What is the goal of this prototype? What do you want to investigate by building it? What do you want to accomplish?

The broader goal of this prototype is to render the function $z=x^2+y^2$ in 3D, or a scaled version of it, so it fits on the display screen, and to time the average time it takes for each segment of the code to run. I chose this function because it's a generic polynomial function that pretty much everyone is familiar with but it's also not computationally that easy.

Of course that's a rather broad goal with plenty of possible solutions. I could even just manually calculate each coordinate myself and then paste them into SFML. So with that goal, I need to also define how good a solution is:

1. Minimize total runtime
2. The user must be able to control the position of the shapes so that the polytope may be translated onto any congruent polytope in R^3

I get into this in more detail in the last section.

What do you intend this prototype to look like? Is it a piece of code? Or is it a user interface mock-up? Or is it a test of a feature?

It's probably going to be a visual studio project that runs on 64 bit SFML (SFML 2.5.1, 64 bits for windows 10). It's going to be functionally equivalent to the final product, but slower.

What do you intend this prototype be able to do?

It's going to render each of the positive standard axes in R^3 as well as every user specified object. It's likely going to be a 3D graphing calculator that just renders 3D objects in 2D equipt with a basic shader. What it can't do is deal with variable material types, such as transparency and mirrors.

Are there costs or time limitations on this prototype?

My computer is relatively slow, and I don't have a lot of RAM allocated to this project, so the limitations will be resolution: high resolution will mean a slow speed. Also the shader I have in mind isn't perfect, and cannot handle specific combinations of shapes, it's designed like that to be faster but also less accurate, so accuracy will be another limitation.

As part of the whole project, what might impact how the features of this prototype will function?

The main impact is speed, if it's too slow, then there's a good chance that I might have to go back to the drawing board and figure out new solutions since my computer would simply just be too slow to render things properly.

As part of the whole project, what might the features of this prototype rely on in order to function?

The basic structure of variables and file structure. The project was designed around 4 basic functions that are integral to the project. I assumed that those four components would be the slowest parts of the project, but if, in fact, the rest of my project (handling input, sorting information, etc) is actually slower, then I need a new prototype.

If the prototype does not work properly, how could that impact the project?

Pretty much just start over. There are two ways the prototype can fail, one where the system is too slow, one where my logic is just wrong. In either case, I need to go back to the drawing board, come up with replacement components, and try again with a new prototype.

What circumstances or scope might change that could require a change in your prototype?

If I decide to allow for more types of input, such as implicit functions, complex valued functions, differential equations, and others that are difficult to define without a CAS.

What will the prototype do when integrated into the whole project? How will the prototype fit into the whole project?

Although it's functionally equivalent, every important part of this prototype will hopefully be replaced to make it faster. So for now, it'll provide the basic structure, variable names, and keyboard/mouse input handling needed for the final project.

Where or when will the users encounter the features of this prototype when the solution is complete?

Likely everywhere, the plan for right now is to have the prototype provide the same services, just slower.

Use Cases:

I don’t know if this pertains to me, because the prototype structure is going to be ubiquitous in the final project.

What are other features that you want to include in your prototype that you have not discussed already?

Object oriented programming, but C++ and visual studios has a habit of making objects a constant problem every time I try anything with an object, so for now it’s just arrays and vectors.

What are some features that would add to the functionality or user experience of your prototype, but would not change its ability to achieve your prototype scope?

Any sort of visual interface, something that makes the input fields look nicer and more organized. The default VS command line is awful for the user experience. It wasn’t the focus of my project to make this look nice, but I tried formatting some of the test cases earlier today and it took me hours just to define a few edges. If I ever get the chance to, I am definitely making a proper UI.

Also, intuitive controls would be nice, trying to control the system right now is like trying to play a video game where WASD was set to 1234. I simply chose the most mathematically sound set of controls, rather than spherical coordinate rotations, which is the most intuitive set of controls for the human mind. *

*rewrote this after starting the project

Requirements:

- Name
    - Point
- Description of critical feature
    - It draws a Point
- Target
    - Define and color in a point in 3D
    - Rotate and translate the point*
    - Display the point in 2D so that it aligns with the laws of physics
- Checkbox

- Name
    - Line
- Description of critical feature
    - It draws a line
- Target
    - Define and color in a line in 3D
    - Rotate and translate the line*
    - Display the line in 2D so that it aligns with the laws of physics
- Checkbox

- Name
    - Surface
- Description of critical feature
    - It draws a triangular surface, defined by 3 points
- Target
    - Define and color in a surface in 3D
        - Shade it accordingly
    - Rotate and translate the surface*
    - Display the surface in 2D so that it aligns with the laws of physics
- Checkbox

- Name
    - Function
- Description of critical feature
    - It draws a predefined function
- Target
    - Define and color a function in 3D by cutting it up into Surfaces
        - Shade it accordingly
    - Rotate and translate the surface*

        - Display the function in 2D so that it aligns with the laws of physics
- Checkbox

*The user must be able to control the position of the shapes so that the polytope may be translated onto any congruent polytope in R^3