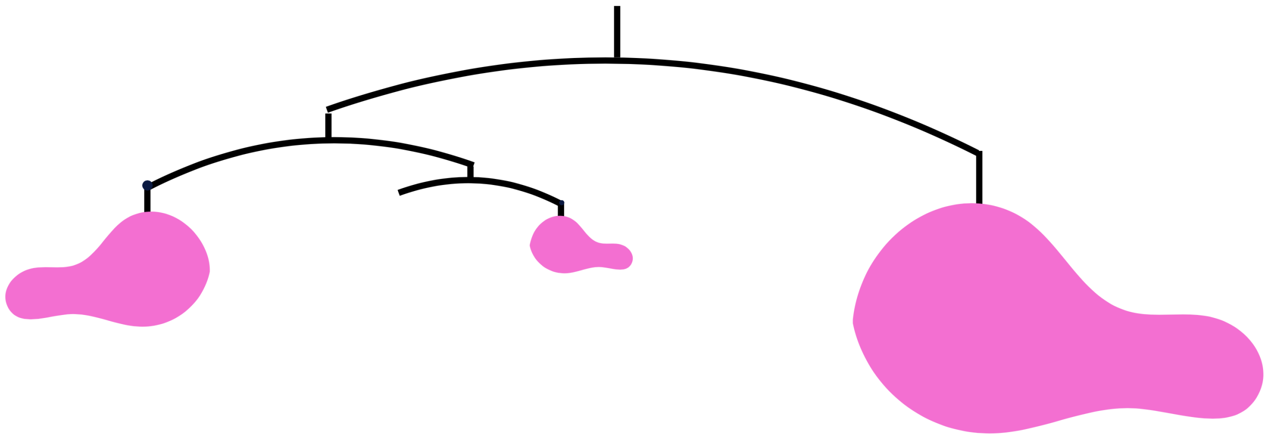
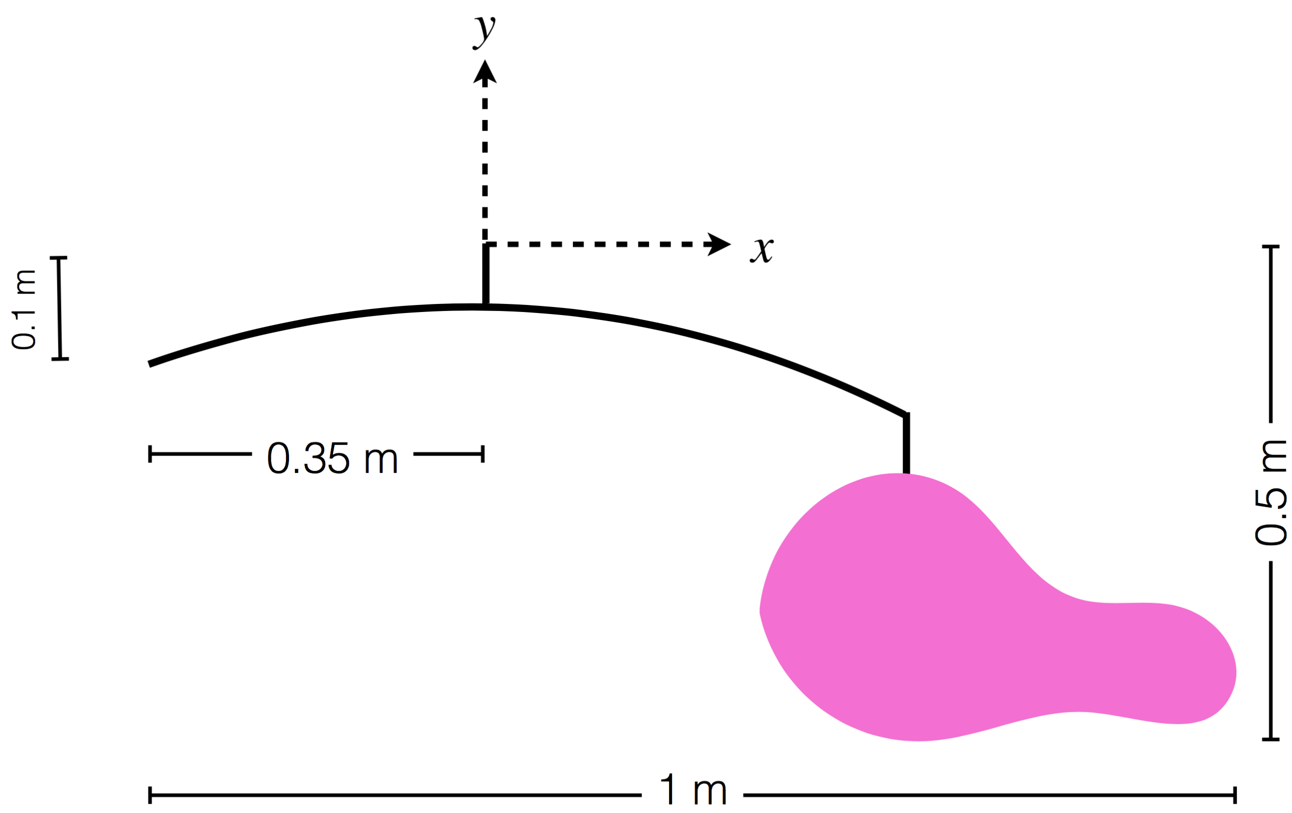
Spinners Spinny Mobile Sculpture

As a member of the software team at Spinners, Inc., a producer of high-speed rotating (and potentially body harming) decorative mobile sculpture. The design department has provided you with their latest design, pictured below:



The mobile is composed of three pieces, and for their website, they want an animated version of their model. The CAD (computer-aided design) department provides you with the following geometric model which is drawn using a function named `drawBlob()`, which renders one of the pieces sized as noted in the following technical specification diagram:



As a top-notch programmer (and someone who can name functions better than `drawBlob()`… seriously), you decide to use your handy `MatrixStack` class to aid in managing the transformations to render the animated model. As you can see, the mobile is composed of three parts that are differently sized versions of the mobile component. The largest component of the mobile is drawn at its native size (i.e., as shown in the above diagram). The middle-sized component is uniformly 40% the size of the largest component, and the smallest is 20% the size of the largest component.

While the production version of the website is supposed to have an animated version of the mobile with the elements rotating about the mounting point (i.e., the point at the origin in the specification mode), for the prototype, it's sufficient to use fixed angles for the rotations (i.e., angles that would match the parts orientation in the design picture above).

Using our handy planning document (the pages that following in this document), please:

- detail the sequence of matrix stack commands (e.g., `load()`, `push()`, `translate()`, etc.) including their parameter values
- include considerations for working with the viewing transform (just represent this as **V**, and don't worry about how it's generated for this situation)
- indicate when the model should be rendered, by calling `drawBlob()`
- and indicate the state of the matrix stack at each step, using our common notation of **R** for a rotation, **S** for a scaling operation, **T** for a translation, and **V** for the viewing transform

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State

Javascript functions	Matrix Stack State