Out[1]: (Show / Hide code)

Projections

CSE 4303 / CSE 5365 Computer Graphics

Brian A. Dalio, 2019 Spring

Objects exist in *world space*, a three-dimensional space with infinite extent. That is, the coordinates of points are $\in \mathfrak{R}$. To create a display or image that can be viewed, these coordinates must be converted into *screen space* (sometimes called *view space*). Screen space is limited by the size of the display or the image that is being created. The process of conversion is known as *projection*.

We will develop a simple version of projection to demonstrate the process. In particular,

- The plane of projection is the XY plane. That is, the process of projection converts a point (x, y, z) in world space to a point (x', y', z') *also in world space* where $z' = 0$. Thus below we speak of projecting a point (x, y, z) to a point (x', y').
- After a point is projected onto the XY plane, we transform it from world space to screen space. That is, we translate and scale its projected world space coordinates (x', y') so that they match the position and size of the defined screen space.

Since all of these operations can be performed by general matrix multiplications, it's possible to combine them into a single operation for efficiency's sake. We will keep them separate at first to make what is happening more obvious and easier to understand. Once this process is well understood, extending it to more efficient and more general cases is not difficult.

Parameters of Projection

The process of projection is controlled by a set of parameters.

① The first parameter of projection is how much of the world space is visible. We have expressed the visible portion of the world space in the model files with the values given on the `w` line.

```
w w_xmin w_ymin w_xmax w_ymax
```

This line says that we want a rectangular region stretching from (w_{xmin}, w_{ymin}) to (w_{xmax}, w_{ymax}) visible. For example, to display all objects with x and y coordiates not less than 0 and not more than 5, the `w` values would be `0.0 0.0 5.0 5.0`. Note that there is no explicit z range given. All objects with (x, y, z) coordinates within the rectangle defined by corners (x_{min}, y_{min}) and (x_{max}, y_{max}) are (potentially) visible no matter what their z coordinate might be.

② The second parameter of projection is the size of the region on which we are to draw (also known as the size of the *projection region*). This can be expressed by the $width$ and $height$ of the canvas when drawing on the screen or the size of the image that is being generated. Note that the world space coordinates are $\in \Re$ but the $width$ and $height$ of the canvas are integer numbers of pixels.

③ The third parameter of projection is the fraction of the draw region that is supposed to be used. These fractions are given on the `s` line in the model file.

```
s v_xmin v_ymin v_xmax v_ymax
```

The v_{xmin}, v_{ymin}, v_{xmax}, and v_{ymax} values are $\in [0, 1]$. For example, to reserve a 10% border around the drawing area, the `s` values would be `0.1 0.1 0.9 0.9`, meaning draw only from 10% to 90% along both the x and y dimensions of the drawing area.

With only these three sets of parameters, the simplest projection may easily be constructed.

Parallel Projection

The simplest projection is the *Parallel* projection. In the parallel projection, the z coordinate of the object is just ignored. That is, the projection of any point (x, y, z) is simply (x, y). The projection lines from points to the projection plane are parallel; they do *not* converge. A line in world space is a line on the drawing area *unless* it is parallel to the z axis in which case it appears a just a point. Assuming the lines are not parallel to the z axis, parallel lines in world space remain parallel when projected and their lengths are proportional. Angles in general are *not* preserved.

When converting the projected point to screen space, the (x, y) coordinates are simply transformed by translation and scaling so that the visible region of the world space fits the available drawing region of the canvas.

This projection and conversion was developed in detail in a previous handout. We summarize it here as,

$$
\begin{aligned}
(f_x, f_y) &= (-w_{xmin}, -w_{ymin}) \\
(g_x, g_y) &= (width \cdot v_{xmin}, height \cdot v_{ymin}) \\
\\
s_x &= \frac{width \cdot (v_{xmax} - v_{xmin})}{w_{xmax} - w_{xmin}} \\
s_y &= \frac{height \cdot (v_{ymax} - v_{ymin})}{w_{ymax} - w_{ymin}} \\
\\
a_x &= f_x s_x + g_x \\
a_y &= f_y s_y + g_y \\
\\
(x_{screen}, y_{screen}) &= (\, s_x x_{world} + a_x,\ \ s_y y_{world} + a_y \,)
\end{aligned}
$$

Once the `w`, `s`, $width$, and $height$ parameters are known, the s_x, s_y, a_x, and a_y values may be computed. Using these values, the projected coordinates (x_{screen}, y_{screen}) may be computed for any coordinates (x_{world}, y_{world}). In particular, when drawing an object, the s_x, s_y, a_x, and a_y values are computed *once* and then all of the points of the object are transformed.

The next time the object has to be drawn, the s_x, s_y, a_x, and a_y values need to be recomputed only if the `w`, `s`, $width$, and $height$ parameters have changed. Computing s_x, s_y, a_x, and a_y costs 6 multiplies, 2 divides, and 4 additions/subtractions, but having them means each point can be transformed for a cost of only 2 multiplies and 2 additions. This is definitely a good investment given that objects can have tens, hundreds, even thousands of points that need to be transformed.

Perspective Projection

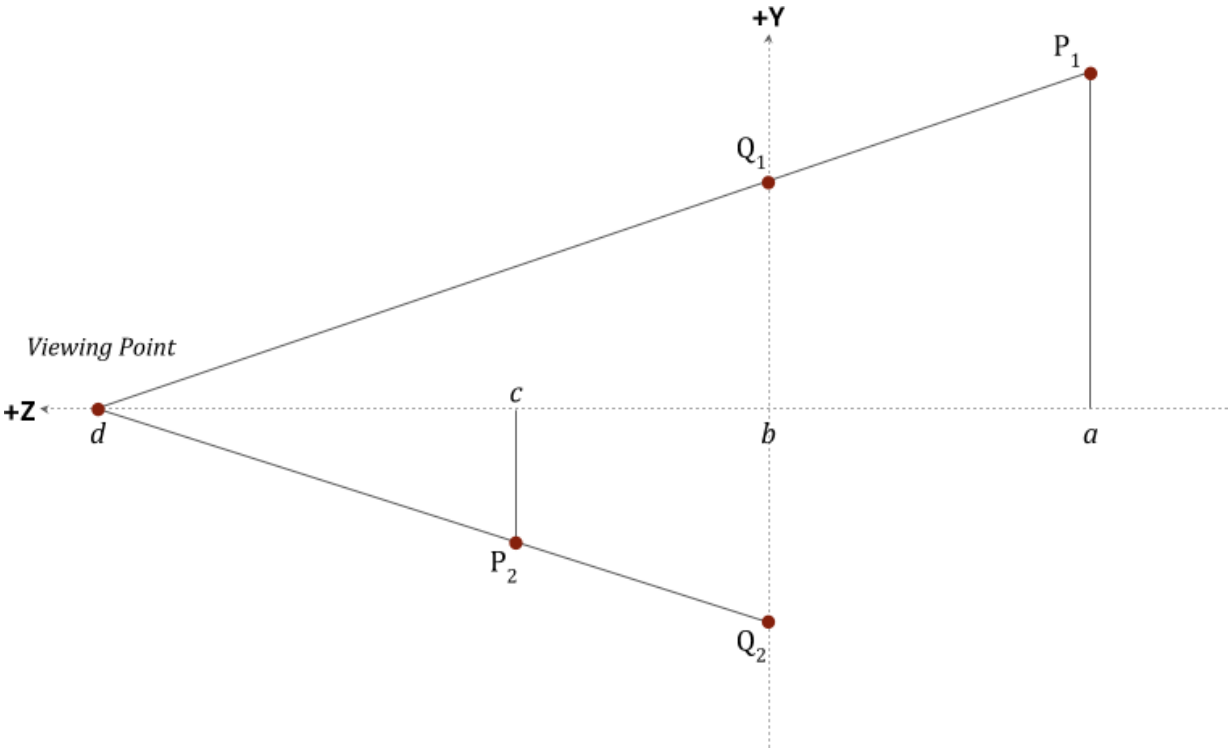
Parallel projection is useful in certain technical circumstances when it is required that parallel lines remain parallel, lengths remain proportional, and measurements are possible from the generated imagery, but it does not match human vision. For more realistic image creation, *Perspective* projection is used. In a perspective projection, we start with the same parameters that were used in the parallel perspective case, but there are also three viewing parameters: the *viewing point*, the *viewing direction*, and the *viewing orientation*.

To start, we take the simplest configuration of viewing parameters,

- The viewing point is somewhere on the z axis, having the coordinates $(0, 0, d)$. The position d is usually positive, but it doesn't have to be.
- The viewing direction is in the negative z direction.
- The viewing orientation is y up.

With these parameters, we can now derive a method for computing the projected coordinates (x^*, y^*, z^*) from the original coordinates (x, y, z). As in the parallel projection case, we project the objects onto the XY plane so the z^* coordinates of all projected points will be 0. We will therefore refer to the projected point's coordinates as (x^*, y^*).

In the following picture, we show a two-dimensional version of the problem. The positive z axis goes to the left, the positive y axis goes up. The viewing point is on the left, at $z = d$. The points P_1 and P_2 are to be projected onto the y axis so the problem is to determine the y coordinates of the points Q_1 and Q_2.



We can directly observe that the triangle formed by d, a, and P_1 is *similar* to the triangle formed by d, b, and Q_1. This means that that the ratio of the y coordinate of Q_1 (Q_{1y}) to the distance d will be the same as the ratio of the y coordinate of P_1 (P_{1y}) to the distance $d - P_{1z}$. Since we already know d, P_{1y}, and P_{1z}, we can compute Q_{1y} thus,

$$\frac{Q_{1y}}{d} = \frac{P_{1y}}{d - P_{1z}}$$

$$Q_{1y} = P_{1y}\frac{d}{d - P_{1z}}$$

$$Q_{1y} = P_{1y}\frac{1}{1 - \frac{P_{1z}}{d}}$$

Dividing both the numerator and denominator of that fraction by d puts the factor we multiply P_{1y} by in a more useful form. We'll see why just below.

The same similar-triangle reasoning holds for the triangle formed by d, c, and P_2 and the triangle formed by d, b, and Q_2. This means that that,

$$\frac{Q_{2y}}{d} = \frac{P_{2y}}{d - P_{2z}}$$

$$Q_{2y} = P_{2y}\frac{d}{d - P_{2z}}$$

$$Q_{2y} = P_{2y}\frac{1}{1 - \frac{P_{2z}}{d}}$$

Though we drew the diagram above with z going to the left, y going up, and x coming out of the page, we could equally well have drawn it with x going up and y going into the page. If we do that now, we can follow similar lines of reasoning to derive the relationship for the x coordinates of Q_1 and Q_2,

$$\frac{Q_{1x}}{d} = \frac{P_{1x}}{d - P_{1z}}$$

$$Q_{1x} = P_{1x}\frac{d}{d - P_{1z}}$$

$$Q_{1x} = P_{1x}\frac{1}{1 - \frac{P_{1z}}{d}}$$

and

$$\frac{Q_{2x}}{d} = \frac{P_{2x}}{d - P_{2z}}$$

$$Q_{2x} = P_{2x}\frac{d}{d - P_{2z}}$$

$$Q_{2x} = P_{2x}\frac{1}{1 - \frac{P_{2z}}{d}}$$

So for points P_1 and P_2, we can compute scaling factors

$$\frac{1}{1 - \frac{P_{1z}}{d}} \text{ and } \frac{1}{1 - \frac{P_{2z}}{d}}$$

that are used to obtain the projected (x^*, y^*) from the corresponding (x, y). Pretty simple, huh?

Matrix Form

It would be nice to have this projection in the form of a matrix similar to that used for translation, scaling, rotation, etc. This is easily possible if we express the point in homogeneous coordinates. We can use a matrix of this form,

Out[2]:

$$\begin{bmatrix} x^* \\ y^* \\ z^* \\ w^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ 0 \\ 1 - \frac{z}{d} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{x}{1-\frac{z}{d}} \\ \frac{y}{1-\frac{z}{d}} \\ 0 \\ 1 \end{bmatrix}$$

$$(x^*, y^*) = \left(\frac{x}{1 - \frac{z}{d}}, \frac{y}{1 - \frac{z}{d}}\right)$$

Note that the z^* coordinate will always be 0. This should have been expected as the projection is onto the XY plane, which has $z = 0$.

Perspective projection is *not* an affine transformation as can be seen from the transformation matrix having a non-zero value in the third column of its fourth row. We can therefore end up with a w^* value that is not one. This will happen whenever the point that is being projected has a $z \neq 0$. We therefore *rehomogenize* the result by dividing each component by w^*.

Some Observations

Two interesting observations may be made about limiting cases of perspective projection.

① *Viewing position* $d \to +\infty$. As the position of the viewing point d moves to infinity along the positive z axis, the value of $\frac{z}{d}$ gets smaller and smaller, that is,

$$\lim_{d \to \infty} \frac{z}{d} \to 0$$

$$\lim_{d \to \infty} 1 - \frac{z}{d} \to 1$$

$$\lim_{d \to \infty} \frac{1}{1 - \frac{z}{d}} \to 1$$

$$\lim_{d \to \infty} (x^*, y^*) \to (x, y)$$

In other words, as the viewing point moves to positive infinity along the z axis, the *perspective* projection becomes the *parallel* projection, projecting each point onto the XY plane by merely ignoring the point's original z coordinate.

Given this limiting case for *perspective* projection, we can instantly write the transformation matrix for the *parallel* projection. Since the perspective projection becomes the parallel projection as $d \to \infty$, the parallel projection transformation is,

$$\begin{bmatrix} x^* \\ y^* \\ z^* \\ w^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

$$(x^*, y^*) = (x, y)$$

② *Point's* $z \to -\infty$. As a point moves to infinity along the negative z axis (that is, gets farther and farther away from the viewing point and the projection plane), the value of $\frac{z}{d}$ becomes more and more negative, that is,

$$\lim_{z \to -\infty} \frac{z}{d} \to -\infty$$

$$\lim_{z \to -\infty} 1 - \frac{z}{d} \to \infty$$

$$\lim_{z \to -\infty} \frac{1}{1 - \frac{z}{d}} \to 0$$

$$\lim_{z \to -\infty} (x^*, y^*) \to (0, 0)$$

In other words, as the point being projected moves to negative infinity along the z axis, its projection becomes the origin. Therefore, the farther away an object is, the smaller it is when perspective projected. No matter how big an object is, eventually it vanishes in the distance. Given this characteristic, we can ignore objects that are "too far away" as they will not contribute any detail to the image being created. (Exactly what "too far away" means depends on the exact projection parameters that are used and the size of the object.)

Getting the Screen Coordinates

Once the point has been perspective projected, we have to transform it from world space coordinates to screen space coordinates. This can be done by using the same transformation from the parallel projection case. Therefore, the complete perspective projection from world space to screen space coordinates is of this form,

$$x_{screen} = s_x x^* + a_x \qquad = s_x \frac{x_{world}}{1 - \frac{z_{world}}{d}} + a_x$$

$$y_{screen} = s_y y^* + a_y \qquad = s_y \frac{y_{world}}{1 - \frac{z_{world}}{d}} + a_y$$

where the s_x, s_y, a_x, and a_y values are computed as they are for the parallel projection case.

Note

> Examination of these expressions shows that if a point has a $z = d$, a divide-by-zero exception will occur when trying to perspective project the point. Further, if $z > d$, the sign of the denominator will become negative and the projection will reflect about the x and y axes. Neither of these cases is reasonable, so when programming this, ensure that no points with $z >= d$ are projected. Such points should be rejected as not visible. When $z = d$, the point is in-line with the viewing point, which would require the view to see "sideways" to itself. When $z > d$, the point is behind the viewing point, which would require the view to see "behind" itself.

A Numerical Example

[*Do we need a specific numerical example here?*]

Other Examples

The following plots show the "Pillars Nine" data set drawn with a variety of projection parameters. They all have the world space and screen space definitions of,

```
w -8.400000  -8.400000  8.400000  8.400000
```

```
s  0.100000   0.100000  0.900000  0.900000
```

but vary in projection type and (in the case of perspective projection) the z position of the viewing point.

One may wonder why in the perspective projection cases even though the z position of the viewing point is getting farther and farther from the projection plane, the sizes of the tops of the pillars don't get any smaller than they are in the parallel projection case.

The explanation is that the world space definition was not changed for any of these projections. Since the same size region is being displayed in the same screen space, the tops of the pillars do not get any smaller. What is happening is that the field of view (FOV) angle is getting narrower and narrower. The result is a magnification that exactly balances for the increased distance of the viewing point.

FOV Angle

Calculating the FOV angle is fairly easy since we know the amount of world space that is being seen and the distance of the viewing point from the projection plane.

$$w_{diag} = \sqrt{(w_{xmax} - w_{xmin})^2 + (w_{ymax} - w_{ymin})^2}$$

$$FOV_{diag} = 2 \arctan \frac{w_{diag}}{2d}$$

where d is the z position of the viewing point. This FOV is then the field of view as measured *diagonally* from the lower-left to the upper-right corner of the world space visible region.

Focal Length

While the FOV angle is perfectly accurate, many persons find it easier to grasp the *focal length* of an equivalent physical lens instead. As 35mm lenses are commonly used, we can restate the FOV angle as the focal length of an equivalent 35mm full-frame camera lens instead,

$$\begin{aligned} ff_{diag} &= \sqrt{36^2 + 24^2} \\ &= \sqrt{1296 + 576} \\ &= \sqrt{1872} \\ &\approx 43.267 \end{aligned}$$

$$fl_{35mm} = \frac{ff_{diag}}{2 \tan \frac{FOV_{diag}}{2}}$$

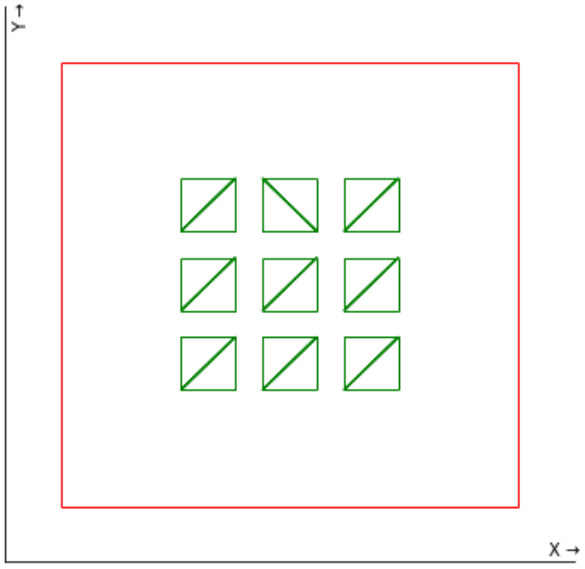
In this derivation, ff_{diag} is the length of the diagonal of a full frame of 35mm film. (A frame of 35mm film is 36mm wide by 24mm high. The measure 35mm comes from the height plus the space for the perforations.)

For reference, the viewing point z values, focal lengths, and FOV angles of the examples are given in the following table,

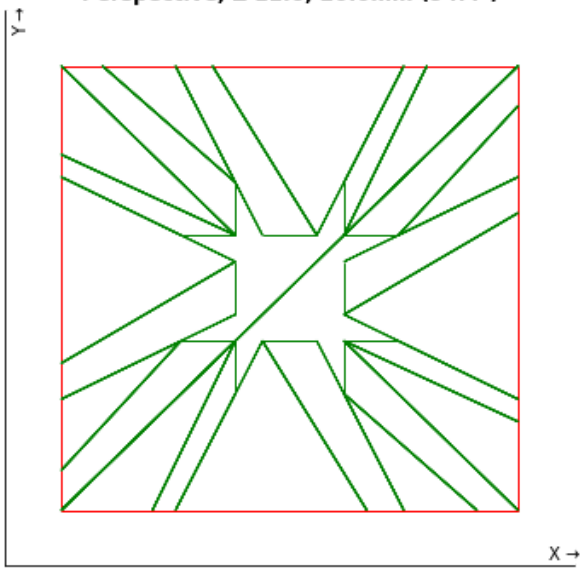
z value	focal length	FOV	z value	focal length	FOV

z value	focal length	FOV	z value	focal length	FOV
11	20.0mm	94.4˚	80	145.7mm	16.9˚
15	27.3mm	75.8˚	160	291.4mm	8.5˚
20	36.4mm	61.4˚	320	582.7mm	4.3˚
30	54.6mm	43.2˚	640	1,165.5mm	2.1˚
40	72.8mm	33.1˚	1,280	2,331.0mm	1.1˚

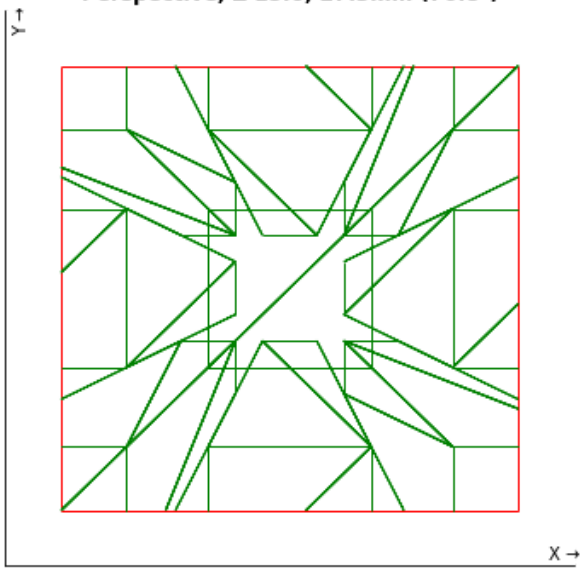
Parallel



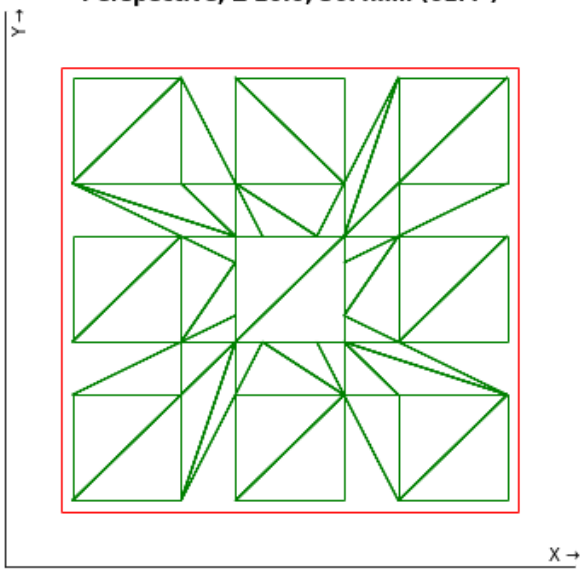
Perspective, Z 11.0, 20.0mm (94.4°)



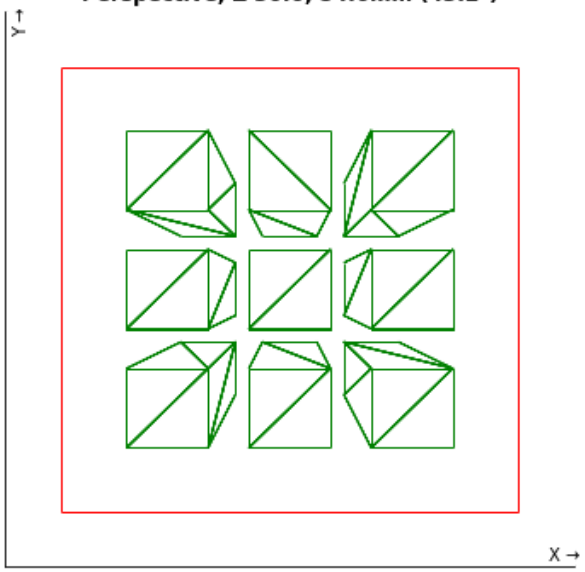
Perspective, Z 15.0, 27.3mm (76.8°)



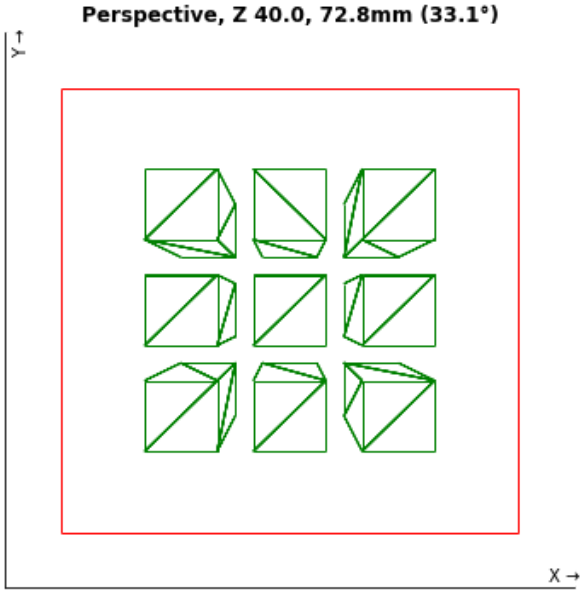
Perspective, Z 20.0, 36.4mm (61.4°)



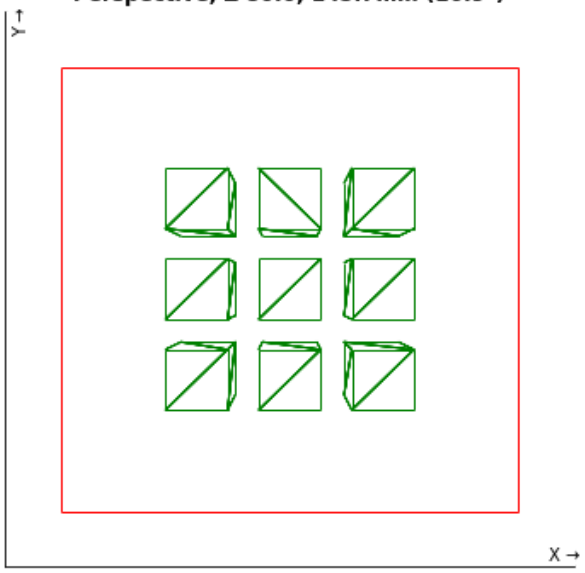
Perspective, Z 30.0, 54.6mm (43.2°)



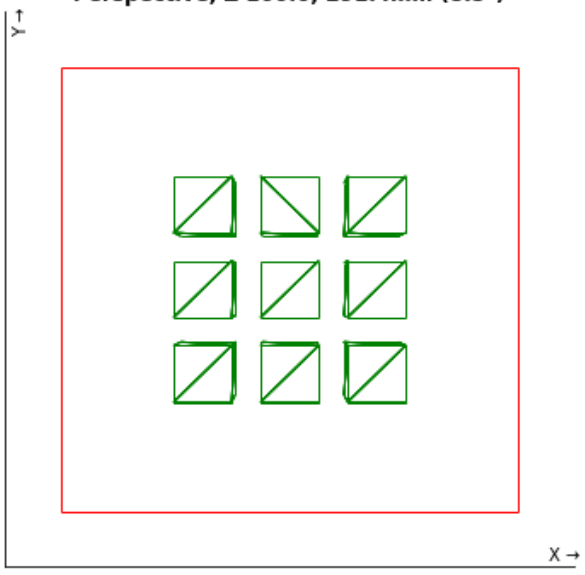
Perspective, Z 40.0, 72.8mm (33.1°)



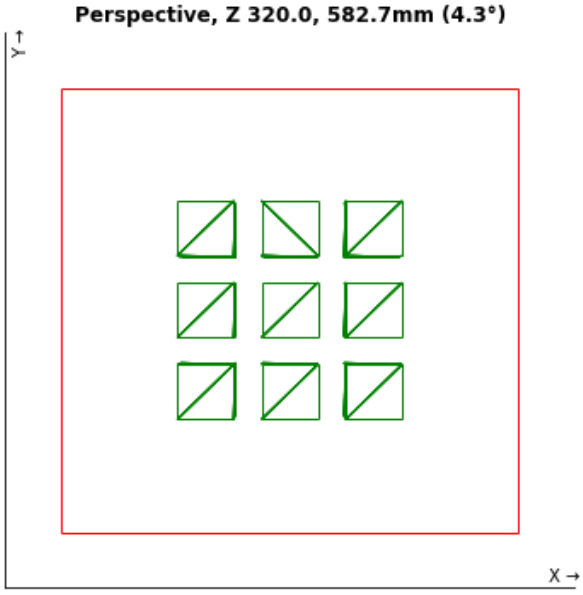
Perspective, Z 80.0, 145.7mm (16.9°)



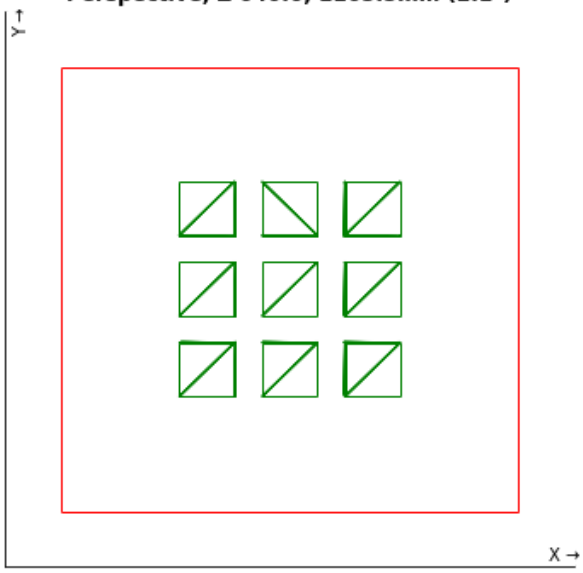
Perspective, Z 160.0, 291.4mm (8.5°)



Perspective, Z 320.0, 582.7mm (4.3°)



Perspective, Z 640.0, 1165.5mm (2.1°)



Perspective, Z 1280.0, 2331.0mm (1.1°)