(0,0) – (3,0); (2,0) arc (0:19:2cm); at (9.5:2.2) α; The red arc AB is a y-scaled copy of the blue arc UV.
Given the positions A, B, and M on the ideal unit sphere, computed as outlined below, here are the steps to draw line segment AB:

1. If A and M are not nearly antipodal (i.e. the angle between A and $-M$ is bigger than $SETTINGS.nearlyAntipodalAngle$) then compute the cross product $\mathbf{N} = [N_x, N_y, N_z]$ of the vectors A and M to determine the normal vector to the plane containing the red circle (i.e. the plane containing A, M, B, and O). If A and M are nearly antipodal let $N = A \times B$.

2. The major axis of the ellipse is the intersection of the plane containing the red circle and the XY plane. This means that in an orthogonal projection of the red circle plane onto the XY plane the major axis of ellipse is in the direction $\mathbf{PQ} = [-N_y, N_x, 0]$

Lines and line segments are managed as a `Two.Group` of `Two.Path`; one path for the foreground semicircle and another path for the background semicircle. The two semicircles are separated by the ellipse major axis. This approach allows us to apply different visual styles to each semicircle. In order to benefit from this organization, all drawing details must be expressible in terms of the ellipse major and minor axes.

The red arc AB is rendered by applying y-scale to the blue arc UV. The y-scaling essentially projects U and V to A and B respectively, using a projector line parallel to the minor axis (hence perpendicular to the major axis).

Given the two mouse click positions A and B, we can determine the locus of U and V on the boundary circle using the following steps:

1. Compute the normal vector of the plane containing the red circle, which is the cross product $\mathbf{N} = [N_x, N_y, N_z]$ of the normal vectors at A and B.

2. Since the ellipse major is on the XY plane and is also $\perp$ to $\mathbf{N}$, we can fix the major axis direction $\vec{\mathbf{PQ}} = [-N_y, N_x, 0]$

3. Determine the angle α between PQ and the X-axis

4. Rotate the entire diagram by $-\alpha$, so now:

   - The major axis PQ aligns with the X-axis
   - All the projector lines UA and VB are parallel to the Y-axis

   After this rotation by $-\alpha$ the following must be true: $a_x = r_x = u_x$ and $b_x = s_x = v_x$ and it is possible to determine the x-coordinates of U and V from the x-coordinates of the (rotated) mouse click positions.

5. Since both U and V are on the unit circle, we can calculate the following quantities $u_y = \sqrt{(1 - u_x^2)}$ and $v_y = \sqrt{(1 - v_x^2)}$

6. Determine the `startAngle` ($\angle\ QOV$) and `endAngle` ($\angle\ QOU$) and use these two values to determine which arc sections that must be rendered in foreground and background.

This is going to be unstable if A and B are nearly antipodal and doesn't work if A and B are exactly antipodal. This means that we need introduce a "midpoint" of A and B, called M, that is dynamically updated as the user mouse downs (at A) and drags. The location of M **must be** constrained as the user mouse downs and drags. The location of the mouse moved (and potential mouse release location - i.e. B) event is **not** the location of B.

Let A be the location in the unit sphere of the mouse down event. Initially, $M = A$. Now suppose that the user drags so that a mouse move event happens at location C on the unit sphere. Let:

- B be the displayed endpoint of the drawn line segment (could be temporary or final if C is also a mouse release point),
- $M_{old} = M$,
- $U = \frac{1}{2}(A + C))$ and $T = \frac{U}{|U|}$ (T is the unit normalization of the midpoint of the line segment AC that goes through the sphere), and
- $\alpha = SETTINGS.maxMidpointMovementAngle$.

To compute B and the new M use the following:

- If the angle between T and M_{old} is less than α then $B = C$ and $M = T$.
- If not then M is the point on the circle of radius α about M_{old} that is closest to T and B is the point so that if you normalize $\frac{1}{2}(A + B)$ you get the point M.
  To compute M:

  1. Let $F = M_{old} \times T$ and $G = F \times M_{old}$. (Notice that F, G, and M_{old} form a unit orthogonal frame. To avoid numerical instability F and G might need to be normalized.)
  2. Then $M = \cos(\alpha)M_{old} + \sin(\alpha)G$

  To compute B:

  1. Let $H = A \times M$ and $J = H \times A$. (Again notice that H, J, and A form a unit orthogonal frame. To avoid numerical instability H and J might need to be normalized.)
  2. Let θ be the angle between A and M, then $B = \cos(2\theta)A + \sin(2\theta)J$. (Again B should be normalized.)

Each semicircle is rendered in CCW order which implies that startAngle $<$ endAngle.

`startAngle` (S)	`endAngle` (E)	Background	Foreground
$S > 0$	$E > 0$	–	$[S, E]$
$S > 0$	$E < 0$	impossible	
$S < 0$	$E > 0$	$[S, 0]$	$[0, E]$
$S < 0$	$E < 0$	$[S, E]$	–