**Analysis**

**Introduction**

The problem I am trying to solve with this problem is one with real world impact. My user is Andrew Church, the director of a turf installation business. To find out how much turf is required for a job my user must visit a site, get a measuring tape out and write down the amount, using these measurements then they must divide this number into rolls and order them. This problem could be solved by using drawings provided by architects, or by using another top down image (possibly taken from satellite imagery).

This would allow the user to quote the site quickly from home without needing to waste fuel and time by travelling to the site. Over a course of a year you can imagine that this problem is monumental and costly on a business with lots of competition. If you can put a quote in quicker and cheaper (since you save money on transportation) than the competition this could lead to the user receiving more business. Furthermore, being able to visualise how each strip of turf fits into an area would mean the user could adjust the positions and orientations of the turf to reduce wastage.

**Current Solution**

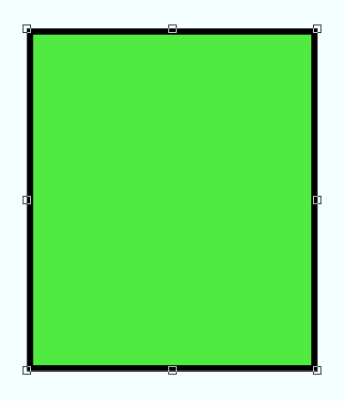


Fig 1: Shapes are fine to be scaled if they are oriented normally

This problem is currently being solved by Adobe Photoshop, which has some tools for measuring areas, creating shapes which can be imagined by the user as turf. Whilst this solution just about gets the job done, there are many flaws to using Photoshop for this problem.

Firstly, shapes must be created in terms of pixels and not real-world units like metres, this means there is a lot of guessing about if a shape fits the sizes that the turf comes in. If the user could specify that they wanted a 4m x 8m instead of 176 pixels x 352 pixels, furthermore, without manually measuring the shapes on screen using the ruler tool, you cannot see the dimensions of the shape in real units.

Secondly Photoshop has too many unnecessary features for the task, blending properties and colour gradients are just some of the endless features of photoshop. Whilst some may consider this a good thing since the program is versatile, I and my end user do not (see interview), having lots of unnecessary features makes it easy to get lost and makes it impossible for my end user to mark out an area of turf without accidentally blurring the whole project or turning it multi-coloured.

Lastly, Photoshop is a proprietary product and Adobe charge for the usage of this product, furthermore not being able to access the code means that it is impossible for my user to access new features. For a solution that doesn’t alleviate the problem to our specifications renting this service in simply unnececary. If the user didn’t have this extra monthly cost, they would make more profit and have a tailor made application which could be adapted privately if there was ever new features needed.

**Initial Questions**

To help me start with the project I first listed the foundation problems that I would have to solve.

* What do I want the program to do?

For this I decided to consult my user and conduct an interview, written in the **Interview** section of the Analysis.

* Which programming language is the best to use?

I am familiar with 3 languages, JavaScript, Delphi and Python. I feel like to step outside of these languages would be unnecessary and endanger the viability of the project.

To find which of the three is the best to use I created a basic prototype in Delphi, using VCL Forms. Whilst VCL forms had some good building blocks which I could drag in, like the TImage, TScrollBox and TButton, I found that there was seemingly a lot of road blocks to get to even a basic prototype, for instance Zooming was hard to get right as the program would often crash or bug out if I zoomed in too far. This ruled out Delphi as a candidate for the programming language, although deeper analysis of Dephi can be seen in **Research**.

I have become very familiar with JavaScript since I did my Extended Project in the language, through this I experienced lots of HTML Canvas, a tool which allows you to have dynamic scriptable rendering of shapes and images. I have discovered through stack overflow that this is a very feasible project in JavaScript.

* What should the GUI look like?

I have created a basic outline of what the overall user interface should look like. This UI features a bar for buttons, the main area where the image will be shown and an information tab, this obviously can change depending on what is needed later on in the project as I cannot accurately foresee the full required components for this UI this early on in the project.

Fig 3: First drawing of what a basic user interface could look like.

* Should the program be a web app or a compiled program?

From the research done on programming languages I have concluded that a web-based application would be best suited for this project. This also comes with more advantages, if I had this hosted then the user could use the app wherever they are no matter if they have their own computer with them, which could be the difference between getting a job and missing out on a job as the user did not quote in time.

**Interview**

* **What features are essential for this project?**

“This project needs to have a few key features for me, I need it to take an image in from my computer, put it on the screen and be able to navigate and zoom in, around the image. Then I should be able to specify the width of a piece of turf, have the turf appear on screen for me to move around the image for where the turf must be installed. Obviously, the turf has to stay in the correct location on the image when navigating around. After creating, I should be able to increase the length of the grass, so it fits the area. This should be repeatable until I have laid out all of the virtual grass on the site. I should be able to view the sizes of the pieces of turf I need. These are the basic components which would be nice to have, obviously there are some things which would be nice but are unessential.”

* **What features would be nice to have but are not essential?**

“Ooh, if it could calculate wastage, that would help to see which jobs are worth doing, which could save me money. Also converting turf sizes into cash equivalent would be nice, it would help me to prepare and give the customer a price there and then rather than me having to do it on paper. Saving could be nice, it would mean I could requote if the customer changes what they want, rather than me having to remake the whole thing.”

* **Do you care if the project is online based or a compiled application?**

“Not really, it shouldn’t matter too much as the vast majority of my quoting is done from my computer at home. Although if it was online, on the rare occasion that I wasn’t in the office, then I could still access the program, however this really isn’t a big deal for me.”

* **How is the problem being solved currently?**

“Currently I use photoshop, well, I actually can’t work photoshop because it’s so damn complicated with all its buttons and weird features, so I just get someone else to work on it usually. If it was simple enough for me to be able to use it that would be nice as it would speed me up, not having to ask others to do it for me.”

* **Is a user interface important to you?**

“Hmm, good question. I would say that there is a balance to this, I definitely don’t want a user interface which has hundreds of buttons and is made impossible to navigate. Similarly, too few buttons and features on screen would make it hard for me to do anything, especially if I have to memorize 200 shortcuts. So, the interface is important, but only in the sense that it must be usable, I don’t mind if it’s a white, basic interface without fancy transitions or unnecessary complexities.”

**Evaluation of Interview**

From this interview I have gathered lots of conclusions and it has helped me to see what the user wants this project to be. This has greatly informed my list of objectives and features, which has helped as it would be hard to meet my user’s requirements by thinking them up myself.

**Objectives**

Fundamental

1. It must be able to import an image, that the user uploads, onto the screen.
2. It must allow the user to rotate the image.
3. It must allow the user to navigate this image, zoom in and out and move around the image.
4. It must be able to create rectangles which display on screen and the user should have full control over the size of these rectangles.
5. It must be able to show the sizes of these rectangles with a real scale and not just pixels.
6. It must have a user interface, using buttons or other graphical elements to control the program.
7. It must support multiple rectangles at once.
8. It must be able to turn virtual units (i.e. Pixels) into real world units (i.e. Metres).
9. It must be usable, with no complicated features or application breaking bugs.

Optional

1. It could convert the sizes of grass into how much they would cost.
2. It could calculate wastage either in metres squared or in cash equivalent.
3. It could allow the user to save their session to allow the user to continue editing it later.
4. It could estimate the amount of labour necessary to complete the job i.e. 4 people needed for job.
5. It could calculate the cost of any labour.
6. It could come up with an overall cost of the job including transport fees, labour costs or any machinery.
7. It could allow the user to enter a margin of profit so that the user knows the price that they should give the customer.
8. It should be simple to use.

**Research**

Delphi

During the process of discovering if Delphi was the correct programming language, I created a prototype of what the solution could look like as well as implementing some basic features. I allowed uploads of images as well as zooming, this gave me a good idea of any struggles I would face along the way and also seeing if this language was a good match for my project.



Fig 4: An image of my prototype created in Delphi

It was immediately clear that programming was made a lot more complicated by Delphi, to put in features which would be fundamental to my project I would have to figure out ways to manipulate Delphi’s restrictions. A practical example of this occurred when enabling zooming, to zoom I would need to store the size as decimal numbers, but the zoom on screen only allowed integers, this meant I would have to round it, which eventually meant that small zooms were rounded to zero.

To conclude, this language would not be a good suit for my project, it seemed like Delphi was very good for simple applications since most things were done for you but would struggle at more complex ones, for instance, there was a module for almost every feature I would need, scroll box, importing images and more. Whilst to some this may seem very good as I would only have to focus on programming the back end, this removed a lot of my control of the program which often meant that I had to refer to the cryptic Delphi documentation.

To pursue a project in the application would be accepting that the end product would inevitably be a buggy mess, with half of the features that are required. This doesn’t fit with one of my end requirements for this project, which is to have a usable application. This means that Delphi is not a suitable match to program my project in.

JavaScript

Whilst experimenting, I decided to create a basic prototype to implement a few features which I would need. I split the program up into its basic components to see if it will be viable for this project to be completed in JS.

One of the first features I tested was if you could upload files into the browser. Since I had seen this before when browsing around the web, I knew that this was possible, but I wasn’t sure on the complexity of it. As it happens, this can be completed in one line of HTML and shows a convenient “Choose File” Button. This was a good indicator that HTML and JavaScript would be a good environment to enable me to complete the project.

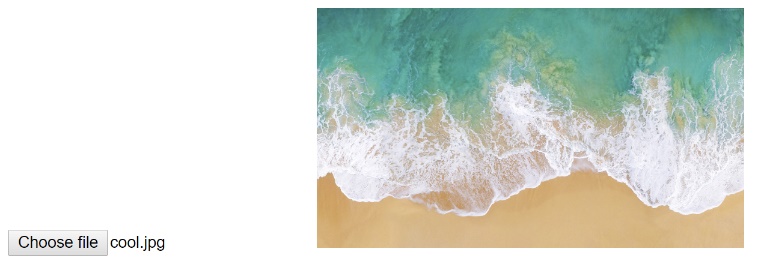


Fig 5: Third JavaScript prototype of features showing the Choose File and image preview combined together.

The next feature I decided to add was showing the image on the screen, I presumed that this would be trivial with the IMG tag, which I discovered through the W3Schools website (which was one of the most convenient tools throughout this research), however this feature was more complicated, as the IMG tag only allows you to show images which the website has loaded its self. However, there was a process to showing this image through a file reader. JavaScript has lots of active users, who help other programmers out on forums such as Stack Overflow, I specifically used this forum to find out about the file reader and implementing some of the tips they provided.

I combined my knowledge gained from both features and combined them into a single application where the user could upload a photo and have it show on screen (Fig 5), this was very helpful in showing how you could easily develop features separately and combine them into one application. This would help me as making features individually would allow me to ensure one feature works by prototyping it before combining it with the main project.

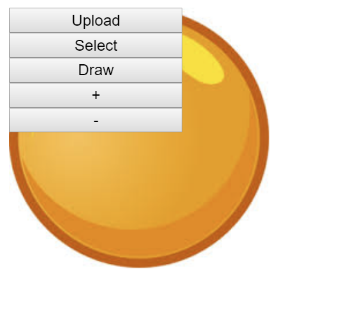
Next, I tried to enable Zooming, I discovered that with the IMG tag this would be very hard, if not impossible. Instead the common suggestion from friends was that I should use a canvas, which would allow me to easily control the zoom of an image since the drawImage function directly allowed for this. From this I began rebuilding my project without the IMG tag and with a canvas. Additionally, I introduced a layout which would be similar to what the end product would have (Fig 6).

Fig 6: Demo of image upload and canvas display with an example menu.

This final test required small visual improvements but was a good proof of concept. These tests with JavaScript cemented my faith in the language for use in this project. This language would allow me to implement all the features I require without restricting me or overcomplicating the project beyond understanding.

**Modelling**

During this project I will have to use complicated formulae to meet my objectives. Since my program is very visual, lots of these are related to mathematical formula.

* Collision detection

Since my program will enable the user to drag rectangles around, I will have to be able to detect when the mouse is over a rectangle. Discovering how I would do this I came across this forum post from 2015. Where it states that you can detect if mouse is over a rectangle by using the formula.

(mouseX > rectX && mouseX < rectX +rectWidth && mouseY > rectY && mouseY < rectY +rectHeight)

Figure 7: Detecting Hits on a rectangle (<https://forum.processing.org/two/discussion/12172/detect-mouse-withing-a-rectangle-shape>)

Furthermore, I will need to detect if the mouse is over a circle, if I decide to use a circular button. I did not need to research this as I already knew the formula. This can be detected by calculating the distance between the centre of the circle and the mouse, if it is less than the radius of the circle then it the mouse is within the circle (Figure 8).

Sqrt((circleY-mouseY)2+(circleX-mouseX)2)<circleRadius

Figure 8: Equation for detecting mouse over a circle