Creation and Implementation of Tile Visualisation Software

Abstract

Software for assisting customers in visualising items bought online have become more and more valuable to commercial websites. Most notably for industries where the look and feel of the product they are selling is part of what makes it appealing. The tiling industry is one such area and many companies have made attempts to create visualisation tools for their customers. This report outlines the implementation and testing of a commercial app to visualise tiles for the company Tile Flair.

Introduction

Most tile sales companies have some form of visualisation tool to help sell their tile designs. The intention with these tools is to help customers imagine how their selected tiles may look in their own kitchen or bathroom. The ability for customers to imagine their dream room and how the tile sales company can help make this a reality is exactly why tile visualisation tools can be an important part of the sales journey. This report covers the research and development of a multiplatform app which allows a user to provide details of their room and then visualise their chosen tiles on the walls of that room.

The project itself is the result of a request from tile sales company Tile Flair. In which it was requested that a visualisation tool be made to exhibit their range of tiles in a way that can account for the details of a customer's room.

Research

Research into the market and what was available in terms of tile visualisation showed that the current tools from most tile companies were woefully inadequate. (Topps Tiles, 2020), (Tile Mountain, 2020) and (Tile Warehouse, 2020) are all examples of tile visualisation tools that seemingly appear to do the bare minimum when it comes to tile visualisation. Instead of allowing a customer to customise in any fashion, these tools simply take a stock computer generated image and place the image of the users selected tile onto the walls of that image. Some allow for different patterns for selected tiles to be placed but most do not even have this functionality.

Figure 1 Topps Tiles Visualiser

In addition to this, most of the tools had selection methods for specific tiles that were difficult to use or understand. With categories that are unclear or some with no categorisation at all, giving the impression that no UX research was done on these tools to ensure that potential customers are understanding what they are doing.

(CTD Tiles, 2020) Is an example of a tile visualisation tool that does allow for a large amount of customisability. However, this example is for more complicated to use than any of the other tools. There are no clear instructions within the tool on how to use it, meaning a first try can prove difficult to visualise tiles in a desired room. Tile designs are completely uncategorised and there is no clear direction on how to add additional walls beyond the four walls that are given at the start. This appears not to be tool designed for a customer to explore and play with but for a trade professional to use that has experience with the tool already and has used multiple times before learning how it can be best used.

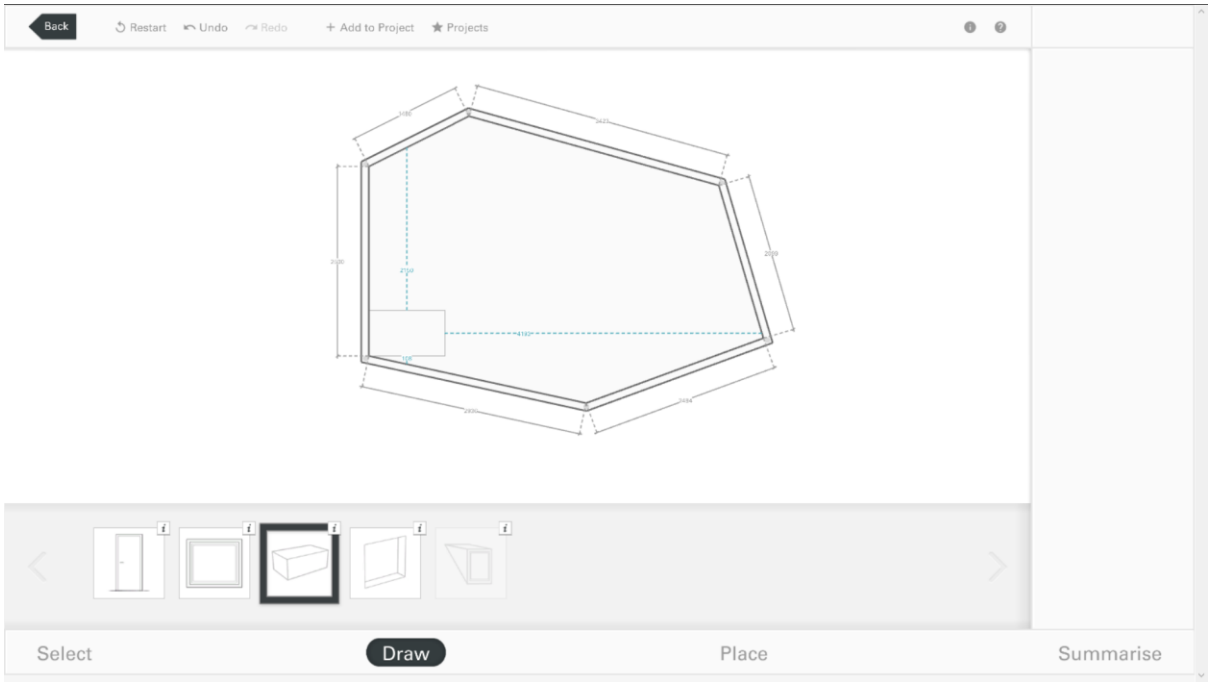

Figure 2 Gemini Blueprint tool from CTD Tiles

This research showed that there are a lot of areas that the tile industry in general could improve upon when it came to the design of a tile visualisation app. Games could also influence this, “The Sims 4” (Electronic Arts, 2014) could be argued to be one of the simplest and most fun tile visualisation tools available. The controls and interface have been carefully crafted to be both easy and fun to use while still allowing players to create rooms that are somewhat accurate to their desired rooms or entire buildings.

Figure 3 Example of the building tools in "The Sims"

To help establish objectives and ensure that the app provided was what the client wished for, a meeting was set up with Matthew Johnson, the managing director of Tile Flair.

Meeting

When meeting with the client, one example of an app they asked to replicate was the Dulux app for mobile devices. This app was impressive as it allowed users to take any picture of any wall and apply a desired colour of paint to that wall. The client asked if something similar could be done for tiles, where a user could take a picture of a wall and place their desired tile on that wall, taking into account light sources and shading. The client stated that the importance was about the colour, shading, how that tiles look in the room and most importantly that it be about the customers room. While the desired output is achievable when using paint colours, tiles demand much more detail especially when meeting these objectives. Tiles require accommodations to the material of the tile along with light sources and other factors. It is because of these factors that it was decided that implementing a fuller creation suite would be a much better idea, as any app that made use of images would be much harder to determine the desired details.

Aims

Through the meeting with the client and the research into other works, clear objectives were laid out that needed to be followed in order to measure the success of the app. There were four objectives in total, two devised from the prior research done and two that came out of the meeting with the client.

The first objective from the research was to “simplify what is out there”. Many of the tile visualisation tools that are currently available suffer from the fact that they have interfaces that do not seem to be researched for ease of use. Most having complicated selection menus for different features and no way to search for a specific tile set. Some tools even seem not to be designed for customers but for tradespeople looking to plan out a design for work. The desired outcome should be an app that is much easier to use than the other options.

The second objective from the research was to “make it more powerful than what is out there”. Most tools found in research were very similar to (Topps Tiles, 2020) where stock images of tiles would be simply placed onto the walls of stock computer generated imagery of bathrooms and kitchens. The app should allow the user to create a much more custom experience.

Out of the meeting, the first objective that was established was that the experience was “Essential that it be about customers room”. As much of the research showed, most tools available do not put the customer’s room first and only allow them to imagine on at best a similar looking room. Like the second research objective, the app needs to be one that allows a Tile Flair customer to see how their own room would look with their desired tiles and that is essential to the experience.

The last objective to come out of the meeting was that “The Importance is colour, shading, how the tiles actually look in the room”. Again, stressing the importance of the customer experience and that it show how the tiles look in the customers room. Many tools offer realistic lighting in their images, but these are pre-baked, raytraced images that cannot be replicated easily for any room. However, modern game engines allow for realistic lighting to be possible on relatively low-end hardware.

Implementation

When developing the app, these four objectives were referred to throughout the process. To help keep the implementation simple, the design was sectioned off into four parts for the user experience: Plan, Place, Paint and View.

The “Plan” section of the app is designed to be the part where the user places the walls for their desired room. This took inspiration from the popular game “The Sims” (Electronic Arts, 2014) where walls are placed on a grid structure and are updated with player mouse movements. The app has a similar structure where the user simply has to click and drag where their wall starts and ends within a grid area with the camera being a top down, orthographic view. The grid itself is on 1-meter units so help achieve the objective of simplicity while still allowing a user to fully customise their room.

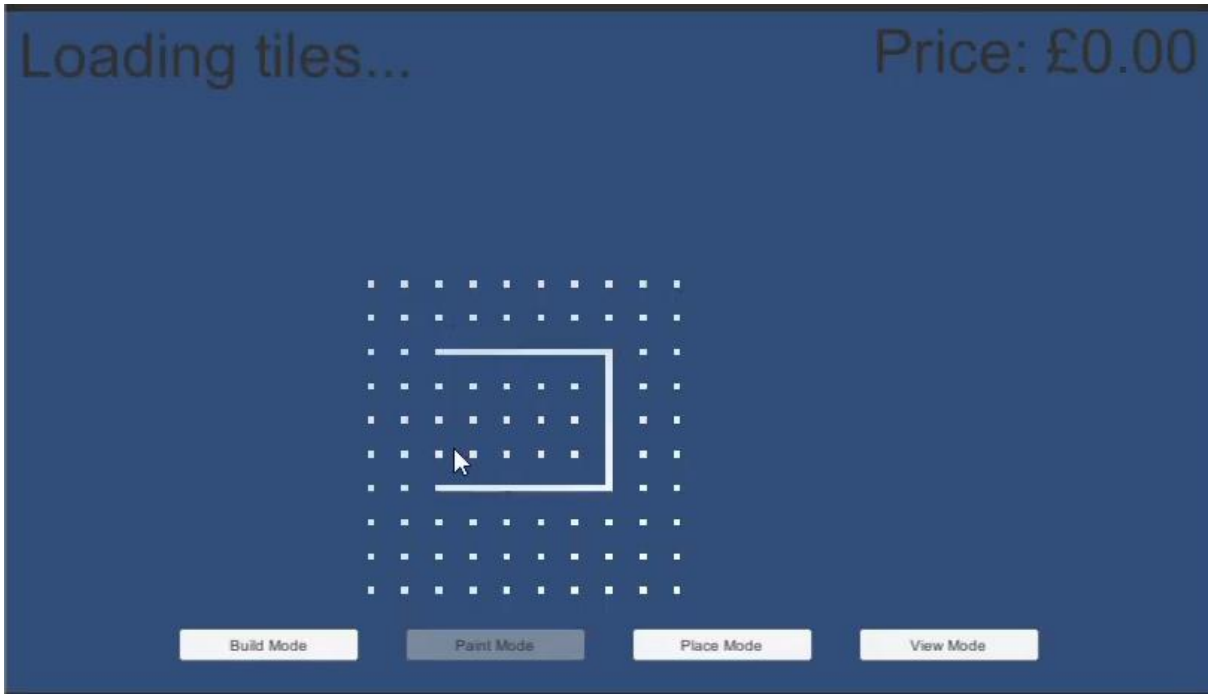

Figure 4 Planning out walls in the app

“Place” allows the user to place objects within the room that they have constructed. In the current implementation this is only counter tops, a fuller implementation would allow for a selection of items such as sinks, light sources and windows. The user simply has to right click and hold to place their desired object within the scene. They can click and drag with the left mouse button to move the camera around the scene. With objects now placed in the scene, the user can go on to the paint mode.

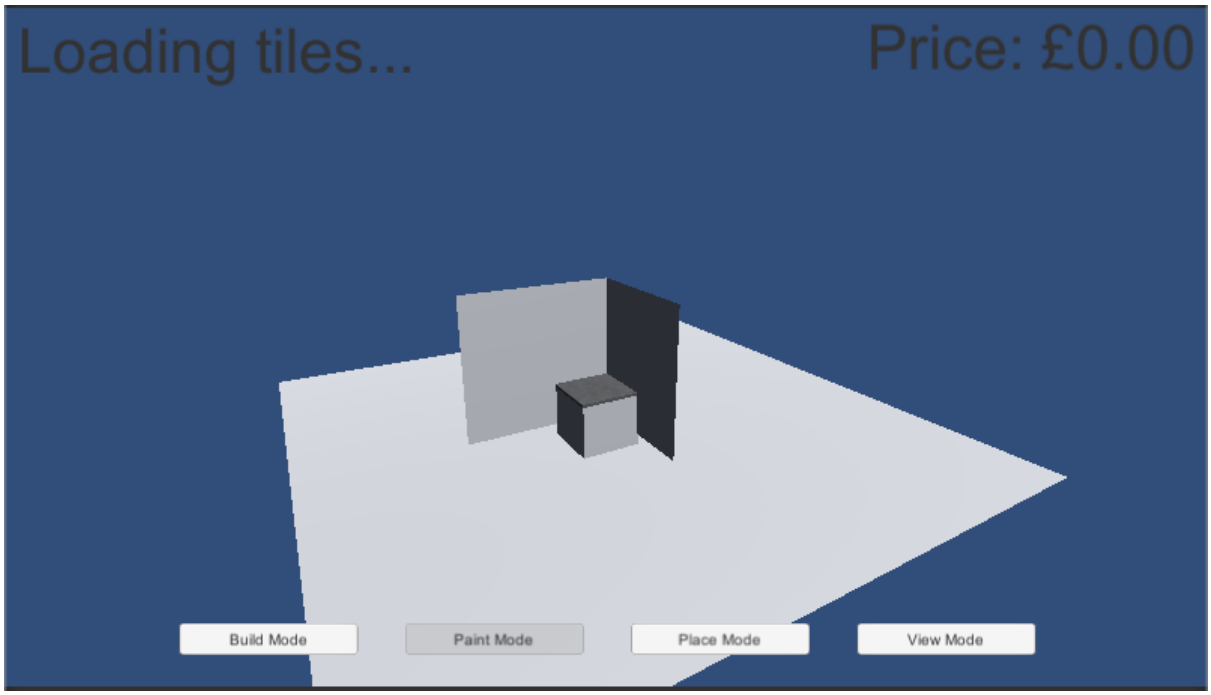

Figure 5 Placing objects within a scene

“Paint” allows the user to paint their selected tile onto each individual wall. The user can cycle through each individual wall which they will see from a side on orthographic perspective, then select their desired tile design and paint a rectangular area that they wish to place the tiles on. In the current implementation, there are only 3 types of tile available, a fuller implementation would allow for all types of tile from the tile flair website to be made available. With this done, the user can now view their room.

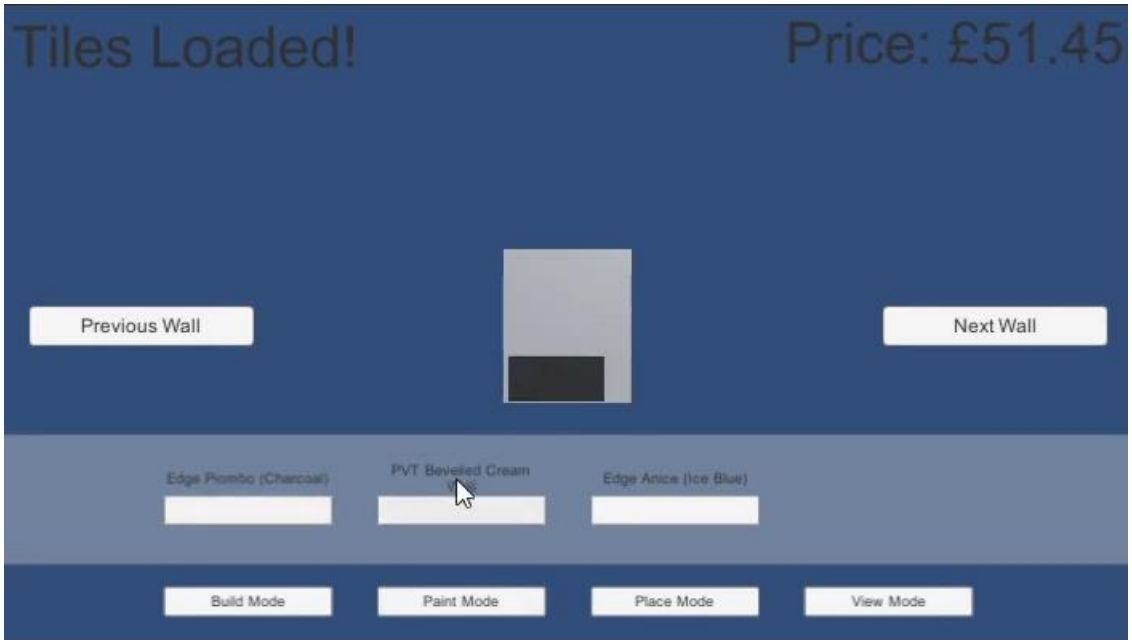

Figure 6 Painting tiles onto individual walls

The final stage "View" allows the user to see their finished room. The controls for this are very similar to the "Place" mode, where the user can move around the room in a 3D perspective by clicking and dragging with the left mouse button. A fuller implantation of this mode could possible allow for the use of a VR headset to view the room from an actual human perspective.

To create tile textures that look realistic according to light sources and the environment of the room, the idea of generating normal maps for each tile was explored and experimented with. There are various methods available for achieving this within Unity including a simple free plugin from the asset store that allows this to be done within runtime. The plugin itself generates a basic normal map, however this took at least 10 seconds to load during runtime on development hardware which was significantly more powerful than the expected hardware of tablets and phones. Such a long load time would be significantly longer on the desired hardware. As a result, it was decided not to use the plugin and instead to look for other methods that worked outside of run time.

Unity's own texture importer tools could generate a normal map for a given texture. However, the generated normal map came out very distorted and did not really generate anything other than what could be described as a bumpy texture. It was after this that using external tools was investigated. GNU Image Manipulation Program is a free, open source image editing software that has the option to generate normal maps from a given image using various techniques. Unfortunately, this produced a normal map that was just as distorted as the output from Unity. At this stage, it was apparent that the problem was more likely to do with the source image as the distortion seemed to reflect jpg compression upon further inspection. With this being the case, it was concluded that it be better not to have normal maps be generated for the tiles and instead to simply go with a two dimensional texture, as any generated normal map would simply make a tile look "dirty".

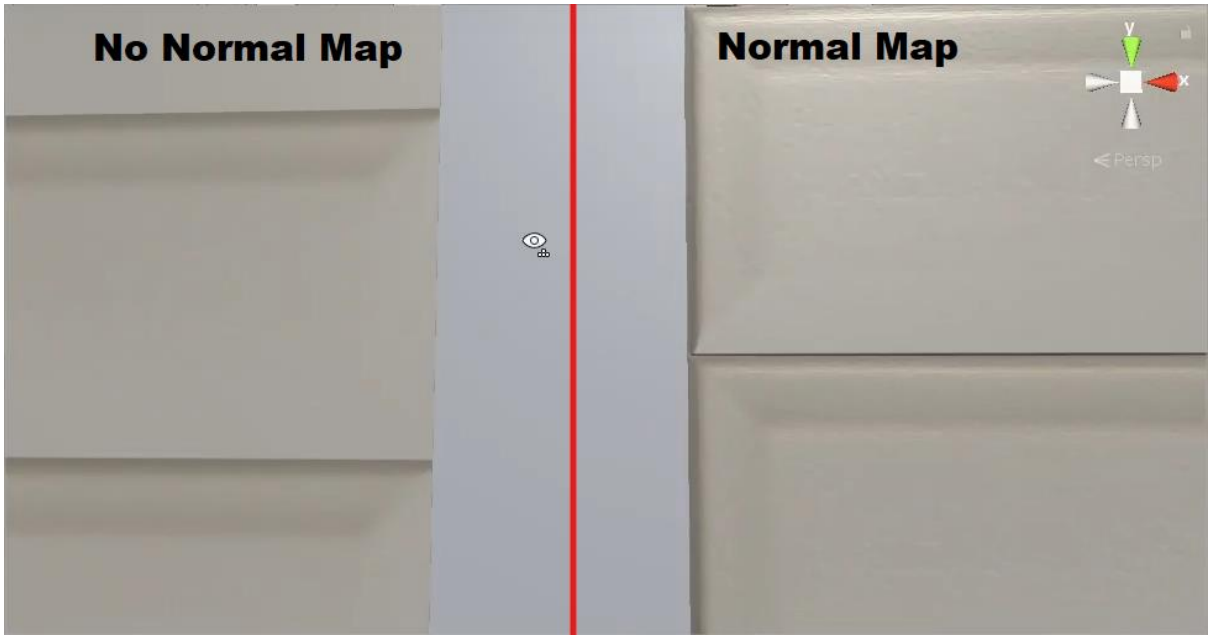

Figure 7 The difference between a tile with a generated normal map and a tile without a normal map

Tiles are stored within the app in their own class with various information stored such as name, height, width and other information. Originally, the implementation was that information would be stored in a CSV file which the app would load when starting up and use this to populate the tile info for each individual tile type. This implementation would mean that any changes to tiles could be made and added to the CSV file without having to rebuild the entire program. However, Compatibility issues occurred when building for both WebGL and Android apps. Android could not easily access the CSV file and the WebGL could not access the file at all since WebGL has no support for local storage. Having encountered these problems, an alternative was devised that did in fact produce better results that are more adaptable to changes in tiles and their information.

Through making use of Unity's web page tools, a system was created in which when given the URL of the store page of a tile from the Tile Flair website, the app would scan the HTML data of that page and find all relevant information about the desired tile. Extracting the name, height, width, price all directly from the website. Not only was this easier to implement than the method of reading from a CSV file, but this was also better from a maintenance perspective. With this method, information no longer needs to be updated to another file but is just taken directly from the website, implementation of this ensures that there is no need to update two databases to keep information consistent with what is advertised on the Tile Flair Website.

While this method is modular and adaptable to many different varieties of tile, it does require some "hard coding". In order to add a new tile into the implementation, a prefab has to be created in the Unity editor containing both the URL for the desired tile and an image of the tile texture. Unfortunately a lot of the images of tiles on the tile flair website are inconsistent in the way they were taken and would require someone to manually download a picture of the tile on a flat surface so there would be no feasible way to automate downloading a tile texture directly from the site. All other information can be downloaded loaded automatically, however.

The method used for building and viewing the walls of a given space went through many iterations. The first method was very basic and unoptimized, A grid of points would be generated upon first loading the app and with every mouse click, the app would determine which point the mouse click was closest to and use this as the start point for the desired wall. When the user would lift the mouse button, the app would again determine the closest point on the grid and use this as the end point of the wall. When a start and an end position for the wall had been determined, a series of

“Wall” objects would be instantiated along the full distance covered. These objects would then be grouped together as a single “Wall” and when in view mode, a raycast would be done to make the nearest series of wall objects invisible. This method produced results that were unoptimized at best and incredibly glitchy at worst. The objects would have small gaps between them, making the walls look undesirable and the method of making the walls invisible during view mode would cause them to flicker in and out of visibility. This method was done as an early implementation and it was clear from the first week that this would need to be changed.

This method was refactored so that only one object was created for each wall which would be scaled along the z axis of the object to fit the desired space. This had the side benefit of being able to adjust the wall while the user is making the selection, giving an instant feedback. While better, this still used the old method of raycasting to make walls invisible and so walls would still flicker in paint mode.

The current implementation adds one sided plane objects to the wall object. Being one sided means that they appear invisible if the camera is pointing the at the object in the wrong direction. Allowing any camera looking into a room can see through the closest walls. There was a bug that arose out of this unfortunately that mean that wall would have to be placed in a specific way to allow the user to see into a room due to the rotation of walls. This was fix however by a simple method of determining whether each wall was rotated toward the centre of the room or not and rotating them by 180 degrees along the y axis if they were facing the wrong way.

Since this app is a commercial product used to help sell tiles, it is vital that it be compatible with as wide a variety of platforms as possible as well as being easy to use. It is because of this throughout development, it was ensured that the app worked on both android and WebGL platforms, testing on iOS was considered however there was no access to iOS hardware during development so any attempts at building for these platforms would involve guesswork. Development on the Android and WebGL platforms was a surprisingly simple task and most controls on android were able to be translated from the Windows version essentially through simulated mouse clicks. The only major hurdle faced was in the WebGL build as security settings on most browsers only allow WebGL apps to access web pages from the same website they are hosted one, which meant any testing done would be blocked. This was able to be circumvented through downloading an extension for firefox to bypass this for testing. This isn’t perceived to be a problem for the overall implementation as in a real-world situation, this app would be hosted on the Tile Flair website, avoiding the security issues the WebGL build ran into.

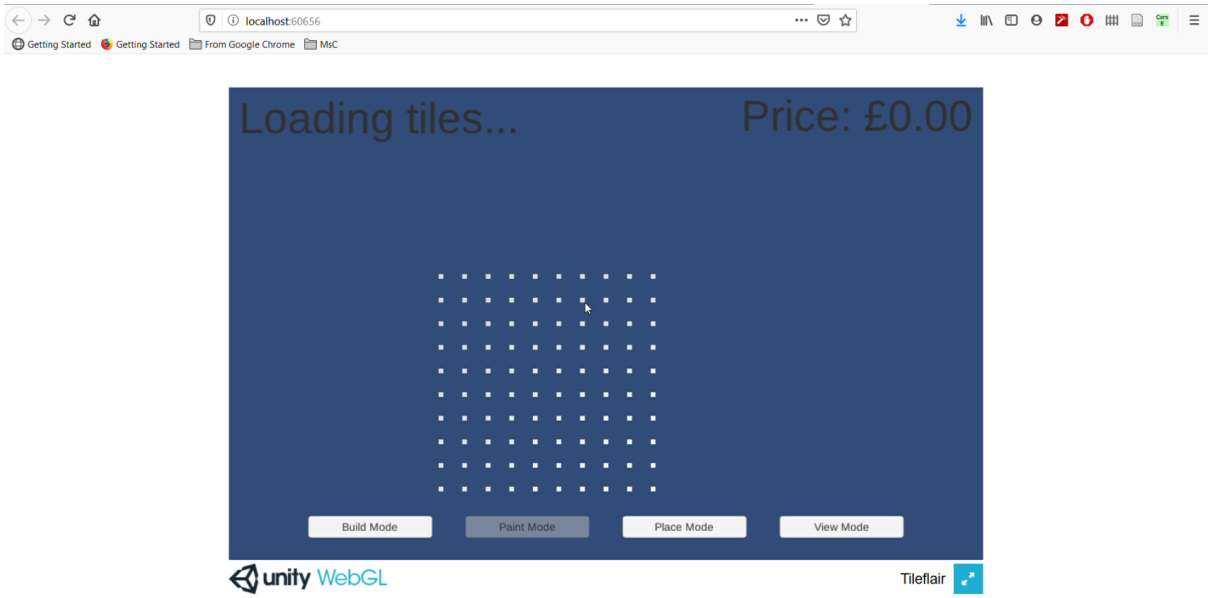

Figure 8 WebGL build of the app

User Testing

Due to the nationwide lockdown at the time of writing in response to the COVID-19 pandemic, full user testing of the app was not able to be conducted. This section will cover how testing would have been implemented had a nationwide lockdown not been in place and how success would have been measured through interpretation of the results of user testing.

Research was conducted to investigate the best way for testing the developed app against the objectives that were set out (Nielsen, 1994) details a series of different methods for usability of user interfaces. Making the user interface easy to use is a vital part in making the customer experience in using the app fun and easy. (Tan, Liu, & Bishu, 2009) Explores two methods for conducting evaluations on websites and found that “Heuristic” methods were beneficial for finding problems with websites. With this information, it was decided that a more user based approach would be appropriate since the purpose of the testing at this stage was not to find issues but to see if the app design was appropriate to help achieve the tasks set out.

User testing for this app would have involved giving users a tile visualisation tool and asking them to use it to imagine their desired room. Once finished, they would then be asked a series of questions based on their experience and if they managed to achieve the task. Their experience would also be recorded using desktop recording software with no audio or identifying features being recorded. The software given to each participant would be randomly selected from 3 possible options: the visualiser on Topps Tiles website (Topps Tiles, 2020), the visualiser from CTD Tiles (CTD Tiles, 2020) and the implementation created for this report. Each participant would only use one software to avoid bias or comparison with other experiences. After finishing the task, participants would be asked a series of questions about their experience with their method.

Ideally, the sample would be gathered from willing Tile Flair customers in store. This would involve spending the day in a store and approaching customers or asking staff to refer customers to the testing of the app. Such a sample would already be the intended audience for the app and would most likely be predisposed to want to use some form of visualiser since they are looking to buy tiles already.

The short survey is made up of 9 questions that cover relevant topics about the software they have used and relating to the original objectives laid out for the project. Covering simple questions over how easy it was to use their given tool and also asking how accurately they were able to create their room. The intention to see if a balance has successfully been struck between ease of use of software and the ability for the user to create their own room.

Data Interpretation

With testing finished, the accumulated data would have been studied to determine whether objectives had been met according to the given specification and the objectives laid out from research and from the meeting with the client.

Survey results would most directly determine the success of the software. It would be expected that most participants would find the Topps Tiles tool easiest to use but hardest to imagine how their tiles would look in their room. Conversely, it would be expected that the CTD Tiles tool would be the hardest tool to use overall but, if successfully used, would be the best for imagining how tiles would look in their room. The objective of the implementation created for this report would be to achieve results that balance between the ease of use of the Topps Tiles tool and the power of the CTD Tiles tool. If the new implantation managed to achieve better results in ease of use and ability to accurately create a room, then the software would have exceeded expected results.

Recordings of each users experience of the software would also be used to determine how quickly they were able to use the software and to determine any consistent problems or errors with the software. The main interpretation of the video recordings would be simply to see how long it takes for each user to finish using their given tool. Again, it would be expected that the Topps Tiles tool would have the shortest time to being finished as the controls are limited and do not allow for a lot of options when looking at tiles. The CTD Tiles tool however, would be expected to take much longer before a user felt they were finished. This is because it provides a much larger toolset and options to users so a user could go into much more detail but may have difficulty using the tool. Much like the survey results, the desired outcome of the app made for this report would be to achieve a balance between the time taken for Topps Tiles and CTD Tiles. With that said, if it was found that users were able to be finished even quicker than with the Topps Tiles tool while still achieving a much more accurate result to their desired room, then the app would have exceeded expectations.

The video recordings would also be used to determine and bugs or specific flaws with the implementation. Examples of this could include if users got stuck on a particular section or if they get frustrated with a particular feature. If consistent across a large number of users, such recordings could help to discover and remove issues and bugs from the software.

Conclusion

Many tile companies have made attempts at creating tile visualisation tools but few truly allow users to create their own rooms and accurately visualise their room with the tiles they want. Most allow users to simply place an image of a tile over a blank wall in a computer generated still picture of a bathroom or kitchen. Some allow more control but are needlessly complex and hard to use. This report sought to create a solution that bridged this gap and allowed a user to have more control over how the visualisation looked without being too complicated to use.

Through research and meeting with the client, objectives were established and followed throughout the development of the software. With an ease of use being at the forefront of the design of the app. The end product is one that seems to fulfil the laid out objectives and could potentially be a

jumping off point for a much larger tool that could be integrated into the Tile Flair website. User testing would have to be done to know if the objectives of the app were actually achieved however. This would require a comparison between other tools available and evaluating its success against the ease of use and power of those tools.

Further Development

There are various ways in which this tool could be expanded upon for further use. The tool in its current state provides a decent “proof of concept” or jumping off point for expanded features. A better user interface and tutorialisation of that interface could be made to create a much easier experience for the user along with of course expanding the range of tiles available and objects to place within a scene.

For features, a full integration with the Tile Flair website could be done where any tiles that a customer has favourited or added to their basket, could be instantly available when they move to the visualiser. This would make the customer experience much easier and would have the added benefit of less work having to be done on the visualiser side of tile selection. Having such an implementation could avoid the need to have the full catalogue of tiles available within the visualiser since tiles could simply be selected from the website.

There is also the possibility of implementing a Virtual Reality solution. Simple experimentation has concluded that tile measurements are accurate to VR size units and translate to real life scale. Unity allows for a simple implementation of a VR camera which certainly worked in theory. This would of course require much more research to fully implement but the principal of VR would be relatively simple to implement which would only add to the customer experience.

The possibility of an Augmented Reality app is not out of the question though implementation would not allow for real time AR with current technology. (Luo, Huang, Szeliski, Matzen, & Kopf, 2020) shows a method of depth estimation from video that is calculated after a video is taken. Such technology could potentially be used to accurately detect walls and apply tiles to them while tracking light sources. However, this would require much more research into the software before making a conclusion on if this is a good route to take.

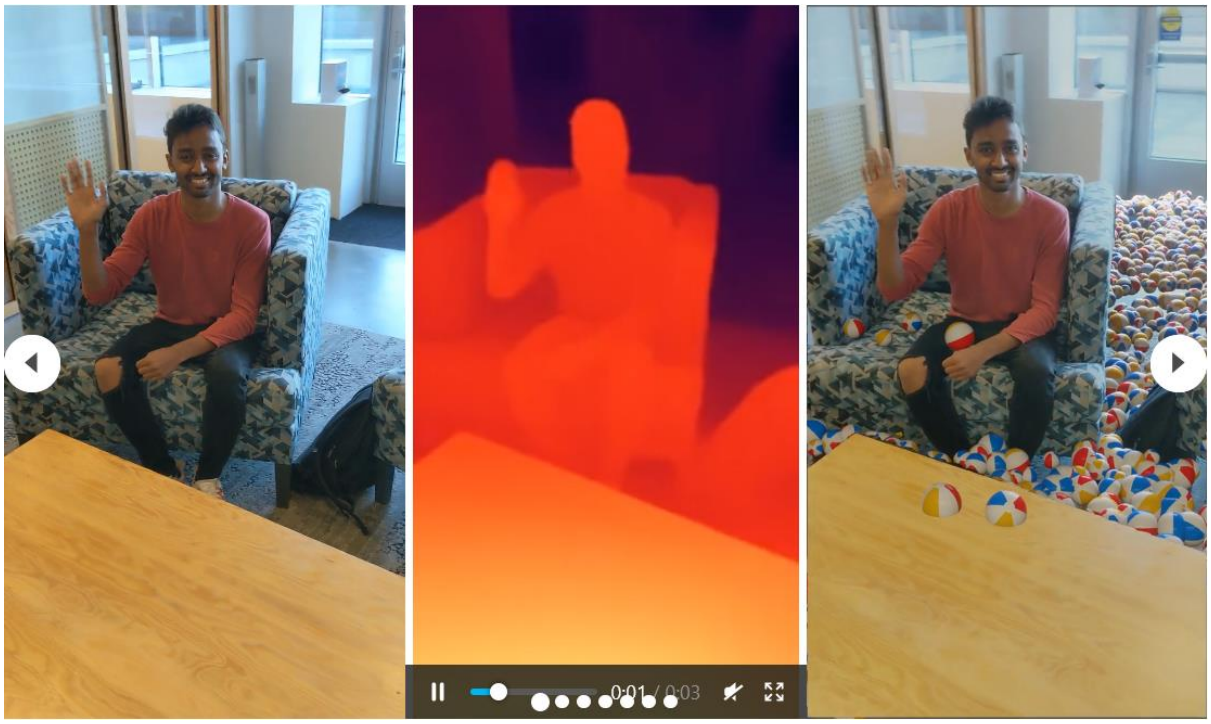

Figure 9 (Luo, Huang, Szeliski, Matzen, & Kopf, 2020) could potentially be used in a future AR implementation

This software does not necessarily have to remain exclusive to Tile Flair. There is potential that this tile visualisation software could be licenced to various other tile companies to help them accurately visualise their own tiles with an integration with their website. The current implementation is only tied to Tile Flair through the specific tiles used and the website information, everything else could quite simply be applied to another company if so needed.

This software works as a jumping off point for much further expansion and could be used in any number of ways to help visualise tiles for any site. While there are certainly improvements to be made before this could be used as a full commercial product, the software acts and a proof of concept that can be used as the basis for a full integration if required. The above improvements could all add to this create a tool that is far above anything else that is on the market right now.

Bibliography

CTD Tiles. (2020, May 5th). *CTD Tiles Gemini Blueprint Visualiser.* Retrieved from CTD Tiles: https://www.ctdtiles.co.uk/blueprinttool.aspx

Electronic Arts. (2014, September 4th). The Sims 4.

Luo, X., Huang, J.-B., Szeliski, R., Matzen, K., & Kopf, J. (2020). Consistent Video Depth Estimation. *ACM Transactions on Graphics*.

Nielsen, J. (1994). Usibility Inspection Methods. *Conference Companion on Human Factors in Computing Systems* (pp. 413-414). Boston, Massachusetts, USA: Association for Computing Machinery.

Tan, W.-S., Liu, D., & Bishu, R. (2009). Web evaluation: Heuristic evaluation vs. user testing. *International Journal of Industrial Ergonomics*, 621-627.

Tile Mountain. (2020, May 05). *Tile Mountain Tile Visualizer.* Retrieved from Tile Mountain: https://www.tilemountain.co.uk/visualiser/

Tile Warehouse. (2020, May 5th). *Tile Warehouse Visualiser.* Retrieved from Tile Warehouse: https://www.tilewarehouse.co.nz/thevisualiser

Topps Tiles. (2020, May 5th). *Topps Tiles Room Visualiser.* Retrieved from Topps Tiles: https://www.toppstiles.co.uk/visualiser/