**Definitions**

To center the scene, we first compute minXO, maxXO, minYO, and maxYO. Note, a subscript “O” on a variable means that it is in object coordinates, a subscript “S” means that it is in screen coordinates).

We require a delta of 16S on all four sides of the window, such that the wall length texts can fit easily.

Since we havefor any point P, we can define the following requirements such that the min and max coordinates fit in the available window, and thus all objects fit in the available window:

Note that the Y coordinates are inverted for screen coordinates, because screen coordinates increase from top to bottom while object coordinates increase from bottom to top.

**Zoom level**

From the first inequality, we get, which we can fill in in the second inequality to get. Simplifying this inequality, we obtain thathas to hold.

From the third inequality, we get, which we can fill in in the fourth inequality to get . Simplifying this inequality, we obtain that has to hold.

For both directions to fit within the screen, we require that both of these inequalities hold. We define zoom as the largest integer for which both equalities hold:.

**Offset**

Now that we know the zoom level, we can fill in the original four inequalities to compute the offset in both directions. Specifically, we can rewrite the original four inequalities to the following:

Note that there are multiple values for both offsetX and offsetY such that the inequalities will hold. We would like to center the scene on the screen where possible. The two bounds represent the lower and higher screen edge, so we can take the midpoint of these two edges to center the scene.

Firstly for the x-direction:

We can simplify this formula to the following:

Secondly for the y-direction:

We can simplify this formula to the following: