From the Vertex Shader code, we have the `model` (transformation) uniform for movement.

```
...
uniform mat4 model;

void main() {
  gl_Position = model * vec4(aPos, 1.0f);

  ...
}
```

In the main program, we create representations of the speed and position of the texture. In my case, it looks like this:

```
...
glm::vec2 tex_pos(0.0f, 0.0f); // as for the texture rectangle location
glm::vec2 tex_speed(0.0001f, 0.0001f);
renderloop {
  ...
  tex_pos += tex_speed * delta_time;
  model = glm::translate(
        model, glm::vec3(dvd_texture_position.x, dvd_texture_position.y, 0.0f));
}
...
```

Mathematically, taking that these values are combined to fit the model structure used to update the position of the texture; we would have the following

tex_pos $= \begin{pmatrix} x & y \end{pmatrix}$
tex_speed $= \begin{pmatrix} s_x & s_y \end{pmatrix}$
and for some given scalar δ;

tex_pos $= \begin{pmatrix} x & y \end{pmatrix} + \begin{pmatrix} s_x & s_y \end{pmatrix} \cdot \delta$
tex_pos $= \begin{pmatrix} x & y \end{pmatrix} + \begin{pmatrix} \delta s_x & \delta s_y \end{pmatrix}$
tex_pos $= \begin{pmatrix} x+\delta s_x \\ y+\delta s_y \end{pmatrix}$

$$\text{model} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x+\delta s_x \\ y+\delta s_y \\ 1 \end{pmatrix} = \begin{pmatrix} x+\delta s_x+T_x \\ y+\delta s_y+T_y \\ 1+T_z \\ 1 \end{pmatrix}$$

This used back in the vertex shader would be;

$g_p = \begin{pmatrix} x+\delta s_x+T_x \\ y+\delta s_y+T_y \\ 1+T_z \\ 1 \end{pmatrix} \cdot \begin{pmatrix} P_x & P_y & P_z & 1 \end{pmatrix}$, where g_p is gl_Position.

Then this means our task is to create an equation that ensures the vector from each does not exceed the values of the coordinates $[(1,1),(1,-1),(-1,-1),(-1,1)]$ for the OpenGL XY plane.