Project Charter

CS 4110
Benjamin Gillott (bg357)

Logistics plan:

I plan to use git for source control and backup management. I do not have a partner for this project. This will be written in OCaml.

Proposal for the system:

Summarize the system you intend to build. Tell us what will be the most important functionality of your system. This summary should be about one page long. Provide:

Key idea:
This will be a system to represent vector calculations for computer graphics.

Key features:

- Represent vectors in 3D space (their basis, value, and type)
- Represent certain functions on vectors (such as cross product or transformation matrices)
- Read in a set of vector definitions
- Read in a set of vector calculations using predetermined functions
- Output if those calculations are valid / correct
- Stretch goal: output the result of the calculations

Narrative description:
Within computer graphics there are a number of subtle and difficult to solve bugs when doing vector calculations. An example of this is transforming a vector from one reference to another, such as from “Model” to “View” for getting representations of objects in 3D space. *

My approach to this is twofold:
Vectors - I will represent N dimensional vectors as per usual, but will assign “Tags” to them. This can include what this vector is in reference to (Model, View, an Object, World), and possibly other information such as if it is homogeneous or not, or what coordinate system it is in. The most basic goal will simply be for what it is in relation to though.

Functions - Functions will encompass anything which can take in one or more tagged vectors and output another vector or value. Some examples of this are transformation matrices between references, or the cross product formula.

A vector output by any function must have the correct tags assigned to it. For example a vector which is put into the “Model-View Matrix” function must have a Model reference tag, and the output must have the View reference tag.

This restriction could either be done by hand and then checked by the type system, or it could be automatically calculated by the system. I am unsure which is more feasible yet, and will follow whichever is doable in the time frame, or support both if time allows.

*Note: this idea is heavily inspired by the GATOR research language from Cornell CAPRA, but approaches the issue in a different way. I do not plan to reuse GATOR code or structure, and will approach this idea from scratch.

Roadmap:

Alpha:

- Setup a basic lexer and parser
- Design a basic “tag” system and programming language logic behind it
- Read in vectors and associated tags, represent in a defined structure

Beta:

- Choose a basic function and define on vectors with tags
- Check correctness on a single function with input and output
- Restructure the tag structure if necessary

Final:

- Extend the function support to a larger number of functions
- Read an extended program of vector calculations and determine correctness
- Stretch goal: automatically assign correct tags and values when not specified. Throw errors on expected vs actual tag and value mismatch.

Preliminary design sketch:

What are the important systems that need to be formalized or modules that will be implemented? What is the purpose of each?

- Vector definition and storage
- Function definitions

What data will your system maintain? What formats will be used for storage or communication? What data structures do you expect to use as part of your implementation?

Vector data and their associated tags. I will likely use a form of enum or list of the form (Reference, homogeneous,...etc), and store the possible references and other options in a list elsewhere.

What third-party libraries (if any) will you use?

I will likely find and use a library to help lexing and parsing. I will likely be using OCaml.

How will you test your system throughout development? What kinds of unit tests will you write?
I will write unit tests using graphics logic on expected values and types. I will look at verified graphics shader programs and take the types and values from those to use as expected results.

How will you, as a team, commit to following your testing plan and holding each other accountable for writing correct code?
I will refer back to this document and update it while following the original intent as best as possible.