Hilbert Interstellar Algorithm
For Hilbert Metric Space Visualization (Implementation Details)

1. Press Sprite Mode Button

- `storeOriginalOriginalGeometry()`
  - Computes the *John Ellipsoid* of the *original* polygon configuration. In code:

    ```
    storeOriginalOriginalGeometry()
      -> computeJohnEllipsoid(originalPolygonVertices)
    ```

  - Stores the resulting center $\mathbf{c}$ and matrix A.
- `updateSpriteToCentroid()`
  - Moves the sprite to the centroid of the original polygon (i.e. sprite position := $\mathbf{c}$).

2. pressKey{`left`, `right`, `up`, `down`}

- Pressing one of these moves the sprite by some $(\Delta x, \Delta y)$.
- Let $(dx,\, dy)$ represent this immediate displacement.
- Then we *accumulate* this in a velocity or displacement vector:

$$(v_x,\ v_y)\ \leftarrow\ (v_x,\ v_y)\ +\ (dx,\ dy).$$

3. `StoreOriginalGeometry()`

- Normalizes the original polygon vertices *and* any relevant centers (e.g. asteroids, sites like HB, TB, RFB, etc.).
- By "normalize," we do:
  1. *Shift* all points so that the polygon's centroid becomes the origin. If $\mathbf{c} = (c_x, c_y)$ is the centroid, then

$$\text{normVertex}\ =\ \big(\text{origVertex}_x - c_x,\ \text{origVertex}_y - c_y\big).$$

  2. *Scale* the translated vertices

$$\text{scaleFactor}\ =\ \frac{2}{\max(\text{width, height})}.$$

     Then

$$\text{normVertex}\ \leftarrow\ \text{normVertex} \times \text{scaleFactor}.$$

     (This helps avoid large denominators in the projection formula.)

4. Project Points (*velocity vectors*)

- *Scale* the velocity vector by a constant $\alpha = 0.001$.
- `projectPoint()`:
  - Applies the projection $\left(\frac{p}{1+\langle p,v\rangle}\right)$ to *all points of interest*:

$$\text{POI} \;=\; \{\text{normalized vertices, asteroid centers, site centers, } \dots\}.$$

- *Un-normalize* these points of interest by the *inverse mapping*:

$$\mathbf{x}_{\text{unnorm}} \;=\; \frac{\mathbf{x}_{\text{projected}}}{\text{scaleFactor}} \;+\; \mathbf{c}_{\text{original}}.$$

5. Compute New John Ellipsoid

- Using the polygon's updated (projected + un-normalized) vertices, compute a *new* John Ellipsoid:

$$(\mathbf{x} - \mathbf{c}_{\text{new}})^{\top} A_{\text{new}}^{-1} (\mathbf{x} - \mathbf{c}_{\text{new}}) \;\leq\; 1.$$

- Store $\mathbf{c}_{\text{new}}$ and A_{new}.

6. Compute Final Mapped Points of Interest

- For each projected & un-normalized point of interest $\mathbf{x}$:
  1. *Map* the point to its new position by the transformation that sends the *new* John Ellipsoid to the *unit circle*:

  $$\text{LL}^* \;=\; \text{chol}\big(A_{\text{new}}\big), \quad \mathbf{y} \;=\; \text{L}\left(\mathbf{x} - \mathbf{c}_{\text{new}}\right).$$

  (Hence " maps New John Ellipsoid → Unit Circle.")
  2. Then map that $\mathbf{y}$ to the *original* John Ellipsoid for centering. we do:

  $$\text{LL}^* \;=\; \text{chol}\big(A_{\text{orig}}\big), \quad \mathbf{x}_{\text{final}} \;=\; L^{-1}\,\mathbf{y} \;+\; \mathbf{c}_{\text{original}}.$$

  (So "maps Unit Circle → Original John Ellipsoid.")