be selected. Each platform's data from Appendix A can be compared with the optimal values in Appendix B. How well each platform's metrics match the optimal values will be recorded, after which conclusions can be drawn about the value of the platforms.

The result of this comparison is available in Appendix C.

The results in Appendix C are colored from green to red based on how well each platform fits our game within that metric. The coloring is based on the relative difference between the highest and lowest amount of fit in that specific metric. The coloring is only to quickly assess what platforms have positive and negative matches to the optimal data. Conclusions should be made based on the information the text provides, not the coloring.

2.5 Conclusions

2.5.1 Mobile platforms

Mobile is a very solid, all round fit for Stratego. Mobile games are very accessible since most people have a phone that can run games. This also explains the big user bases that both Android and IOS offer. These user bases do have a lower average engagement due to the abundance of casual and hypercasual games, where users often don't stick with the game for long periods of time. However, this is compensated with the large size of the user bases. On the monetization side there is the drawback of being required to implement the native payment systems for In App Purchases (IAP), however no forced ads and the revenue split of 12/88 for the Play Store is very appealing. It is not surprising to see that these platforms are already in the workflow. Overall, the mobile platforms are a relatively low effort way to make games accessible to a large user base.

2.5.2 Console platforms

Consoles start out as very attractive options, metrics 1 through 6 all have a very good match with the ideal metrics in appendix B. Consoles are a good fit on demographics with relatively big user bases. The revenue split is also standard. The development costs however far outweigh these benefits. Requesting the devkit from the console developers and rewriting the full game in an engine that compiles to that console costs a lot of development time. Add to that the fact that we have no experience developing for these systems within Wanted 5 Games, which will inevitably cause systemic issues that experienced developers will know to avoid. Implementing the platform's wallet/payment system is another deterrent when considering console platforms. Overall, the console platforms have a lot to offer: big audiences with good retention for the Strategy genre. The development costs are just too high for a small studio that is not familiar with console development.

2.5.3 Web platforms

The web platforms are very easy for us to take on, but offer little value in terms of acquiring dedicated players. A mismatch in age, Low engagement/retention, and the unpopularity of the genre predicts a mismatch in the platform's users and our target demographic. This is contrasted by how easily Wanted 5 Games could release Stratego on these platforms. A web build of the latest version of the game is already being made in the workflow, which means that all that is needed for these web ports is to implement the platform's SDK. Implementing SDKs is not a difficult task, as I have heard in my interview with developers here. I believe it is still worth it to publish to web platforms as they are a good way of generating traffic and filling the lower ranks, the high chance of features and promotion supports this. Overall, there are definitely options with a much better demographic fit that I would pick over web platforms.

2.5.4 PC platforms

Lastly, the PC platforms. These seem to be very solid choices in the analysis. Large user bases with users that have good retention and generally like turn based strategy games like Stratego. These platforms have a good global reach and are well known in the regions where Stratego is popular. However, the PC platforms lose out to the web platforms slightly in metrics 9, 10, 11 and 12. The results for metric 10 (Distribution channels) are the standard for non-web based gaming, so this is not a big issue. Development costs don't have to be a problem either if the web-to-PC port method is chosen carefully.

2.5.5 Optimal platform for Stratego Online

Out of the available platforms, the PC options appear to have the best match with the ideal metrics in appendix B. Out of the PC platforms, Steam is the best fit due to a better match in demographics. Of course, implementing the payment system will take more time and the revenue split is worse than on

Epic games, but this is a good trade-off for reaching the better fitting user base.
Based on these conclusions, this thesis will move forward with developing a PC port for the Steam platform.

3 The required components when porting a game to Steam

Unfortunately, the current web/mobile builds of Stratego cannot be uploaded to steam for all those millions of users to download. Each game distribution platform has different requirements for releasing games. Games need to be uploaded in a certain format, have a clean presentable store page, and implement platform features like friend lists (these features are optional).

These rules are in place as a form of quality control on steam, so it is important to be aware of them to launch new titles smoothly.
This is why all the technical requirements and procedures need to plot out in a clear and chronological way. This way, the port workflow can be set up so it includes all the steps necessary to launch the game efficiently to steam.

The requirements will be plotted out in two sections, the first section looking into the requirements for the game and its files. The second section will gather information about the additional requirement for releasing a game on Steam.

3.1 Game requirements

Let's start with the requirements for the game files. Luckily, there are very little rules about how games are packaged when uploading it to steam.

Valve's Steamworks page of Stratego Online provides us with checklists that states how the build of the game should be set up. (*Steamworks*, 2024)
These checklists are available in Appendix D.
In concrete terms, the game should:

- ☐ Be able to run as a standalone app.
- ☐ (optional) have the steam API integrated to use platform features.

What this means is that the Stratego Online Cocos project will have to be exported into the ".exe" format for windows, and the ".app" format for mac devices.(and .sh for Linux OSes)
As discussed before, right now only web and mobile builds are made. Further explanation of creating the standalone application can be found in chapter 4, after finishing up the overview of requirements. (*Steamworks Documentation*, 2024)

For an exhaustive description of all the limitations and requirements, the Steamworks documentation should be consulted.

3.2 Pre-release requirements

Looking at the second figure in Appendix D. the "Store Presence" requirements are laid out.

Besides requirements for the build of the game, two more component need to be set up before releasing the game for all to enjoy:

- ☐ A Steamworks account.
- ☐ The store page.

3.2.1 Steamworks Account

Steamworks is the developer portal for devs who release games on Steam, or as they introduce themselves: “Steamworks is a set of tools and services that help game developers and publishers build their games and get the most out of game distribution through Steam.” (*Steamworks*, 2024)

This is where developers can manage different builds of their game, make packages to sell in the store, and set up and manage the store page itself. However, these features are only available when a Steamworks account is set up.

1. On partner.steamgames.com you can sign in.
2. You'll need to fill out some paperwork in order to sign up for Steamworks. This requires your bank account and IBAN number available, as well as your tax information.
3. Along with the sign-up is also a product submission fee of $100.00 per game you want to distribute on Steam.
4. You will need to go through an approval process, which will take about a week.

This account will also be used to upload the builds to the steam servers, or give other accounts the right to do so. (*Steamworks Documentation*, 2024)

For a full description of this process, the Steamworks documentation should be consulted.

3.2.2 Store page

Alongside a PC build and a Steamworks account, the biggest task is to set up the store page. Not only are nice images and promotional material advised to attract the most players to the game, Steam actually requires specific images and information to approve the page.
Figure 2 in Appendix D. shows the checklist of elements that should be thought of. This figure shows that a lot of the store page requirements are designer/artist tasks. Creating banners, writing descriptions and setting descriptive tags are all tasks that can be done in parallel by anyone with the necessary skills.

The images like: screenshots, banners and icons have very specific resolution requirements. Full details about requirements and restrictions on these art assets are available here in the steam documentation. (*Steamworks Documentation*, 2024)

On a final note, the store page needs to be approved and publicly accessible 2 weeks before release of the game on Steam. (*Steamworks Documentation*, 2024)

4 The options to port a Cocos Creator project to Steam

In the last chapter about releasing a game to Steam, we came across the technical requirement “The game should be able to run as a standalone app”. Unfortunately, this is a little easier said than done. Different devices have very different file formats in their executables. The JavaScript in Stratego Online’s Cocos project needs to be converted to be able to execute on PC platforms (as a standalone application).

Luckily, this is a very common problem. Such a common problem that many game engines have porting features built into the software. Unity even exports to PlayStation, Xbox and Nintendo (only for partnered developers). (Unity Technologies, 2024)

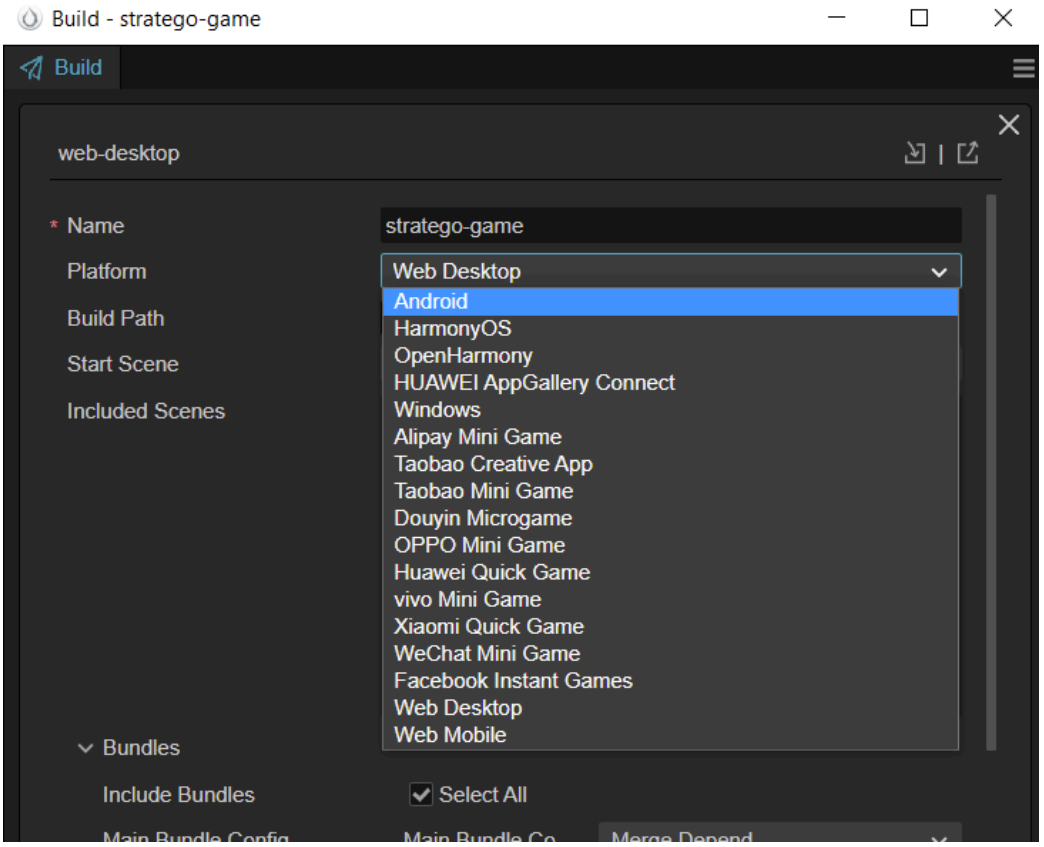
This offers game developers a very appealing workflow for a multi-platform game. Having all the builds for your desired platforms stem from the same project with the same codebase.

This chapter will research the best method of porting in four sections. First, two sections explaining the most common approaches. Then, an overview of the options. And lastly, a section where practical experience with both methods is described, and a final choice is made on the best method for the current project.

4.1 Options offered by Cocos

4.1.1 Native windows export

So let's take a look at the build options for the Cocos creator engine, used to make Stratego Online. To see what options are available for porting.



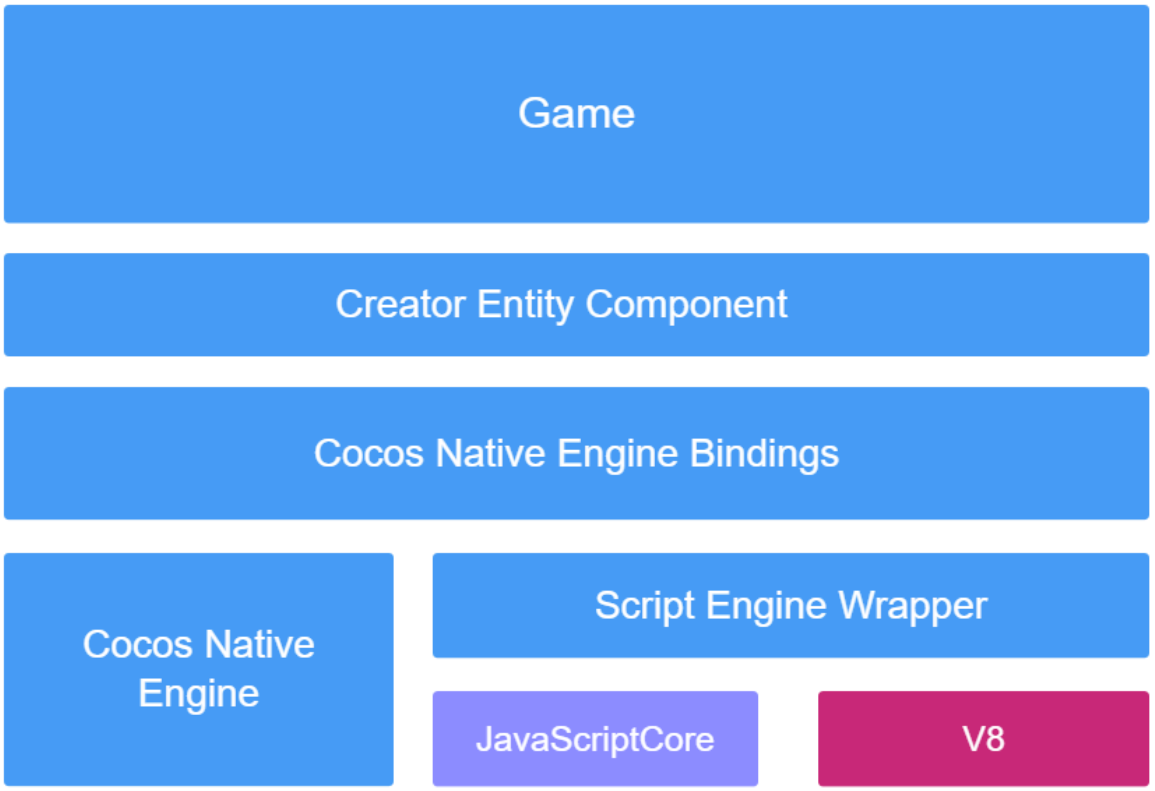
There is a big list of platforms that are supported by the Cocos builder. (*Cocos Creator 3.8 Manual - General Build Options*, 2024) Not surprisingly a lot of these options are for Chinese platforms since Cocos is a Chinese game engine. Included in this list we can see: web, PC and mobile. Just the options we were looking for. When we read the documentation for the Windows build option, we see there is almost no extra work that goes into building to native windows. (*Cocos Creator 3.8 Manual - General Native Build Options*, 2024)

It promises to produce an .exe as long as we have a copy of Visual Studio 2019/2022 with the "Desktop development with C++" and "Game development with C++" installed. It should also be noted that there is a macOS option available when using a mac device. (*Cocos Creator 3.8 Manual - Setting up Native Development Environment*, 2024)

How does it work?

The way these native builds work is with something called JSB, or JavaScript binding.
JSB is used to bridge the gap between JavaScript code (used for game logic) and native code (C++ or other languages used for performance-critical tasks). This allows us to write the game logic in

JavaScript while using the performance benefits of native code for key operations like rendering and physics. (*Cocos Creator 3.8 Manual - Tutorial: JSB 2.0*, n.d.)



(*Cocos Creator 3.8 Manual - Tutorial: JSB 2.0*, n.d.)

In this diagram, we can see this happening visually.
We start on top where the game logic runs in JavaScript, this is the game as we make it in the Cocos Creator editor. All the scenes and scripts run as they usually would, using the components like sprites and hitboxes.
However, when these components need to do something performance heavy, like rendering or hit detection we would much rather do this in the faster native environment (C++ in case of Cocos' windows export option). (*Cocos Creator 3.8 Manual - Tutorial: JSB 2.0*, n.d.)

That's where the JSB layer comes in.
JavaScript binding creates a bridge between the JavaScript methods and data types, and the C++ methods and data types that Cocos has written in the Native Engine.
This way there is a JavaScript sprite, that we used to represent a Stratego piece for example, and a native sprite implementation that has the same functionality, but written in more efficient C++.
The JSB bridge then ensures that the C++ methods execute whenever the corresponding JavaScript methods are called. (*Cocos Creator 3.8 Manual - Tutorial: JSB 2.0*, n.d.)

4.1.1.1 Advantages

The clear advantage of building native apps with JSB is that you can use the same code base for web and native platform builds. Cocos Creator developers have done the heavy lifting for us by creating native engines for PC and mobile, this makes porting relatively easy.

This native functionality also brings better performance by offering use of the GPU for things like rendering, and better use of the CPU by running C++ over JavaScript. (*Cocos Creator 3.8 Manual - Tutorial: JSB 2.0*, n.d.)

4.1.1.2 Drawbacks

At the time of writing, some Cocos components are not supported by the native Windows engine. The “label shadow” and “label outline” do not have a C++ counterpart and do not work as they would on web. We use these components often in Stratego Online at this time. This brings the risk of certain functionality not working on Windows, which would require finding workarounds. (Kingurus, 2023)

Custom components will also need to be manually added to the native engine (along with setting up the JSB) to enjoy the performance boost of the native platform.

Native counterparts of components might cause platform specific bugs. This has been the case already with the mobile builds of Stratego Online: IOS devices experienced crashes due to audio issues, and the native implementation of Spine (the animation software used) would sometimes not display the animations correctly. These issues would need a platform specific fix.

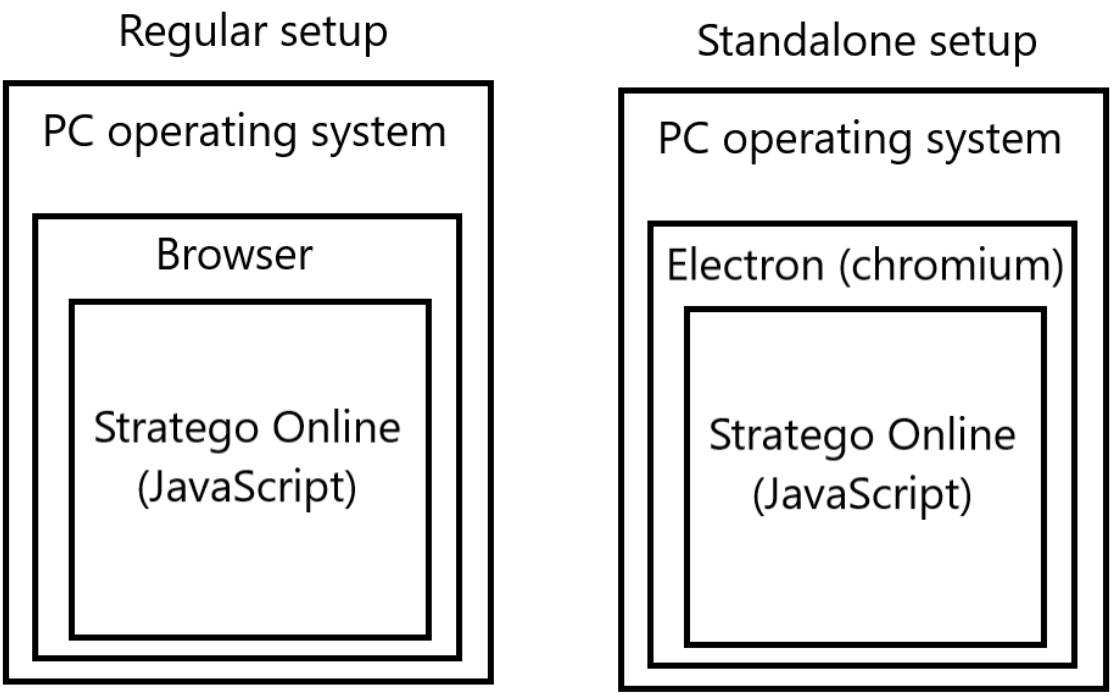
4.2 Alternatives

4.2.1 Electron

Another popular way for web apps to get ported to PC platforms is by wrapping the web build into a lightweight browser that runs as the standalone application. A popular framework within this workflow is Electron. They describe themselves as follows: “Electron is a framework for building desktop applications using JavaScript, HTML, and CSS. By embedding Chromium and Node.js into its binary, Electron allows you to maintain one JavaScript codebase and create cross-platform apps that work on Windows, macOS, and Linux — no native development experience required.” (*Introduction | Electron*, 2023)

Sounds fairly ideal, having our existing codebase in typescript running in a standalone stripped down browser that’s disguised as an executable.

Like they explained in the quote, Electron uses the Chromium browser as its rendering engine. Chromium is an open-source browser that is used by Google Chrome, this provides Electron apps with a powerful web platform.
Here is a diagram showing the JavaScript game logic running like usual, even though to the user it feels as if they are using a native application. (*Introduction | Electron*, 2023)



Electron apps have two main processes: the main process and renderer processes. The main process manages the lifecycle of the application and creates web pages (render processes) to display UI. Each render process runs in its own environment, providing isolation between different parts of the application. (*Glossary | Electron*, 2023)

Building upon this, Electron provides communication between the main process and render processes. This allows for sending messages, sharing data, and triggering actions between different parts of your application. The main process has access to native features that allow you to access operating system functionality such as creating windows, handling system events, accessing files, and interacting with hardware. (*Glossary | Electron*, 2023)

There are a number of other frameworks that use a similar approach, like NW.js and AppJS. However, Electron is the most widely used out of similar options, and the difference between these options does not appear to be very significant.

4.2.1.1 Advantages

- The code base is the same and executes in exactly the same way as the web version. This means very low chance of platform specific bugs.
- The porting process is also very easy, as the web build can be used directly for the whole game.
- A lot of plugins and tools exist for Electron, offering solutions to common problems and integration to the main processing layer of the app.

4.2.1.2 Drawbacks

- The code still runs in JavaScript, and not in a faster native language like C++. (Pereira et al., 2021)

- Reduced security compared to native applications due to the game still technically running in a browser, and access to the console and other browser information can never be fully blocked from the users.
- Due to the inclusion of Chromium and Node.js, Electron apps will experience worse: Resource management, performance, and file size when compared to native apps.

4.3 Overview of options

There are two popular options for porting HTML5 games to a standalone application:

- Building to a native app that utilized JSB binding. Cocos offers this option out of the box.
- Packaging the web build into a chromium based application using a framework like Electron.

Both options of course offer pros and cons, but the conclusion from the research overall is: the more native the build gets the more work is required, and the more advantage you receive from native functionality.

The extreme case here is completely rebuilding the game in an engine using a native language (or language that compiles efficiently into native code). The Unreal or Unity engines come to mind for this option. However, I decided to leave this option out of the comparison since rebuilding the game in a new engine is not realistic for the development team. A second codebase in another engine is too much manual work for Wanted5Games to take on. This effort would not be worth the benefits of having a PC port with better performance than the two options selected above.

4.4 Practical look into port options

Before deciding on the most effective option for making the PC build, hands-on experience with both implementations should be gathered. A benchmark test can be set up to get more practical information about the advantages and disadvantages.

This test will look like this:

1. Define minimal functional requirements.
2. Export the game as a native build until requirements are reached.
3. Export the game as an Electron project until requirements are reached.
4. Compare results.
5. Decide on best option.

4.4.1 Minimal Functional requirements

For this test, the ports should not be fully worked out and polished. The requirements should only ensure that the game and all its components are functional. This is reflected in the following requirements.

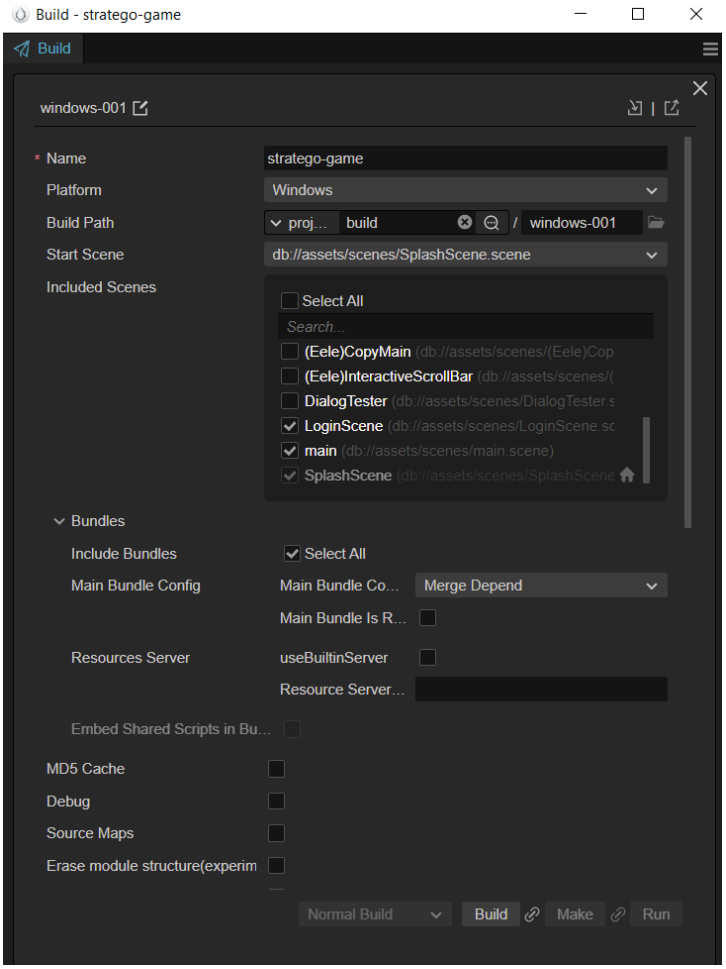
- ☐ Game starts through .exe file.
- ☐ All cocos components used are functional.
- ☐ There is a stable connection with the backend.
- ☐ The gameplay is functional.

A build that fulfills these points can be uploaded to steam and should be able to play a full match against a client connecting from the web build.
These requirements were set up together with Robin, the CTO.

4.4.2 Native Cocos export

4.4.2.1 Making the build

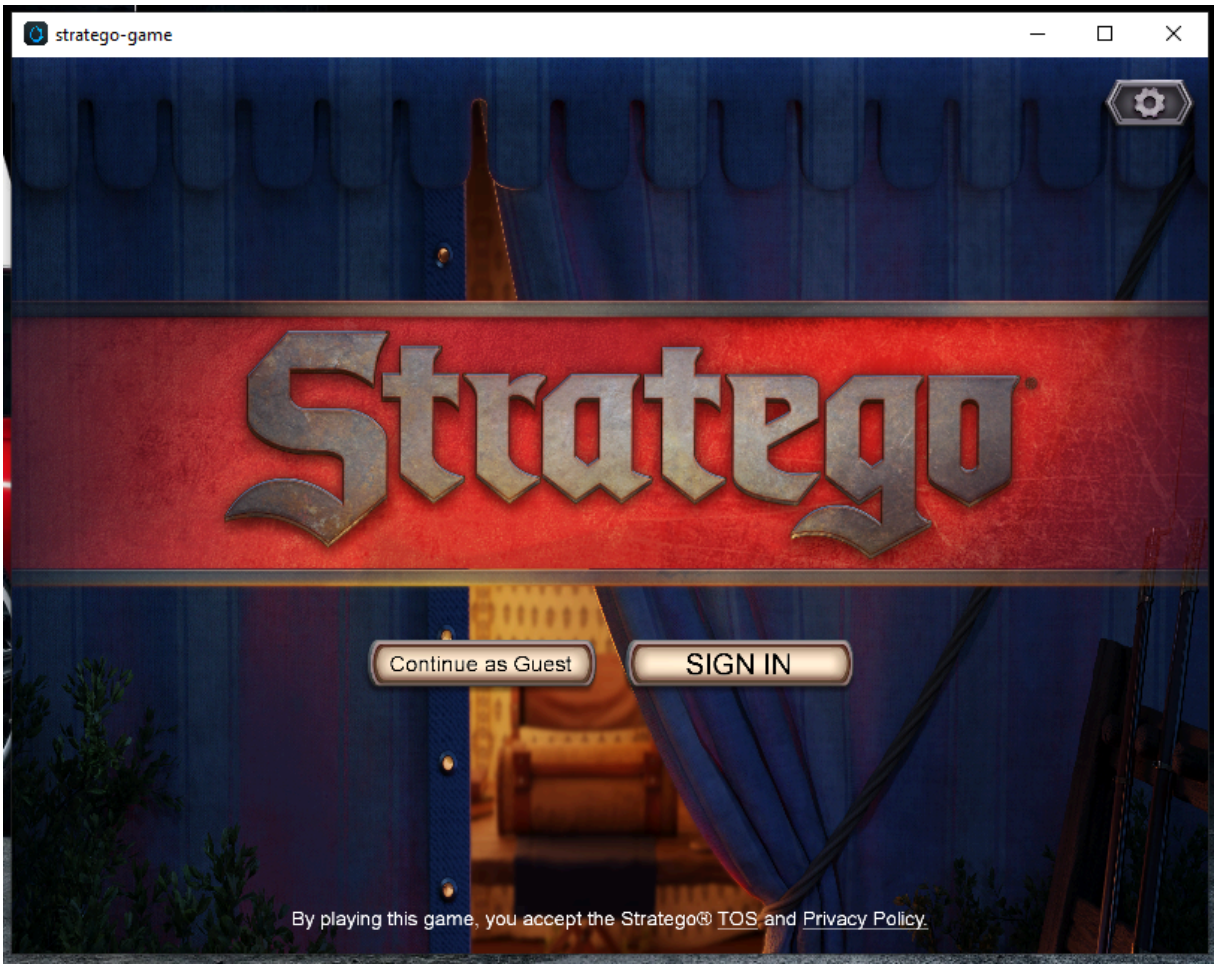
Making the native windows export is very similar to making the standard web build for Stratego.



Simply select Windows as the platform, and then set up the rest of the options as you would do with the web build.

After “Building” and “Making” an .EXE file of the game is produced (*Cocos Creator 3.8 Manual - General Build Options*, n.d.)

Running this build shows the following app:



4.4.2.2 Checking for requirements

Straight away the requirements do not appear to be fulfilled correctly. There are 3 issues that appear in the login screen.

Issue 1: fonts not showing at all

Issue 2: text shadow and outline is not supported yet on native windows

Issue 3: websocket connection failed when logging in.

Further research of these issues is described below in an attempt to find easy fixes to get the build working.

1. Font issue

User theamirbeg (2023) made a post on the Cocos forums in March 2023 about the same issue that is occurring here. The fonts work in the editor and in the web build, but not in the native build. Luckily a

Cosos developer linked to another post where a solution for this problem was posted.

The name of the font is not the same as the name of the .TTF file. This will cause issues in the windows build. After changing the name of the font to the name of the file, the font worked in the native build.

2. Text shadow and outline issues

User rui-gameking (2023) made a post on the Cocos forums in September 2023 about the same issue occurring here. The text outline (*Cocos Creator 2.4 Manual - LabelOutline Reference*, n.d.) and shadow components (*Cocos Creator 2.4 Manual - LabelShadow Reference*, n.d.) did not appear to be working in the native windows build.

A cocos staff member replied that these components were not supported on the windows platform yet. This means an alternative component needs to found and used when the game is being built for windows.

3. Websocket Error

The final issue that appeared from the login screen was that logging in kept failing. The output logs of the application showed the ConnectionManager class printing these messages: “ConnectionManager:createAndConnectSocket - Error”

This error is produced by the networking client, Nakama, that Stratego Online uses to connect the frontend to the backend. Without the websocket running, the frontend cannot send or receive any data from the backend. (Nakama, n.d.)

It would be a problem if this Nakama client could not be used with the native build, since implementing alternatives just for the PC build would bring a lot of work and risk of platform specific bugs. That is why further research was conducted into fixing this issue in an easier way.

After the logs were inspected more thoroughly, an error with the “CA SSL certificate” was found . The certificate was missing while creating the socket. (Nakama, n.d.)

The reason that this is not an issue in the web build is that the browser usually handles this certificate, but the native app of course does not run in a browser. The next step for was to find out how to create a CA certificate to use for the native build, hopefully connect the websocket. (Georgiev et al., 2012)

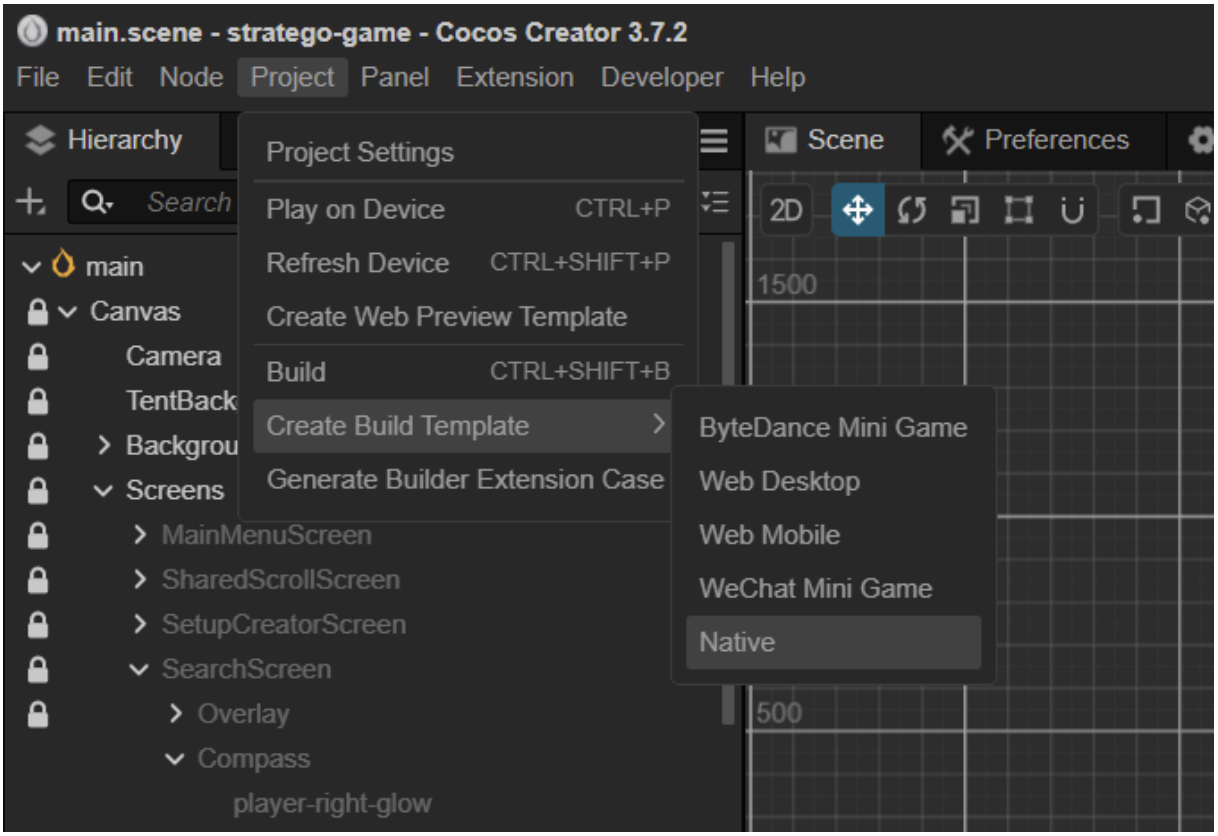
To get a new certificate, there is the correct way and the way that is just for testing:

1. Buy a new CA signed certificate from a publicly trusted CA, this guarantees it works and is relatively cheap over several years. (Crane, 2021)
2. Make a self-signed certificate. This will work as a certificate until a real trust check is done. This certificate is not anchored to a trusted CA and so will not be sufficient in cases that need real security (which we want). A self-signed certificate will be tried first to see if the windows port works. (Crane, 2021)

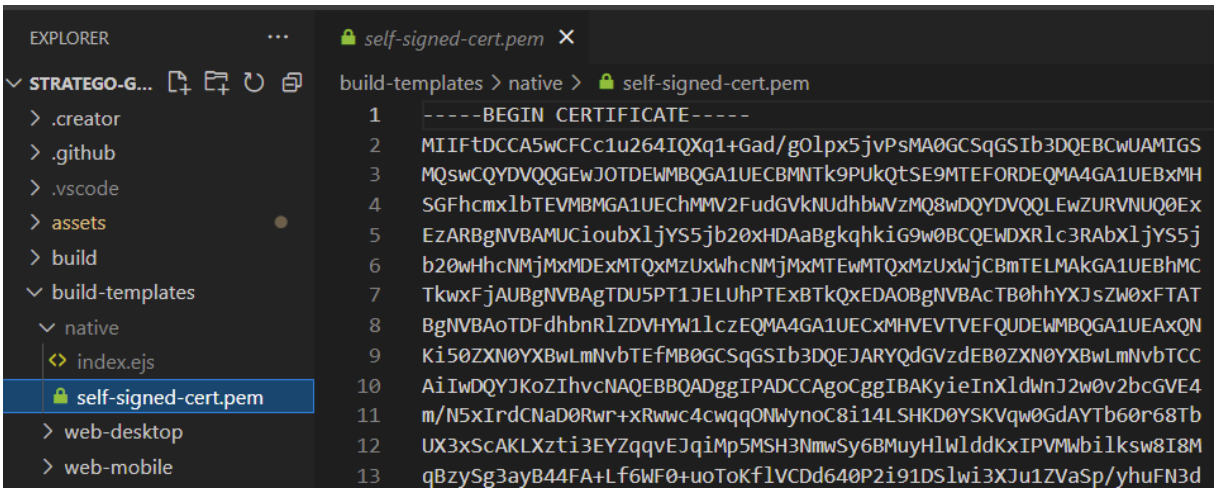
For debugging purposes, a self-signed certificate was made and implemented. An article by Dissanaike (2023) was followed to make this test certificate.

Firstly, the certificate needs to be added to the files of the build. Files can be added using build templates like this:

- Create a new native build template in cocos creator here:



- Then add the files you want to add to the build in this project folder. Here the self-signed-cert.pem file will be added to the root folder of the build:



- Then the certificate can be used by the Windows app scripts to create the websocket. See this example in the WebSocket-libwebsockets.cpp class :

```
struct lws_vhost *WebSocketImpl::createVhost(struct lws_protocols *protocols, int *sslConnectionOut) {
    #if(CC_PLATFORM == CC_PLATFORM_ANDROID)
        LOGD("override cafile path");
        _caFilePath = "cacert.pem";
    #endif
    auto *fileUtils     = cc::FileUtils::getInstance();
```

The method described above was implemented, but issues still arose when creating the websocket, which are likely not resolved due to the certificate to be self-signed.

After consulting the CTO about this issue, he provided a CA signed certificate that is being used by the backend of the game. This solved the error message of a missing CA file, but the websocket could still not be created.

For a final analysis of the cause of the problem, a test was performed in an existing project where a websocket would be initialized for a Windows build. This project is the cocos-test-projects (Cocos, 2023), a collection of example projects that show implementations of key features in the Cocos engine.

One of these implementations is networking. In one example, a websocket connection is set up and tested for native applications. It is specified that SSL certification is necessary when connecting to websockets from native applications. So in the code, the CA certificate is passed to the WebSocket C++ implementation.

The attempt to replicate this in a new project failed.

The game itself has been tested after bypassing the login step, and all the other gameplay features and animations appeared to work as intended.

4.4.2.3 overview

The standard windows export has some clear issues out of the box.

- [x] ~~Game starts through .exe file.~~
- [ ] All cocos components used are functional.
- [ ] There is a stable connection with the backend.
- [x] ~~The gameplay is functional.~~

There are problems with cocos components and custom networking libraries that have been added to the project. After researching these problems, it did not appear that there were easy solutions for these problems. If this port option were to be used, extra development time will have to be put into these issues before a satisfactory build can be made.

The native export option in Cocos Creator does not pass the benchmark test. Two of the four requirements were not fulfilled.

4.4.3 Electron framework

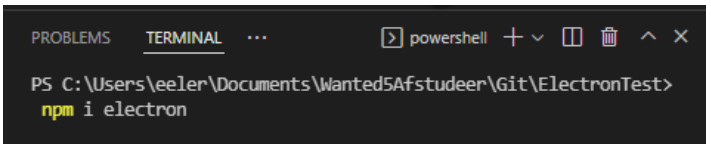
As discussed before, Electron is a framework to run HTML 5 projects as standalone applications. A community post on the Cocos forums by user debugChicken (2021) was followed to set up the first version of the Electron port.

This post walks through the steps of preparing a cocos project for steam using Electron, which is exactly what is attempted here.

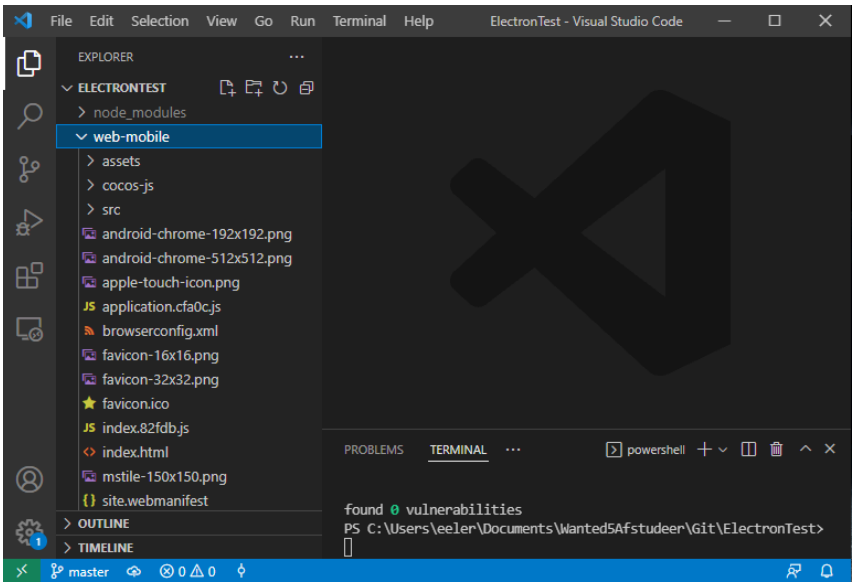
4.4.3.1 Making the build

An Electron build can be created by following the easy steps below.

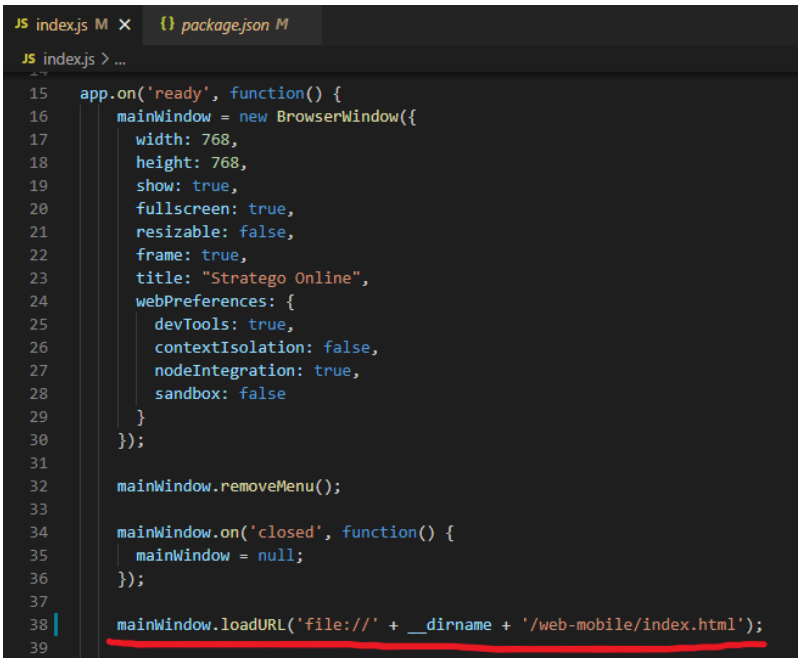
1. Start by creating a new project in Visual Studio Code and initialize it as a git project for the sake of version control.
2. Then set up the Electron framework as by installing the npm package.



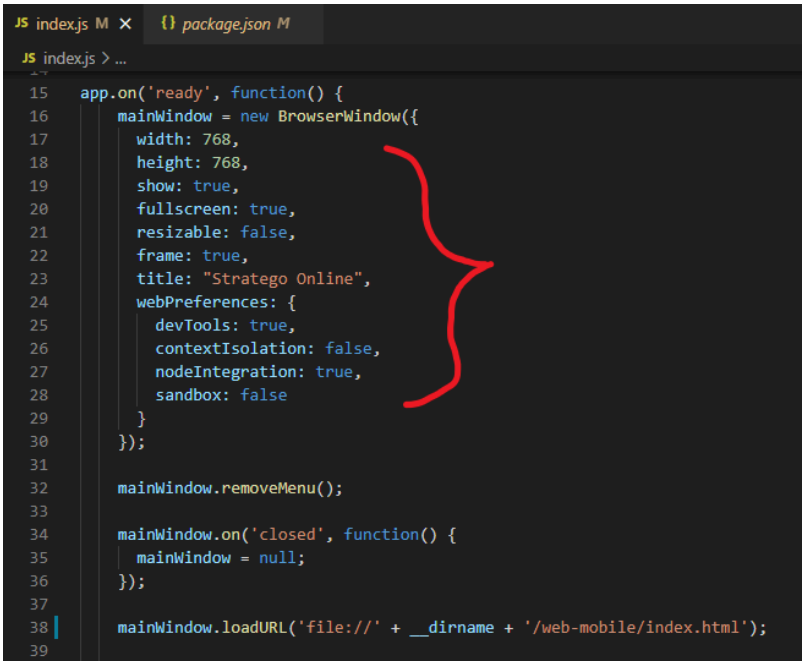
   The last step of setting up is creating an index.js file that will create the web page that will run inside the application. This index file is where the web build of Stratego will be loaded in, in a later step. An example of a standard index file can be found in Appendix E.
   At this point, the empty project should be able to launch by running “electron .” in the terminal.
3. Now make a web build in Cocos and paste it in the root of the Electron project like in the figure below.



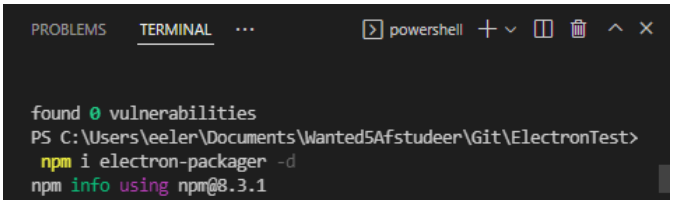
4. By loading the index.html into the “mainWindow” of the index.js file the game will run as soon as the application is started.



5. Preferences for the window can be changes while instantiating the browser window. This offers the option to toggle features like: full-screen, context isolation, and resolution.



6. When everything is working as expected, the application can be package into an executable. The chosen method to do this is with another npm package, “electron-packager” (Electron, n.d.). Like the name suggests, this package will wrap the project into a nice executable for distribution.



7. It can make builds for windows, macOS, and Linux. The figure below shows the command for creating a Windows build titled “stratego-game”.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\eeler\Documents\Wanted5Afstudeer\Git\ElectronTest> electron-packager . stratego-game --platform=win32
Packaging app for platform win32 x64 using electron v27.3.2
WARNING: Found 'electron' but not as a devDependency, pruning anyway
Wrote new app to: C:\Users\eeler\Documents\Wanted5Afstudeer\Git\ElectronTest\stratego-game-win32-x64
PS C:\Users\eeler\Documents\Wanted5Afstudeer\Git\ElectronTest>
```

   The build will be exported to the root of the project, unless a target folder is specified.
8. finally, the exported .exe file was run to test the requirements.

4.4.3.2 Checking for requirement

Everything seems to be working as it should inside the Electron application.
Users can create an account, log in, start looking for a game, and play a full round of ranked.

All requirements seem to be fulfilled.

4.4.3.3 Overview

The Electron export works very well out of the box.

- ☑ ~~Game starts through .exe file.~~
- ☑ ~~All cocos components used are functional.~~
- ☑ ~~There is a stable connection with the backend.~~
- ☑ ~~The gameplay is functional.~~

The project is easy to set up and only needs to be instantiated once. If a new version needs to be built, only 1 folder needs to be swapped out in step 3.
There were also no bugs or issues that the web build does not have because the exact same code is running.
Overall, this is a reliable and easy to use method of producing an executable.

The Electron porting method does pass the benchmark test. All four requirements were fulfilled.

4.4.4 Comparison

Looking at the implementations of both options a can see that the native windows export in Cocos is missing some requirements and did not pass the benchmark test.
Certain features are not functional and while trying to solve these issues it was discovered that it is also harder to debug the native build (rui-gameking, 2023). Code and scene changes could only be tested after building the whole project again. Reading the logs from the code is also harder because they are printed by the C++ console. This means full objects cannot be logged like in in the browser console.

An easy solution could not be found for the text and websocket issues, but platform specific fixes would likely need to be made in order to get these important features working.

The Electron project on the other hand worked exactly like it should, being quick and easy to use and having very little chance of not functioning like the web version.

Due to the result of the benchmark test and insights gained during the practical implementations, this thesis will continue building the steam version using the Electron framework.

5. Adding the remaining platform requirements

The result of chapter 4 is a working standalone PC build of Stratego Online, so now would be a good moment to refresh on the platform requirements of steam that were covered in chapter 3.

The build of the game should:

- ☑ ~~Be able to run as a standalone app.~~
- ☐ (optional) have the steam API integrated to use platform features.

Pre-release requirements:

Before the game can be uploaded to the steam servers and released, 2 things first need to be set up:

- ☐ A Steamworks account.
- ☐ The store page.

The research and practical implementation results can be found below in four sections. Starting with section one which is an overview of Steamworks and how it is set up. Section two builds upon this by implementing Steamworks API features. The third section contains an overview of the tasks for the Steam store page, and finally section four concludes by checking whether all platform requirements have been met by the porting system.

5.1 Steamworks

This section will discuss the tools from Steamworks that are needed to upload, manage and release Stratego Online on steam.

In order to implement the Steamworks API features, the Steamworks account needs to be set up first.

Valve describes Steamworks as follows: “Steamworks is the set of tools and services that help game developers and publishers get the most out of distributing games on Steam”. (*Steamworks*, 2024)

This section will discuss how to:

- Set up a Steamworks account.
- Upload builds to the steam servers.

- Manage the project in the portal
- Implement Steamworks API features.

5.1.1 Setting up the Steamworks account

Luckily, Valve has an official “Getting Started” documentation on the Steamworks site. (*Steamworks Documentation*, 2024)

This page acts as a great base for anyone looking to become a developer on steam. Right now the interesting topics can be found in the following groups of documents: “Onboarding”, “Steam Direct Fee”, “Managing Your Steamworks Account”. “Onboarding” describes setting up a partnered Steamworks account. “Steam Direct Fee” walks through purchasing a unique appID for a new game that is not released. And the final documents are an overview of how to add accounts to Steamworks for developers to manage the game. (*Steamworks Documentation*, 2024)

5.1.2 Upload builds to the steam servers

When the account information is all approved, and the Steamworks page for the game is set up, the Electron build from chapter 4 can be uploaded and tested.

This subsection will look into the Steamworks SDK, the tool needed to use to package and upload game files to the steam servers.

The Steamworks documentation was used as reference for the subsections on uploading to steam. (*Uploading to Steam (Steamworks Documentation)*, n.d.)

5.1.2.1 What is the Steamworks SDK?

Valve describes the SDK like this: “The Steamworks SDK provides a range of features which are designed to help ship your application or game on Steam in an efficient manner.
The Steamworks SDK is only required to upload your content to Steam, everything else provided through the SDK is optional.” (*SteamWorks SDK (Steamworks Documentation)*, n.d.)

There is extensive documentation which contains all the features on the Steamworks site. (*SteamWorks SDK (Steamworks Documentation)*, n.d.)

But the features needed currently are:

- ContentBuilder - For manually building and uploading to Steam
- SteamPipeGUI - A SteamPipe GUI Tool for Windows to make uploading simple products even easier.

The GUI tool is unfortunately only available for Windows devices, which is why the content builder needs to be used manually on macOS devices. (*SteamWorks SDK (Steamworks Documentation)*, n.d.)

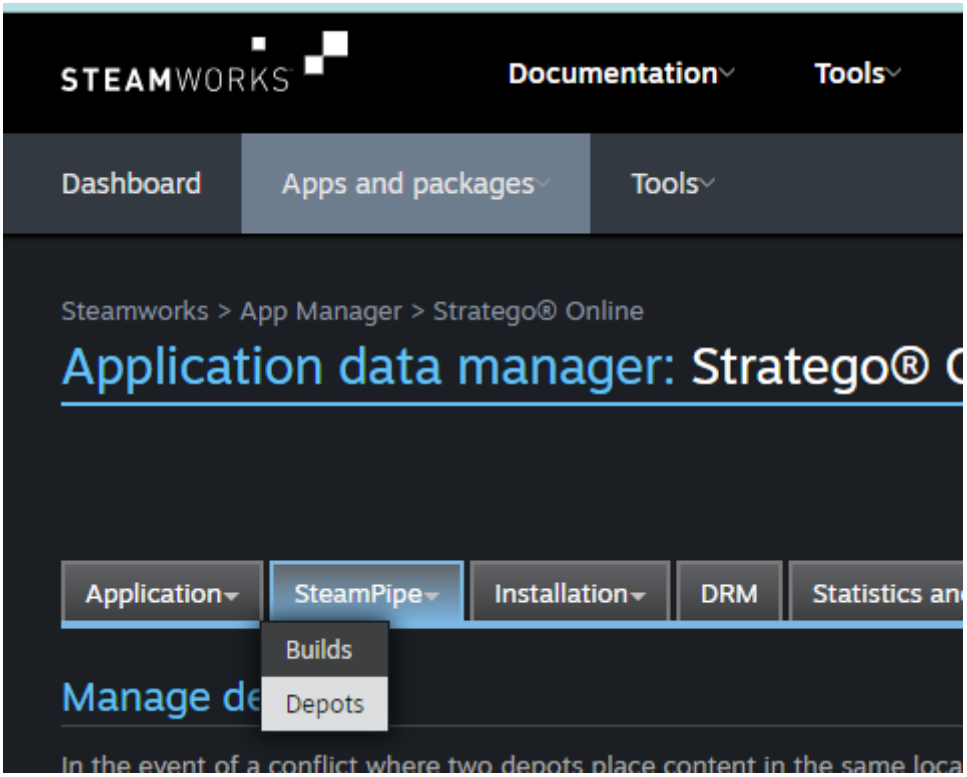
5.1.2.2 Uploading Windows/Linux builds

When uploading to Steam, we have to be aware of the package structure of Steamworks. There are 3 layers in the package that can be created:

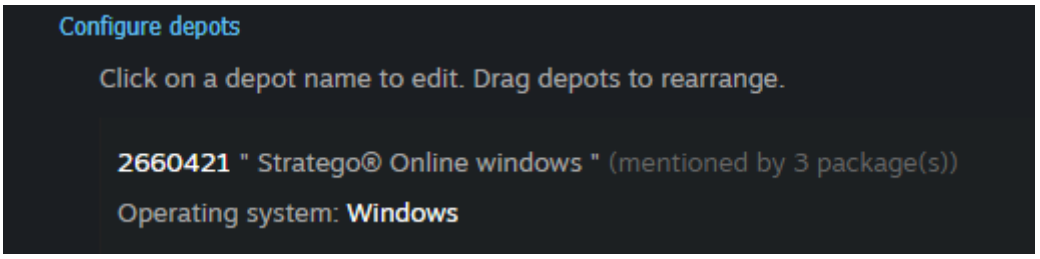
1. Depots - Steam allows you to make depots that will be installed when a user downloads your game, one depot might contain the windows build, and another the mac build. You can decide which users get what depots on download.

2. Builds - You can then make builds which are collections of depots, when you have a new windows depot but not a new mac depot, you can make a build where only the windows depot is different.
3. Packages - Lastly, packages need to be made for users to actually download the game from the store. A standard package would just contain the latest version of the game, but packages can also include other games you made on steam, or DLCs for your game.

To upload the build, a depot needs top be set up. Managing depots is done on the Steamworks web portal. (*SteamWorks SDK (Steamworks Documentation)*, n.d.)



Here a depot can be set up for the base game, and tagged the operating system as Windows.



The depot number (2660421) will be important later on when uploading.

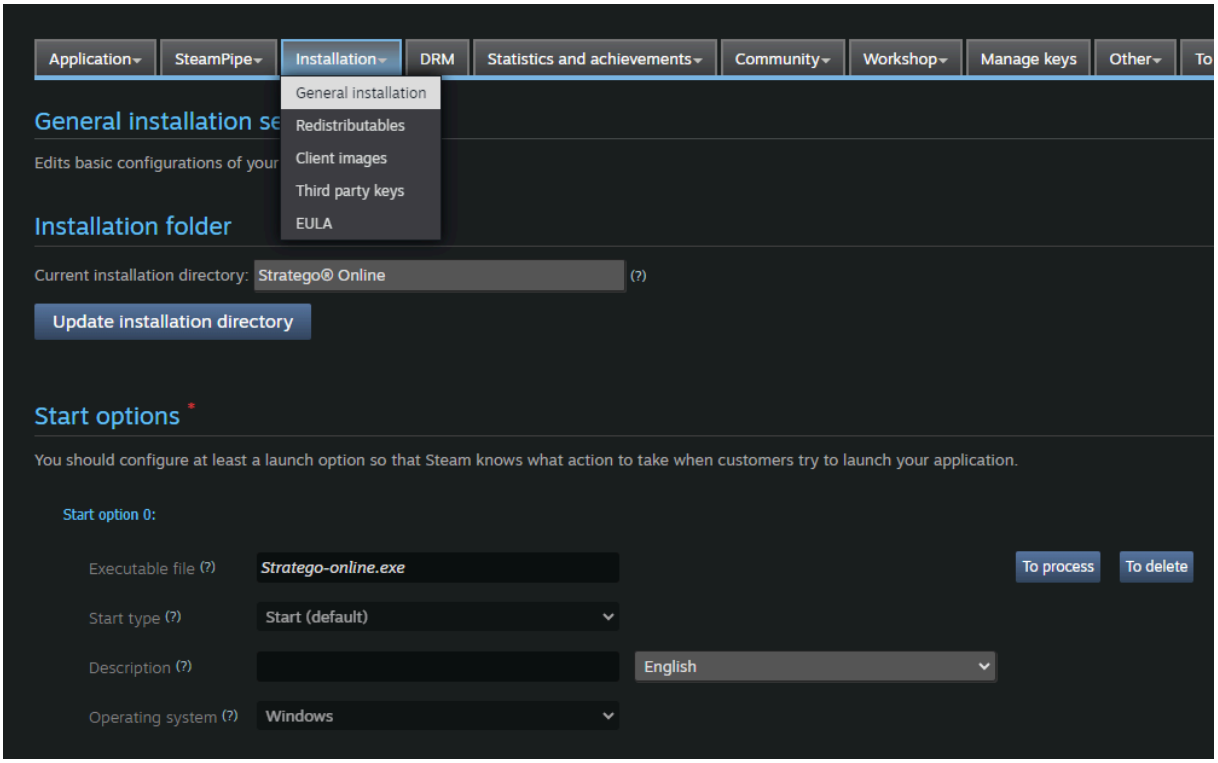
The last thing that is needed are the tools to upload, the Steamworks SDK. The latest version of the SDK is available at this URL: https://partner.steamgames.com/downloads/list

With all the prerequisites finished, the steps of the upload workflow can be executed. This is the Windows workflow, the macOS workflow will be discussed in a later section.

1. Open the file explorer and make a new folder in the sdk/tools/ContentBuilder/content folder of the SDK, folders made here will become the contents of our depots. You only need to do this once for every new depot.
2. Put the working desktop port into the new folder.
3. Unzip the sdk/tools/SteamPipeGUI folder if it is not already, and run SteamPipeGUI.exe.
4. Fill in the following:
    - App ID, found on the Steamworks page.
    - Build Description, name of the collection of depots that you are creating.
    - Add the depots that you want in this build to the list.
    - The path to the ContentBuilder folder, this is critical for the GUI to prepare the content for uploading.
    - Click Generate VDFs if they are not generated yet OR if you add new depots. These will build to sdk/tools/ContentBuilder/scripts. VDFs are used to format the data of our files to Valve's Key Value format. (*Uploading to Steam (Steamworks Documentation)*, n.d.)
    - The login and password info to the steam account linked to Steamworks.
5. Now when we press “upload” a console app will now open. Do the following here:
    - Provide the steam guard code for the relevant account, if necessary.
6. It should now upload and give a success message in the “Upload Log Output” window in the GUI.
7. The build will now be uploaded to the steam servers in Steamworks. Open Steamworks.

The next steps are in place to make the build we just uploaded available for (test)users. These steps are all in steam and on the Steamworks portal.

8. Under the installation tab of the Steamworks portal, fill in the path to the game .exe file, using the build as root. See the example below.



9. To publish the installation setup, go to the ‘publish’ tab. Click “Prepare for Publishing” then click “really publish”. This will not release the game to users if it is not already released, it will just save the config settings.

After this, the option of setting up branches is available, a “beta” branch might have an experimental version of the game that you only want some players to be able to test. Developers can decide what branches will receive what builds. The default branch is already initialized and can be used if no other special branches need to be made.

10. Now go to the SteamPipe/Builds tab to select a branch for the new build that was uploaded.
11. Click “preview change”, then “Set Build Live Now”, this will update the build that users on that branch will receive.
12. Restart steam and go to the game page, if everything went as planned, the game should be in your library, and should now be able to launch from steam. If there is an error while launching, check the launch options that you configured earlier under the “Installation” tabs.

5.1.2.3 Uploading macOS builds

Electron is a good choice for making builds for all operating systems. The node module that is used, “electron-packager” builds executable programs for Windows, macOS, and Linux. The only catch is that the Apple package needs to be built on an Apple device.

Luckily, the Electron project is on GitHub, so the project can run on the company Mac device. No code changes have to be made in the electron project to build the app for Mac instead of Windows. The command only needs to be changed to `electron-packager . Stratego-online –platform=darwin` in the terminal and a “.app” file is created that can be run on macOS.

It was mentioned earlier in chapter 5.1.2.2 that the SteamPipe GUI tool in the SDK files is an .exe file and can’t run on Mac devices.

This leaves us with 2 ways of uploading our depots to Steam.

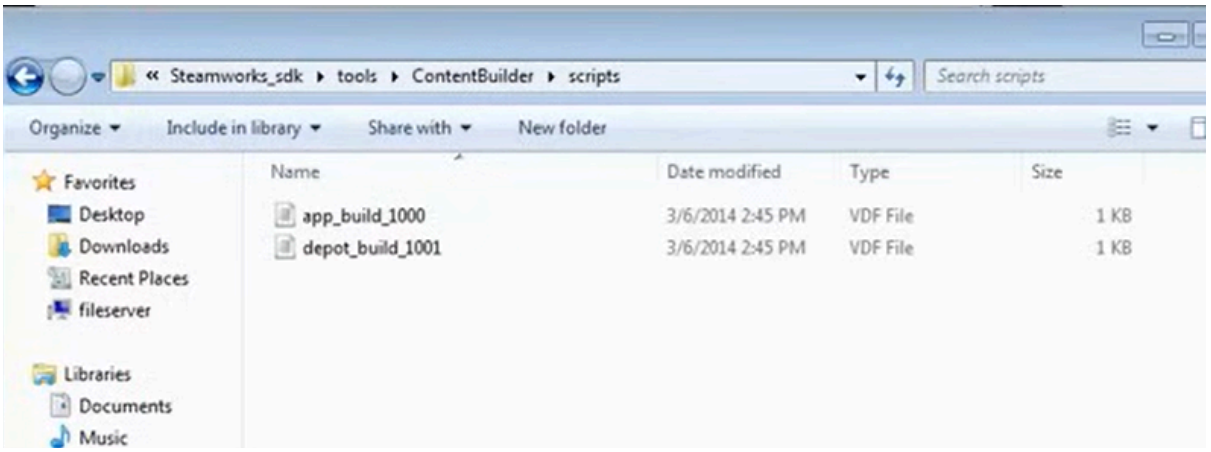
- The first way to upload a depot is by manually using the tools in the SDK. This includes writing VDF scripts and running an upload script in the terminal.
- The second way to upload a depot is by uploading it as a zip file on the Steamworks website. A maximum size of 2 GB is allowed here.

Unfortunately, the second option of using the web uploader does not appear to work for macOS builds. The uploader breaks the symlinks of the .app file, which causes the program to crash when it is started. This leaves developers with manually using the tools in the SDK.

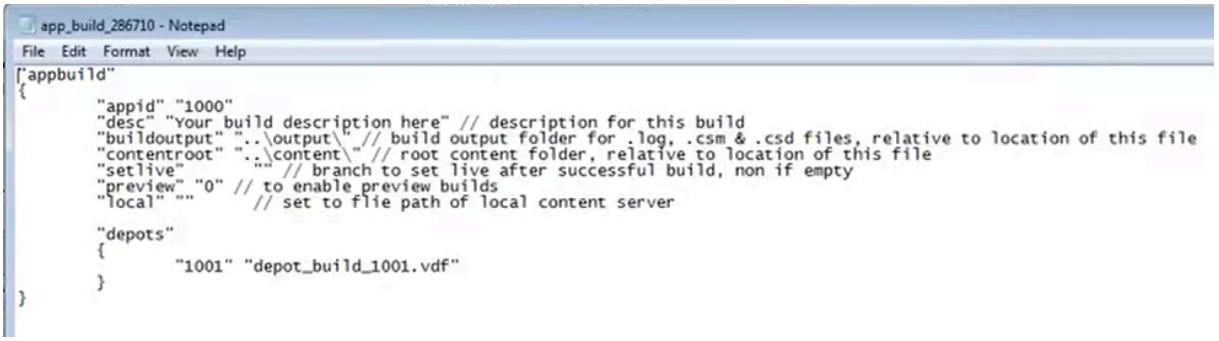
Valve has a YouTube channel called “Steamworks Development” that has a series of tutorials “to help you build your content for Steam via the Steamworks tools.”(Steamworks Development, 2016). The first tutorial shows how to manually set up VDF scripts and running them in the SteamCMD to upload depots.

These steps are described below:

1. App build VDF script.



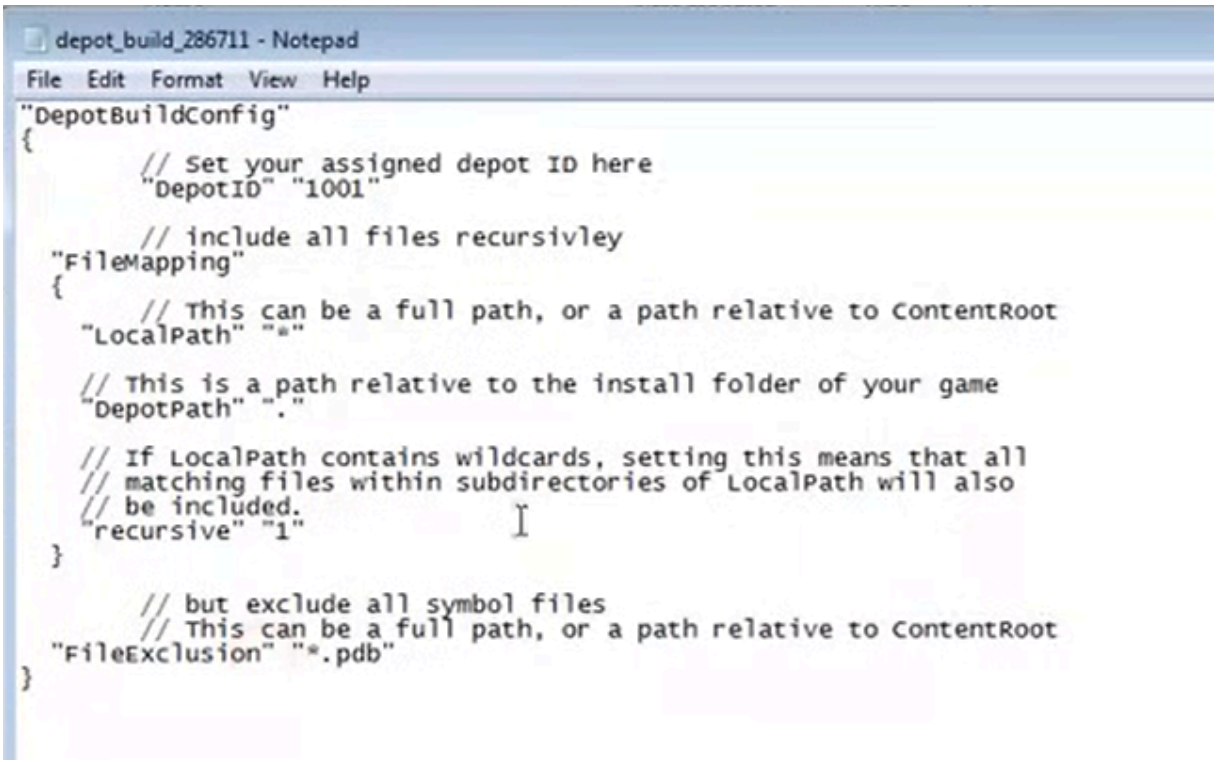
The “app_build_1000” VDF file will be run in the SteamCMD later on. This script calls the Depot building scripts and constructs the full app for uploading. The app build script looks like this:



Here The app id has to be changed to the real app id of the Stratego Online project, 2660420 instead of 1000. The description will be visible in the Steamworks web portal after uploading this build.
For every depot that should be included in this build, the id and depot build script should be listed.

2. depot build VDF script

The depot build scripts describe what content from what folders should be included in each depot. The VDF script for this looks like this:



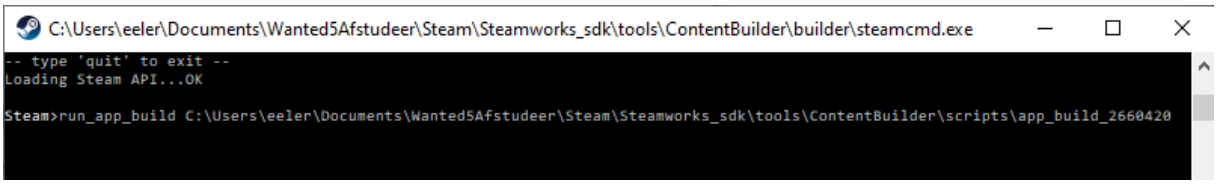
To decide what files will be packaged, developers have to set the “LocalPath” property. Currently, all files in the content folder are included because “LocalPath” is set to “*”. The ContentRoot that is described is the ‘sdk\tools\ContentBuilder\content’ folder.

If, for example, only a version of the game that is in the ‘sdk\tools\ContentBuilder\content\Stratego_Windows’ folder should be included. The “LocalPath” property needs to be set to: “.\Stratego_Windows\”.

3. Building and uploading.

After setting up the app and depot build scripts, they can be run using the SteamCMD. This program is included in the SDK.

After opening the SteamCMD, and logging in, the script can be run with “run_app_build <path to app build script>” like in the example below:



This will wrap the files in the content folder like the depot scripts specify. After building, the depots are uploaded to the Steamworks server. The terminal should print a confirmation message if everything is successful.

5.2 Steamworks API

This thesis spent a lot of time setting up Steamworks and uploading a build, but what was all that work actually for?

Well, now the correct setup is present to utilize the API features that steam offers as a platform. Friends, invites, and achievements are the biggest reasons for implementing the API. These features will add value to the game because they add smooth interactions that players on Steam are used to and like.

This section will look at the API features Wanted 5 Games want to implement, and how to practically set this up in the Electron project.

5.2.1 Implementing JavaScript Steamworks library

Valve states that the Steamworks API is a c++ API in their documentation. (Valve Corporation, n.d.)

Since all the code in the game is currently in JavaScript, we need a way to communicate with the API. Luckily, there are multiple implementations in the form of npm packages. These form a middleman in the communication, transferring JavaScript calls into c++ calls that the API understands.

The source that was used (building Steam games with Electron) suggests that the Greenworks npm package is used. (debugChicken, 2021)

5.2.1.1 Greenworks

To use Greenworks, an instance needs to be built, so it is compatible with our versions of node and Electron. (Greenheartgames, n.d.)

The following steps are required to start implementing the API features. (Greenheartgames, n.d.)

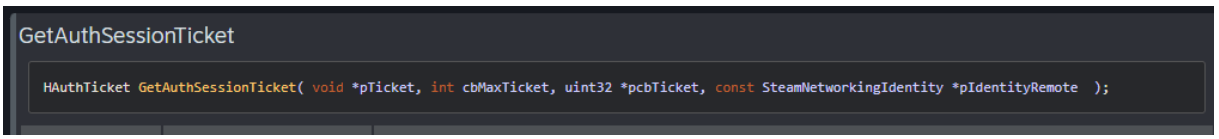
1. Build an instance of Greenworks that uses the installed version of node.js.
2. Copy the Greenworks build to the electron project and load in into the index.js.
3. Set up a class in the cocos project that will invoke Greenworks methods through the app window.
4. Call API functionality through this interface class.

This was implemented in a new Visual Studio Code project. The build of Greenworks was made following the installation guide on GitHub. (Greenheartgames, n.d.)

However, the latest version of Greenworks does not get past step 1 of the forementioned steps.

When the project was built for the first time, the following error appeared `error C2660: 'ISteamUser::GetAuthSessionTicket': function does not take 3 arguments `.

The Steamworks API reference of the GetAuthSessionTicket method states that it indeed takes 4 parameters instead of 3. (Valve Corporation, n.d.)



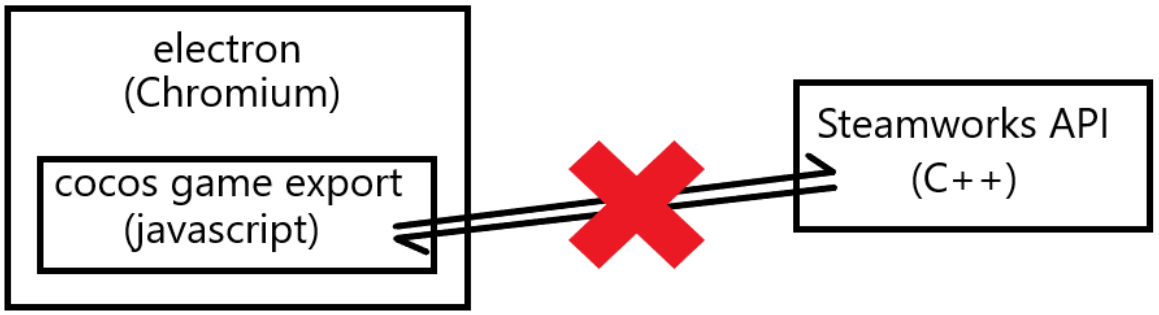
After checking the github page for the latest version of Greenworks, it became clear that the last update to the code was in 2022. There is also a message that says the project will no longer be supported and some links to alternatives were present.

The changes to the Steamworks API were likely made after 2022 and Greenworks was not updated. This meant an alternative way to integrate the Steamworks API into Stratego needed to be implemented.

5.2.2.2 steamworks.js

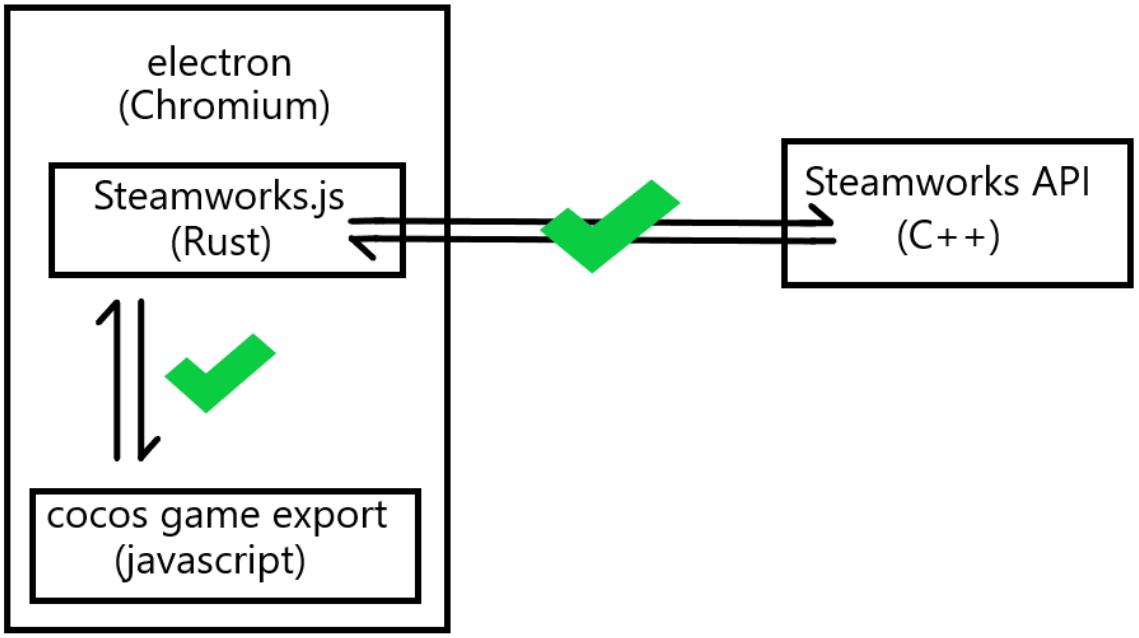
Luckily, other developers also ran into this problem. Liana P. (2022) ran into Greenworks issues and found an alternative called Steamworks.js. They also made a handy tutorial with examples to implement this new tool.

To implement the connection between the game and the steam API, multiple layers needed to communicate with each other The general setup is shown below. Liana P. (2022)



As the figure shows, the communication from JavaScript to the Steamworks API is not possible out of the box. Another layer needs to provide this communication. An existing npm package would offer the ideal solution, due to the ease and speed of implementation. Liana P. (2022)

Steamworks.js fill both these needs. It is written in Rust, but it takes JavaScript function calls. These calls are then converted to c++ calls that trigger the Steamworks API. This library can run in electron alongside the Cocos Project. This setup would look something like this. Liana P. (2022)



The connection between Steamworks.js and the Steamworks API is handled in the Steamworks.js library. So, only the connection between the game and the library needs to be created.

Luckily, electron was built with situations like this in mind. The components that make the Electron app work, like chromium and the connection to the OS, runs on the base layer. The Cocos application that is imported into the electron project runs one layer deeper in the 'renderer'. Events that are thrown from within the renderer can be handled in the base layer with an Electron feature called ipcMain. (*IPCMain | Electron*, n.d.)

Appendix G shows the integration of Steamworks.js in the index.js file of the Electron project. Crucially, the "listenForSteamworks" method, shown in the figure below, instantiates the ipc event handlers.

```
function listenForSteamworks() {
  ipcMain.handle('releaseAchievement', (event, achievementString) => {
    console.warn('releasing achievement', achievementString);

    steamworksClient.achievement.activate(achievementString);
  });

  ipcMain.handle('getAuthToken', async (event) => {
    console.warn('getting auth token');
    console.warn('event: ', event);

    const token = await steamworksClient.auth.getSessionTicket();
    console.warn(token);
  });
}
```

The full Electron index.js file with the Steamworks.js integration is available in Appendix G.

In the figure above, the 'releaseAchievement' and the 'getAuthToken' events are listened to in the base layer. When these events are called from the renderer, the Steamworks.js equivalents are called. This code runs in the index.js of the electron app.

Throwing these events from our Cocos project can be handled best by a static class that invokes the ipc events. By making the class static, extra work, in the form of references, can be avoided while adding API calls to existing code.

A simple class with static methods was set up that can be used in the game whenever interactions with the Steamworks API are required. For this code, please refer to Appendix F.

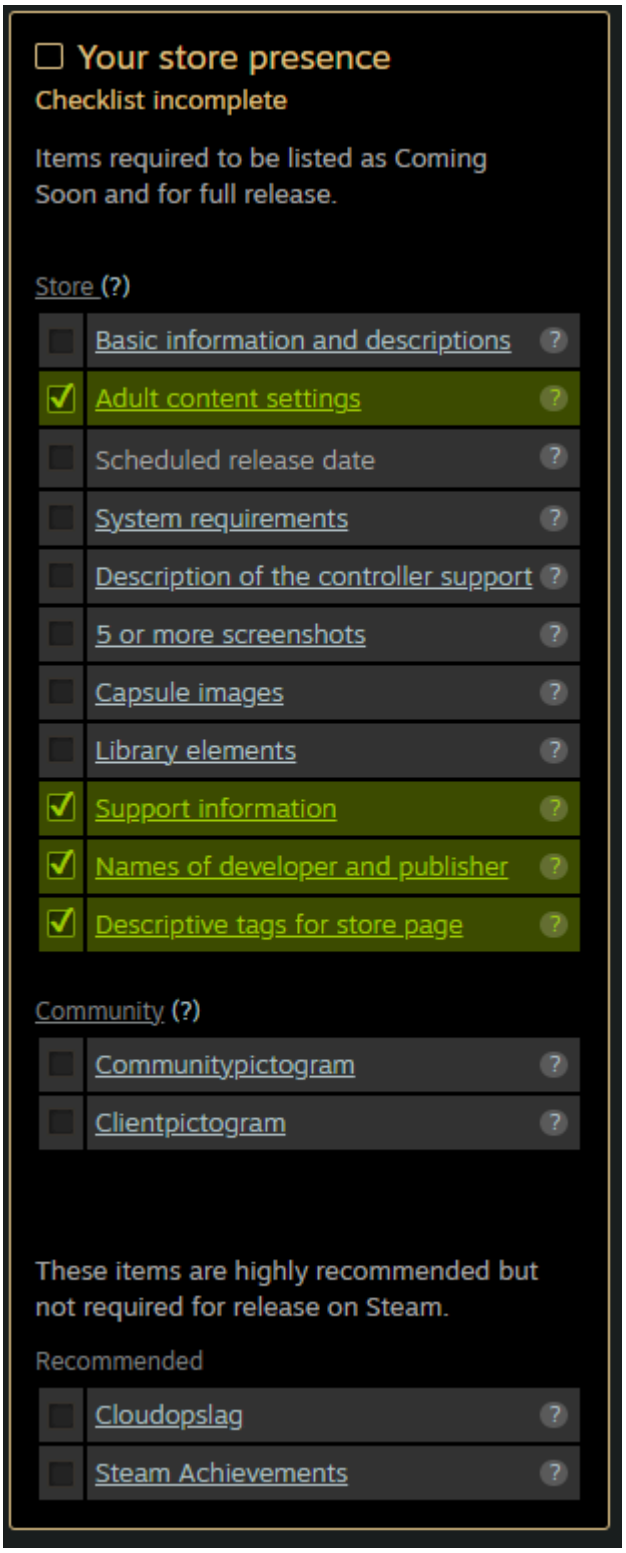
Should an achievement need to be activated, for example, we only need 1 line of code. The figure below shows that simply calling the static method "Steamworks.releaseAchievement()" will result in the achievement to be granted to the players' Steam account.

```
async show(data?: any) {    Unexpected any. Specify a different type.
    if (data?.playBGM !== false) {
        AudioManager.instance.stopAllBGM(['menu']);
        AudioManager.instance.playBGM('menu', 0.5, false, true);
    }

    console.log('aaaa', data);
    this.playerProfileDisplay.setProfile();

    Steamworks.releaseAchievement('Hello_World');
```

This setup makes the addition of a lot of future Steamworks calls easy.

5.3 The store page

A detailed description of setting up the store page is outside the scope of this research paper. This would not add value to this paper since all the information is already clearly displayed in Valve's documentation and on the Steamworks page of the game. (Valve Corporation, n.d.-b)



The bulk of the work for the store page is creating art like: banners, icons and trailers.
This art, together with writing descriptions, is essentially all that is needed to be able to publish the store page to the public.

5.4 Overview

With the ability to upload builds and interact with the Steamworks API, all the platform requirements of a steam game have been fulfilled.

- ☑ ~~be able to run as a standalone app.~~
- ☑ ~~(optional) have the steam API integrated to use platform features.~~
- ☑ ~~A Steamworks account.~~
- ☑ ~~The store page.~~

The current setup can produce builds for the 3 leading PC operating systems: Windows, macOS, and Linux. These builds can then be uploaded to the Steam servers and shipped out to the players.

This workflow (preparing and pushing a web build of a Cocos project to Steam) can be used by Wanted 5 Games to release their Cocos projects to a new, advantageous, platform.

Conclusion

This chapter answers the research question with the information of the previous chapters.

The research question is:
"*What is the optimal platform to release Stratego Online on, and what is the best method for porting the current Cocos Creator project to this platform?*"

The research started with a game distribution platform analysis. This resulted in a list of popular platforms for popular gaming devices. Key data points about each platform provides Wanted 5 Games with an overview of the value that each platform can offer. This data can be reused for future games that might have different target audiences. Comparing the platform data to the needs for Stratego Online, chapter two found that steam appears to offer the most value due to: a big player base, possibly easy port workflow, low risk of bugs, and an audience that fits our demographic.

Next, chapter three looked into what Steam requires for releasing a game on the platform. A new release on Steam has build requirements and there requirements outside the game itself. The game should be a standalone application and have Steamworks API integration. The remaining launch requirements state that a verified Steamworks account and a fully set up store page need to be in place. The store page mostly requires visual assets and text descriptions about the game. This established concrete knowledge about tasks that need to be completed before the launch on steam.

When the requirement of the porting workflow were known, possible porting options were explored in chapter four. This resulted in an overview of the two most popular methods of porting: building a native desktop application, or wrapping the web build using a lightweight standalone browser using a framework like Electron. After implementing both methods using minimal functional requirements it was concluded that the Electron was the best available option for the company. This was due to too many issues and technical shortcomings in the native build option. A Chromium based solution is slightly more work out of the box but minimizes platform specific bugs and allows for easier familiar debugging using the browser console dev tools.

Finally, chapter five concluded with practical descriptions of implementing remaining platform requirements into the chosen Electron solution. This resulted in the following conclusions.
The Steamworks SDK needs to be used to upload builds of the game to the Steamworks servers.
A library like "Steamworks.js" needs to be implemented to call Steamworks API functions from within the game.
Additionally, an overview of the assets and descriptions required to set up the store page was provided.

With the information in chapters four and five, a workflow can be constructed to produce and release builds of Cocos Creator projects to steam.

All these conclusions form the answer to our main research question.
Steam is the optimal platform to release Stratego Online, and the Electron framework is the best method to port the current Cocos Creator project to Steam.

Recommendation

This thesis strongly recommends that Wanted 5 Games proceed with the realization of the Steam port for Stratego Online. Releasing Stratego on Steam will lead to an expanded player base due to access to the large community of Steam.

The research has identified a low-risk and efficient approach to porting Stratego Online to Steam. The existing web build of Stratego can seamlessly be adapted for standalone PC use with Electron. This allows for its deployment onto the Steam platform. This approach minimizes development effort.

For future additions to the Steam build, key Steamworks features should be added such as: the steam friend list, Steam account login, and Steam invite integration. These features are standard expectations among Steam users and can significantly enhance the social and multiplayer aspects of Stratego Online, thereby enriching the overall gaming experience.

Finally, integrating Steam into the Continuous Integration (CI) pipeline will streamline the workflow. By automating the deployment process with CI, Wanted 5 Games can expand the list of active platforms while experiencing a minimal increase in developer workload.

By capitalizing on this opportunity, Wanted 5 Games can unlock more of the potential of Stratego Online and establish a stronger presence within the market.

References

About Jumbo - Jumbo. (2022, October 17). Jumbo. Retrieved February 25, 2024, from https://www.jumbo.eu/en/about-jumbo/

Blancaflor, E., & Miguel, J. M. G. S. (2022). Analyzing Digital Game Distribution in Gaming Industry: A Case Study. *2nd Indian International Conference on Industrial Engineering and Operations Management*. https://doi.org/10.46254/in02.20220232

Bonestroo, W.J., Meesters, M., Niels, R., Schagen, J.D., Henneke, L., Turnhout, K. van (2018): ICT Research Methods. HBO-i, Amsterdam. ISBN/EAN: 9990002067426.

Cocos. (2023, August 14). *GitHub - cocos/cocos-test-projects*. GitHub. Retrieved February 7, 2024, from https://github.com/cocos/cocos-test-projects

Cocos Creator 2.4 Manual - LabelOutline Reference (By Cocos). (n.d.). Retrieved January 26, 2024, from https://docs.cocos.com/creator/2.4/manual/en/components/label-outline.html

Cocos Creator 2.4 Manual - LabelShadow Reference (By Cocos). (n.d.). Retrieved January 26, 2023, from https://docs.cocos.com/creator/2.4/manual/en/components/label-shadow.html

Cocos Creator 3.8 Manual - General Build Options (By Cocos). (n.d.). Retrieved December 13, 2023, from https://docs.cocos.com/creator/manual/en/editor/publish/build-options.html

Cocos Creator 3.8 Manual - General Native Build Options (By Cocos). (n.d.). Retrieved December 14, 2023, from https://docs.cocos.com/creator/manual/en/editor/publish/native-options.html#build-options

Cocos Creator 3.8 Manual - Setting up Native Development Environment (By Cocos). (n.d.). Retrieved December 14, 2023, from https://docs.cocos.com/creator/manual/en/editor/publish/setup-native-development.html#installing-c-compiling-environment

Cocos Creator 3.8 Manual - Tutorial: JSB 2.0 (By Cocos). (n.d.). Retrieved January 10, 2024, from https://docs.cocos.com/creator/manual/en/advanced-topics/JSB2.0-learning.html

Crane, C. (2021, April 28). *What is a CA signed certificate & how do I get one? - Comodo SSL resources*. Comodo SSL Resources. Retrieved February 4, 2024, from https://comodosslstore.com/resources/what-is-a-ca-signed-certificate-how-do-i-get-one/

debugChicken. (2021, May 28). *Tutorial: Transfer your Cocos Creator game to Steam*. Cocos Forums. Retrieved February 7, 2024, from https://discuss.cocos2d-x.org/t/tutorial-transfer-your-cocos-creator-game-to-steam/53694

Dissanaike, S. (2023, June 2). Create a self-signed certificate for testing secure WebSockets in iOS or NodeJS. *Medium*. https://medium.com/himinds/create-a-self-signed-certificate-for-ios-or-nodejs-for-testing-secure-websockets-3df4bc722a3f

Electron. (n.d.). *GitHub - electron/packager: Customize and package your Electron app with OS-specific bundles (.app, .exe, etc.) via JS or CLI*. GitHub. Retrieved February 10, 2024, from https://github.com/electron/packager

Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., & Shmatikov, V. (2012). *The most dangerous code in the world*. https://doi.org/10.1145/2382196.2382204

Getting started (Steamworks documentation) (By Valve Corporation). (n.d.). Retrieved February 15, 2024, from https://partner.steamgames.com/doc/gettingstarted?language=english

Glossary | Electron. (2023). Retrieved January 21, 2024, from https://www.electronjs.org/docs/latest/glossary

Greenheartgames. (n.d.). *GitHub - greenheartgames/greenworks: a node.js plugin to integrate nw.js/electron games with steamworks*. GitHub. Retrieved February 27, 2024, from https://github.com/greenheartgames/greenworks

Introduction | Electron (By OpenJS Foundation). (2023). Retrieved January 19, 2024, from https://www.electronjs.org/docs/latest

IPCMain | Electron (By OpenJS Foundation). (n.d.). Retrieved March 2, 2024, from https://www.electronjs.org/docs/latest/api/ipc-main

Kingurus. (2023, September 25). *Label Outline and Shadow do not work on native*. GitHub. Retrieved January 15, 2024, from https://github.com/cocos/cocos-engine/issues/16331

Liana P. (2022, July 17). *How to integrate an HTML5 Electron based game with the Steam API*. Articles by Liana P. Retrieved February 28, 2024, from https://liana.one/integrate-electron-steam-api-steamworks

Nakama. (n.d.). *Sockets*. Heroic Labs Documentation. Retrieved January 27, 2024, from https://heroiclabs.com/docs/nakama/concepts/sockets/

Obias, R. (2024, March 11). The 7 best video game consoles to buy in 2024. *Rolling Stone*. Retrieved November 6, 2023, from https://www.rollingstone.com/product-recommendations/electronics/best-video-game-consoles-1234977597/

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency. *Science of Computer Programming*, *205*, 102609. https://doi.org/10.1016/j.scico.2021.102609

rui-gameking. (2023, September 13). *LabelOutline and LabelShadow are not working on native yet*. Cocos Forums. Retrieved January 23, 2024, from https://discuss.cocos2d-x.org/t/labeloutline-and-labelshadow-are-not-working-on-native-yet/59412

Sinclair, B. (2018, December 4). Epic launching Steam rival with 88% revenue share for developers. *GamesIndustry.biz*. Retrieved November 12, 2023, from https://www.gamesindustry.biz/epic-launching-store-with-88-percent-revenue-share-for-developers

Steamworks Development. (2016, October 5). *SteamWorks Tutorial #4 - Adding new platforms and languages* [Video]. YouTube. Retrieved February 25, 2024, from https://www.youtube.com/watch?v=PShS32hcing

Steamworks documentation. (2024). Retrieved November 21, 2023, from https://partner.steamgames.com/doc/home

SteamWorks SDK (Steamworks Documentation). (n.d.). Retrieved February 23, 2024, from https://partner.steamgames.com/doc/sdk

Steamworks (By Valve Corporation). (2024). Steamworks. Retrieved November 20, 2023, from https://partner.steamgames.com/

Stratego - Jumbo. (2024). Jumbo. Retrieved February 25, 2024, from https://www.jumbo.eu/merken/stratego/

theamirbeg. (2023, March 6). *[Cocos Creator 3.7.1] .ttf Fonts Not Working in Simulator on Windows Platform*. Cocos Forums. Retrieved January 23, 2024, from https://discuss.cocos2d-x.org/t/cocos-creator-3-7-1-ttf-fonts-not-working-in-simulator-on-windows-platform/58335

Unity Technologies. (2024, March 14). *Unity - Manual: Build Settings*. Retrieved December 5, 2023, from https://docs.unity3d.com/Manual/BuildSettings.html

Uploading to Steam (Steamworks documentation). (n.d.). Retrieved February 19, 2024, from https://partner.steamgames.com/doc/sdk/uploading

Valve Corporation. (n.d.-a). *SteamWorks API Overview (SteamWorks Documentation)*. Retrieved February 27, 2024, from https://partner.steamgames.com/doc/sdk/api

Valve Corporation. (n.d.-b). *Store Presence (Steamworks documentation)*. Retrieved March 4, 2024, from https://partner.steamgames.com/doc/store

Van Leeuwen, H. (2011). *De computer verdwijnt, lang leve ambient intelligence!* [Saxion]. https://www.narcis.nl/publication/RecordID/oai:hbokennisbank.nl:saxion_kenniscentra:DF389BF8351B0DB8C12579420029155E

Van Turnhout, K., Köppe, C., Tankink, T., Schuszler, P., & Bakker, R. (2018). RESEARCH EDUCATION NURTURES INQUISITIVENESS OF PROFESSIONAL DESIGN AND ENGINEERING STUDENTS. *DS 93: Proceedings of the 20th International Conference on Engineering and Product Design Education (E&PDE 2018), Dyson School of Engineering, Imperial College, London. 6th - 7th September 2018*, 770–775. https://www.designsociety.org/download-publication/40864/RESEARCH+EDUCATION+NURTURES+INQUISITIVENESS+OF+PROFESSIONAL+DESIGN+AND+ENGINEERING+STUDENTS

Wanted 5 Games - HTML5 Game Studio. (2021). Retrieved February 25, 2024, from https://wanted5games.com/

Appendix A. Raw data - Platform analysis

Raw data results of the data collection for the platform analysis. Original datasheet can be found here:
https://docs.google.com/spreadsheets/d/18C3-LMsQAyMm219hlan27z72u0m1PUuz/edit?usp=sharing&ouid=114286752083020470263&rtpof=true&sd=true

Metrics 1-8:

		1. Platform Reach: monthly userbase	2. Platform Reach: Global Presence	3. Demographics: Age	4. Demographics: User Engagement and Retention (per game)	5. Demographics: Similar games perform well on platform. (turn based strategy board games)	6. Monetization: no mid-roll ads.	7. Monetization: Option to use own payment system.	8. Monetization: Profitable revenue share. platform/us
	distributor								
PC	Steam	132.000.000 MAU	yes (see figure 1)	13-99	likely to be invested	Similar games perform well	No advertisements	Can only use wallet offered by platform	30/70 (first 10M$)
	Epic Games	56.000.000 MAU	yes (see figure 2)	3-99	likely to be invested	Genre mildly popular	No advertisements	Any payment system	12/88
	GOG.com	13.500.000 visits	yes (see figure 3)	18-99	likely to be invested	Strategy games are popular	No advertisements	Any payment system	30/70
Mobile	App store	*652.000.000 gamers	*yes	3-99	might play a dozen times	Similar games perform well	No advertisements	Platform's payment system needs to be implemented	30/70
	Play store	*1.978.940.000 gamers	*yes	3-99	might play a dozen times	Similar games perform well	No advertisements	Platform's payment system needs to be implemented	15/85 (first 1M$)
Web	Kongregate	3.300.000 views	mostly western countries (see figure 4)	3-24	might play a dozen times	Not a popular genre	Only preroll advertisements	Platform's payment system needs to be implemented	30/70
	Armor Games	2.800.000 views	mostly western countries (see figure 5)	3-24	might play a dozen times	Not a popular genre	Only preroll advertisements	Platform's payment system needs to be implemented	Game by game basis

	Poki	30.000.000 views	yes (see figure 6)	3-13	mostly plays one time	Not a popular genre	Only preroll advertisements	tbd	Game by game basis
	CrazyGames	20.000.000 views	yes (see figure 7)	3-13	mostly plays one time	Not a popular genre	Only preroll advertisements	Platform's payment system needs to be implemented	Game by game basis
	SPIL Games portals (Azerion)	40.000.000 view	mostly Dutch and USA	3-13	mostly plays one time	Not a popular genre	Only preroll advertisements	Any payment system	Game by game basis
	Admeen (Azerion)	40000000 view	mostly Dutch and western countries	3-13	mostly plays one time	Not a popular genre	Only preroll advertisements	Any payment system	Game by game basis
Console	PlayStation Store	107.000.000 MAU	yes (see figure 8)	3-99	likely to be invested	Genre mildly popular	No advertisements	Any payment system	30/70
	Xbox Store	120.000.000 MAU	yes (see figure 9)	3-99	likely to be invested	Genre mildly popular	No advertisements	Platform's payment system needs to be implemented	30/70
	Nintendo eShop	36000000 MAU	yes (see figure 10)	3-99	likely to be invested	Similar games perform well	No advertisements	Platform's payment system needs to be implemented	30/70

Metrics 9-14:

		9. Promotion: Possibilities for features/favorites.	10. Distribution Channels: Channels allign with target audience.	11. Development Cost: Ease of creating platform specific build.	12. Development Cost: Risk of platform specific bugs.	13. Social Features: Users can see their friends play.	14. Social Features: Option to log in with platform account.
	distributor						
PC	Steam	Editor features possible if game becomes popular + promotion through sales (base game/DLCs)	Purely digital store on gaming device	Build needs to be able to run on PC operating systems. The more native this solution becomes the harder	Low/High, Higher risk comes with rewrites in new engines/seperate codebases	Yes	Yes

	Epic Games	Editor features possible if game becomes popular + promotion through sales (base game/DLCs)	Purely digital store on gaming device	Build needs to be able to run on PC operating systems. The more native this solution becomes the harder	Low/High, Higher risk comes with rewrites in new engines/seperate codebases	Yes	Yes
	GOG.com	Editor features possible if game becomes popular + promotion through sales (base game/DLCs)	Purely digital store on gaming device	Build needs to be able to run on PC operating systems. The more native this solution becomes the harder	Low/High, Higher risk comes with rewrites in new engines/seperate codebases	Yes	No
Mobile	App store	Editor features possible if game becomes popular + paid promotion	Purely digital store on gaming device	Build for this platform is already available	Medium, hardware specific issues on IOS come up often	Yes	Yes
	Play store	Editor features possible if game becomes popular + paid promotion	Purely digital store on gaming device	Build for this platform is already available	Medium, hardware specific issues on Android will come up during development	Yes	Yes
Web	Kongregate	Editor features possible if game becomes popular	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	Yes	Yes
	Armor Games	Editor features possible if game becomes popular + "new games" feature for best new games	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	No	Yes
	Poki	Editor features possible if game becomes popular	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	No	No
	CrazyGames	Editor features possible if game becomes popular	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	No	Yes
	SPIL Games portals (Azerion)	Editor features possible if game becomes popular	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	No	Yes

	Admeen (Azerion)	Editor features possible if game becomes popular	Web game portal	Build for this platform is already available	Very low, SDK issues are possible	No	No
Console	PlayStation Store	High quality popular games can get featured	Big digital store + physical stores	Request Devkit + remake game in engine that exports to C++	Very High, different hardware and codebase is very susceptible to bugs	Yes	Yes
	Xbox Store	High quality popular games can get featured	Big digital store + physical stores	Request Devkit + remake game in engine that exports to C++ or C#	Very High, different hardware and codebase is very susceptible to bugs	Yes	Yes
	Nintendo eShop	High quality popular games can get featured	Big digital store + physical stores	Request Devkit + remake game in engine that exports to C++	Very High, different hardware and codebase is very susceptible to bugs	Yes	Yes

Appendix B. Optimal results for valuable platform - Platform analysis

The table below shows the data for the hypothetical, optimal platform to launch Stratego Online on. Original datasheet can be found here: https://docs.google.com/spreadsheets/d/18C3-LMsQAyMm219hlan27z72u0m1PUuz/edit?usp=sharing&ouid=114286752083020470263&rtpof=true&sd=true

Metrics 1-8:

1. Platform Reach: monthly userbase	2. Platform Reach: Global Presence	3. Demographics: Age	4. Demographics: User Engagement and Retention (per game)	5. Demographics: Similar games perform well on platform. (turn based strategy board games)	6. Monetization: no mid-roll ads.	7. Monetization: Option to use own payment system.	8. Monetization: Profitable revenue share. platform/us
Bigger is better	Should firstly target known markets (NL, DE, US, etc.). reach outside these countries is a bonus.	The recommended age of the Stratego demographic is 8 and up. We have written in our terms of service that the game is intended for ages 13 and up.	High retention is important to build the community and fill ranks.	Good fit is better.	Forced mid-roll ads are a dealbreaker.	Own payment system is preferred.	A higher share of the revenue is better.

Metrics 9-14:

9. Promotion: Possibilities for features/favorites.	10. Distribution Channels: Channels allign with target audience.	11. Development Cost: Ease of creating platform specific build.	12. Development Cost: Risk of platform specific bugs.	13. Social Features: Users can see their friends play.	14. Social Features: Option to log in with platform account.
Possibilities for features for smaller (non AAA) titles is great to introduce new	We want users to have the easiest time finding and playing our game. The fewer extra steps like logins and downloads the better.	Lower cost is better.	Lower risk is better.	We hope that seeing friends play Stratego Online will encourage people to play too.	Platform account integration is positive for user experience.

players to Stratego Online.					

Appendix C. Platform comparison - Platform analysis

The raw results of comparing the data from Appendix A. with the data from Appendix B.Original datasheet can be found here: https://docs.google.com/spreadsheets/d/18C3-LMsQAyMm219hlan27z72u0m1PUuz/edit?usp=sharing&ouid=114286752083020470263&rtpof=true&sd=true

Metrics 1-8:

	distributor	1. Platform Reach: monthly userbase	2. Platform Reach: Global Presence	3. Demographics: Age	4. Demographics: User Engagement and Retention (per game)	5. Demographics: Similar games perform well on platform. (turn based strategy board games)	6. Monetization: no mid-roll ads.	7. Monetization: Option to use own payment system.	8. Monetization: Profitable revenue share. platform/us
PC	Steam	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players have high average retention.	Multiple other similar games perform well on this platform	No advertisements	Can only use wallet offered by platform	30/70 (first 10M$)
	Epic Games	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players have high average retention.	Similar games exist but are not popular on this platform	No advertisements	Any payment system	12/88
	GOG.com	Small/Medium userbase	Good reach in all regions	Platform mostly fits the target age of Stratego	Players have high average retention.	Strategy genre is popular on this platform	No advertisements	Any payment system	30/70
Mobile	App store	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players are not likely to stick to new games	Multiple other similar games perform well on this platform	No advertisements	Platform's payment system needs to be implemented	30/70
	Play store	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players are not likely to stick to new games	Multiple other similar games perform well on this platform	No advertisements	Platform's payment system needs to be implemented	15/85 (first 1M$)

Web	Kongregate	Big userbase	Good reach in some of the target regions (US) + some reach in western countries	Platform mostly fits the target age of Stratego	Players are not likely to stick to new games	Strategy board games is not a popular genre on this platform	Only preroll advertisements	Platform's payment system needs to be implemented	30/70
	Armor Games	Medium userbase	Good reach in some of the target regions (US) + some reach in western countries	Platform mostly fits the target age of Stratego	Players are not likely to stick to new games	Strategy board games is not a popular genre on this platform	Only preroll advertisements	Platform's payment system needs to be implemented	Game by game basis
	Poki	Big userbase	Good reach in all regions	Platform users are younger than target age of Stratego	Players will likely only play a couple times	Strategy board games is not a popular genre on this platform	Only preroll advertisements	tbd	Game by game basis
	CrazyGames	Medium userbase	Good reach in all regions	Platform users are younger than target age of Stratego	Players will likely only play a couple times	Strategy board games is not a popular genre on this platform	Only preroll advertisements	Platform's payment system needs to be implemented	Game by game basis
	SPIL Games portals (Azerion)	Big userbase	Good reach in the target regions	Platform users are younger than target age of Stratego	Players will likely only play a couple times	Strategy board games is not a popular genre on this platform	Only preroll advertisements	Any payment system	Game by game basis
	Admeen (Azerion)	Big userbase	Good reach in the target regions	Platform users are younger than target age of Stratego	Players will likely only play a couple times	Strategy board games is not a popular genre on this platform	Only preroll advertisements	Any payment system	Game by game basis
Console	PlayStation Store	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players have high average retention.	Strategy genre is popular on this platform	No advertisements	Can only use wallet offered by platform	30/70
	Xbox Store	Very big userbase	Good reach in all regions	Platform fits the target age of Stratego	Players have high average retention.	Strategy genre is popular on this platform	No advertisements	Can only use wallet offered by platform	30/70

	Nintendo eShop	Big userbase	Good reach in the target regions	Platform fits the target age of Stratego	Players have high average retention.	Multiple other similar games perform well on this platform	No advertisements	Can only use wallet offered by platform	30/70

Metrics 9-14:

	distributor	9. Promotion: Possibilities for features/favorites.	10. Distribution Channels: Channels allign with target audience.	11. Development Cost: Ease of creating platform specific build.	12. Development Cost: Risk of platform specific bugs.	13. Social Features: Users can see their friends play.	14. Social Features: Option to log in with platform account.
PC	Steam	Low/medium chance of features for smaller games + promotion through sales	Platform account and download required	PC port required, can be done easily through soluition like electron or hard thorugh full rewrite.	Low/High risk depending on easy/hard development method. (See 'Development Cost' metric)	Yes	Yes
	Epic Games	Low chance of features for smaller games + promotion through sales	Platform account and download required	PC port required, can be done easily through soluition like electron or hard thorugh full rewrite.	Low/High risk depending on easy/hard development method. (See 'Development Cost' metric)	Yes	Yes
	GOG.com	High chance of promotion for smaller games	Platform account and download required	PC port required, can be done easily through soluition like electron or hard thorugh full rewrite.	Low/High risk depending on easy/hard development method. (See 'Development Cost' metric)	Yes	No
Mobile	App store	Low chance of features for smaller games + paid promotion	Platform account and download required	Build is already included in workflow.	Medium risk, hardware specific issues often pop up on this platform	Yes	Yes
	Play store	Low chance of features for smaller games +	Platform account and download required	Build is already included in workflow.	Medium risk, hardware specific issues often pop up on this platform	Yes	Yes

		paid promotion					
Web	Kongregate	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	Yes	Yes
	Armor Games	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	No	Yes
	Poki	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	No	No
	CrazyGames	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	No	Yes
	SPIL Games portals (Azerion)	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	No	Yes
	Admeen (Azerion)	High chance of features for smaller games	Web portal, no downloads required	Build is already included in workflow.	Very low risk, SDK issues might be possible	No	No
Console	PlayStation Store	Low chance of features for smaller games	Console, platform account and download required.	Requires requesting devkit and rewriting game	Very high risk, different hardware and codebase is very susceptible to bugs	Yes	Yes
	Xbox Store	Low chance of features for smaller games	Console, platform account and download required.	Requires requesting devkit and rewriting game	Very high risk, different hardware and codebase is very susceptible to bugs	Yes	Yes
	Nintendo eShop	Low chance of features for smaller games	Console, platform account and download required.	Requires requesting devkit and rewriting game	Very high risk, different hardware and codebase is very susceptible to bugs	Yes	Yes

Appendix D. Steam platform requirements

The “game build” and “store presence” as displayed on the Steamworks page of Stratego Online.

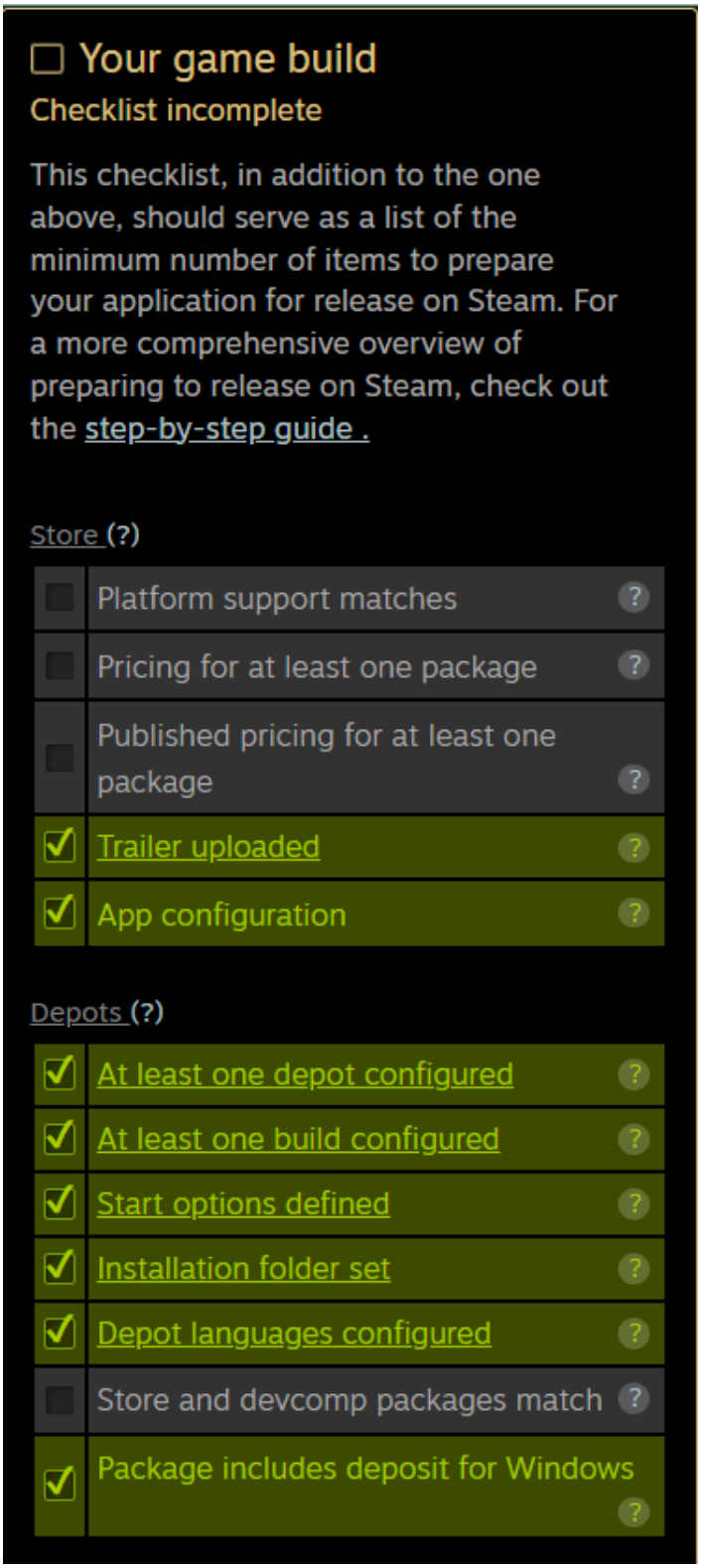


Figure 1. “game build” requirements

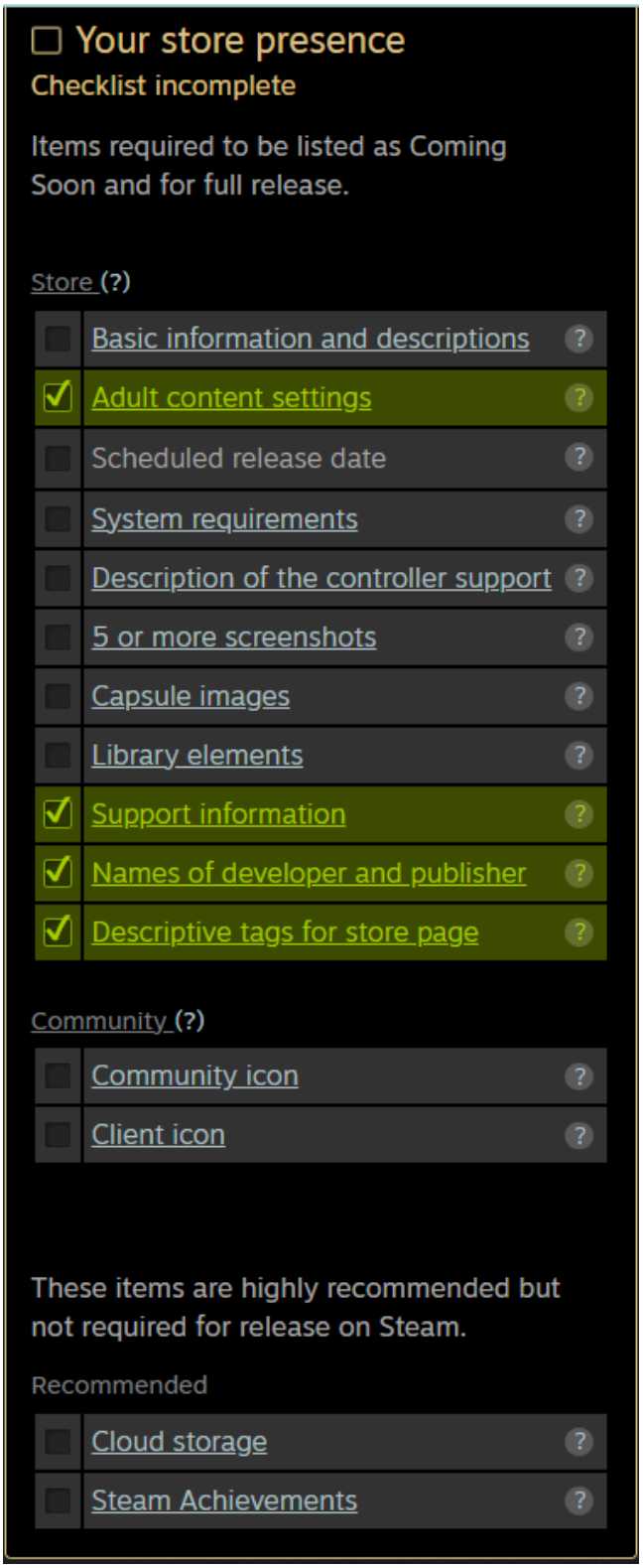


Figure 2. “Store presence” requirements

Appendix E. Boilerplate Electron index.js file

```
const { app, BrowserWindow } = require('electron')

app.on('window-all-closed', function() {
  if (process.platform != 'darwin')
    app.quit();
});

app.on('ready', function() {
    mainWindow = new BrowserWindow({
      width: 768,
      height: 768,
      show: true,
      fullscreen: true,
      resizable: false,
      frame: true,
      title: "MyGame",
    });

    mainWindow.removeMenu();

    mainWindow.on('closed', function() {
      mainWindow = null;
    });
});
```

Appendix F. Steamworks Implementation in Stratego Online Code

```
export default class Steamworks
{
    static releaseAchievement(id: string): void
    {
        if(!this.isSteam()) return;

        (window as any).electron.ipcRenderer.invoke('releaseAchievement', id);
    }

    static isSteam(): boolean{
        return (window as any).electron;
    }

    static getAuthToken(): string{
        if(!this.isSteam()) return;

        (window as any).electron.ipcRenderer.invoke('getAuthToken');
        return '';
    }
}
```

Appendix G. Steamworks implementation in Electron

```
const { app, BrowserWindow } = require('electron')

const steamworks = require('steamworks.js')
const steam_appid = 2660420
const steamworksClient = steamworks.init(steam_appid)
console.log(steamworksClient.localplayer.getName())

const ipcMain = require('electron').ipcMain;

app.on('window-all-closed', function() {
  if (process.platform != 'darwin')
    app.quit();
});

app.on('ready', function() {
    mainWindow = new BrowserWindow({
      width: 768,
      height: 768,
      show: true,
      fullscreen: true,
      resizable: false,
      frame: true,
      title: "Stratego Online",
      webPreferences: {
        devTools: true,
        contextIsolation: false,
        nodeIntegration: true,
        sandbox: false
      }
    });

    mainWindow.removeMenu();

    mainWindow.on('closed', function() {
      mainWindow = null;
    });

    mainWindow.loadURL('file://' + __dirname + '/web-mobile/index.html');

    listenForSteamworks();
});

function listenForSteamworks() {
  ipcMain.handle('releaseAchievement', (event, achievementString) => {
    console.warn('releasing achievement', achievementString);

    steamworksClient.achievement.activate(achievementString);
  });

  ipcMain.handle('getAuthToken', async (event) => {
    console.warn('getting auth token');
    console.warn('event: ', event);

    const token = await steamworksClient.auth.getSessionTicket();
    console.warn(token);
  });
}
```

```
require('steamworks.js').electronEnableSteamOverlay()
```