Set - Round One Spec.

Detail exactly how your program will meet requirements and what it should do in every possible situation.

We will provide an interface for constraint-based modeling, allowing flexibility and extensibility beyond current 3D-modeling solutions.

We will meet this requirement by implementing constraint language both “under the hood” and for the user to interact with. This language consists of formal definitions of functions and parameters and applications of these functions to objects. These define how primitive units (in this case, points and lines) should be rendered to the scene.

On a more concrete level, users interact with a 3D space in a variety of ways. Objects on the stage can be manipulated through point-and-click interaction with objects, as well as traditional view transformations such as camera rotation, panning, and orbiting.

Selection

Selection is point-based; which is to say, users click on points in space, by holding down a control key, users may select clouds of points. clouds may be grouped for easier selection later.

Constraint

Constraints are modeled as functions from objects (objects being constrained or unconstrained shapes, or other functions) into objects. All interaction in **Set** is application of functions. These functions are represented as icons, which can be clicked on to be loaded into scope, and then applied through further clicking. A typing context prevents the user from applying functions nonsensically, through haphazard clicking. A toy example is sketched below:

What it should do in various situations:

-Basic interface acts similarly to popular 3D modelers in that users can input data to the stage using the mouse to draw and then transform their shapes using a library of constraints.

When user clicks in stage area, a point is rendered at the point of the click in world-space.

User drags the click--a line is generated starting from the incident click point and ending where the mouse is released.

When user clicks on a library constraint button, program looks to see if any primitive units are selected in the stage and if so, applies the constraints to them. If constraints are not applicable or are nonsensical, errors are thrown.

User can select and edit the dimensions and positions of both primitive units and full shapes, where a full shapes is defined by all connected primitive units. User selects one of two selector tools to select the type they choose.

If a user attempts to draw or move a shape on top of a pre-existing shape, the user will be asked whether to compound or subtract the new shape from the pre-existing one, or whether to ignore the move. Additionally, it may be easier to have tools that when selected consistently add or subtract new shapes if it is possible to do so.

Tools will be displayed on a standard toolbar interface. Different perspective views of the modeling area will be available. Saving and opening files will also be available.