3D Rendering in Postscript

Ben Newman

March 20, 2022

Contents

1 Overview

2 Isometric Projection

The first rendering method I implemented was Isometric Projection. This method is mainly used in technical drawings, as **PUT STUFF HERE** The method to obtain the proper projection coordinates with Isometric projection is relatively simple, requiring only two matrix multiplications (where one of the matrices is the 3d coordinates, one doesn't change, and the last is based off of the camera angle). The equation I used was:

$$a_x = B_{11}, a_y = B_{21} \tag{1}$$

Where: (2)

$$\alpha = \arcsin(\tan 30^\circ) \text{ (Rotation around the horizontal axis)} \tag{3}$$

$$\beta = 45^\circ \quad \text{(Rotation around the vertical axis)} \tag{4}$$

c is the 3d coordinates (5)

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & -\cos\alpha \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \tag{6}$$

This calculation is relatively light, and for most cases can be run in real time with a few objects. However, because of the way that the points are projected, objects don't scale based off of the camera – In fact, there is no defined camera location. This limitation prompted me to instead look to Perspective Projection.

3 Perspective Projection

4 Code

The following code is the main file of the 3d rendering (i.e. the ”library”):

```
% MODIFIABLE CONSTS
% whether or not to use back-face culling
/enable_back_face_culling true def
% whether to draw full tris or just wireframe versions
/render_method (wireframe) def
% the z scale is super messed up, so you can change this if you want but it will break
% this constant should scale z to the same scale as the toher two axes
/zMult 0.0016 def


% why is atan in postscript but not tan?
/tan {dup sin exch cos div} def

/atan2 {
        2 dict begin
        % atan2 implementation because postscript doesn't have one
        /y exch def
        /x exch def
        x 0 gt { y x atan }{
                y 0 gt y 0 eq or x 0 lt and { y x atan 180 add }{
                        x 0 lt y 0 lt and { y x atan 180 sub }{
                                x 0 eq y 0 gt and { 90 }{
                                        x 0 eq y 0 lt and { -90 } { (undefined atan2 result) error } ifelse
                                } ifelse
                        } ifelse
                } ifelse
        } ifelse
        end
} def

% arbitrary amount of digits
/pi 3.141592653 def

/identity {
        1 dict begin
        /n exch def
        /m [n n mul {0.0} repeat] def
        0 n 1 add n n mul 1 sub {
                m exch 1.0 put
        } for
        m
        end
} def

/vsub {
        2 dict begin
        /b exch def
        /a exch def

        /result a length array def
        0 1 a length 1 sub {
                /i exch def
                result i a i get b i get sub put
        } for
        result
        end
} def

/vadd {
        2 dict begin
        /b exch def
        /a exch def

        /result a length array def
```

```
        0 1 a length 1 sub {
                /i exch def
                result i a i get b i get add put
        } for
        result
        end
} def

/scalar {
        1 dict begin
        /n exch def
        /a exch def
        /result [a length {0.0} repeat] def
        0 1 a length 1 sub {
                /i exch def
                result i a i get n mul put
        } for
        result
        end
} def

/matrix_multiply {
        3 dict begin

        /m exch def
        /b exch def
        /a exch def

        /n b length m idiv def
        /l a length m idiv def

        /result [n l mul {0.0} repeat] def
        0 1 n l mul 1 sub {
                /i exch def
                /row a i m m n eq {idiv}{mod} ifelse l mul m getinterval def
                /col m array def
                0 1 m 1 sub {
                        /j exch def
                        col j b j n mul i n mod add get put
                } for
                /c 0.0 def
                0 1 m 1 sub {
                        /j exch def
                        /c c row j get col j get mul add def
                } for
                result i c put
        } for
        result
        end
} def

/iso_rot_matrix {
        1 dict begin
        /n exch def
        [1 0 0 0 angle2 tan arcsin cos angle2 tan arcsin sin 0 angle2 tan arcsin sin neg angle2 tan arcsin cos] [n cos 0 n sin n
        end
} def

/iso_project {
        1 dict begin
        /point exch def
        /newPoint angle iso_rot_matrix point 3 matrix_multiply def
        newPoint 0 get
        newPoint 1 get
        end
} def

/cPos [0 0 .5] def
```

```
% if any of cAngle are 0, things go very wrong
/cAngle [0.001 0.001 135.001] def
/displaySurface [0.001 0.001 .5] def

/perspective_rot_matrix {
        [1 0 0 0 cAngle 0 get cos cAngle 0 get sin 0 cAngle 0 get sin neg cAngle 0 get cos] [cAngle 1 get cos 0 cAngle 1 get sin
} def

/perspective_project {
        1 dict begin
        /point exch def
        /cPoint cPos point vsub def
        /newPoint perspective_rot_matrix cPoint 3 matrix_multiply def
        % x
        displaySurface 2 get newPoint 2 get 0.1 add div newPoint 0 get mul displaySurface 0 get add
        % y
        displaySurface 2 get newPoint 2 get 0.1 add div newPoint 1 get mul displaySurface 1 get add
        end
} def

% why can I not do mod with decimals
/better_mod {
        2 dict begin
        /n exch def
        /m exch def
        m m n div truncate n mul sub
        end
} def

/dot_product {
        2 dict begin
        /b exch def
        /a exch def
        /result 0 def
        0 1 a length 1 sub {
                /i exch def
                /result a i get b i get mul result add def
        } for
        result
        end
} def

/normal_tri {
        1 dict begin
        /points exch def
        /A points 1 get points 0 get vsub def
        /B points 2 get points 0 get vsub def
        /normal [
                A 1 get B 2 get zMult div mul A 2 get zMult div B 1 get mul sub
                A 2 get zMult div B 0 get mul A 0 get B 2 get zMult div mul sub
                A 0 get B 1 get mul A 1 get B 0 get mul sub
        ] def
        normal
        end
} def

/midpoint {
        1 dict begin
        /points exch def
        /total [points 0 get length {0} repeat] def
        0 1 points length 1 sub {
                /i exch def
                0 1 points i get length 1 sub {
                        /j exch def
                        total j total j get points i get j get add put
                } for
        } for
        0 1 total length 1 sub {
```

```
                /i exch def
                total i total i get points length div put
        } for
        total
        end
} def

/normal_camera {
        /cPos2 [cPos 1 get neg cPos 0 get 0] def
        /normal [
                cPos 1 get cPos2 2 get zMult div mul cPos 2 get zMult div cPos2 1 get mul sub
                cPos 2 get zMult div cPos2 0 get mul cPos 0 get cPos2 2 get zMult div mul sub
                cPos 0 get cPos2 1 get mul cPos 1 get cPos2 0 get mul sub
        ] def
        normal
} def

% from https://stackoverflow.com/questions/5666222/3d-line-plane-intersection
/line_plane_intersect {
        4 dict begin
        /plane_normal exch def
        /plane_point exch def
        /line_point1 exch def
        /line_point2 exch def

        plane_normal ==

        /u line_point2 line_point1 vsub def
        /dot plane_normal u dot_product def
        dot ==
        dot abs zMult ge {
                /w line_point1 plane_point vsub def
                /fac plane_normal w dot_product neg dot div def
                /u u fac scalar def
                line_point1 u vadd ==
                false
        } {
                false
        } ifelse
        end
} def

/back_face_cull {
        1 dict begin
        /points exch def
        % points midpoint points normal_tri cPos cPos cAngle normal_camera line_plane_intersect
        % points normal_tri 2 get 0 ge
        end
} def

% note to me: pay attention to winding order
/create_polygon {
        1 dict begin
        /points exch def
        points 0 get perspective_project moveto
        1 1 points length 1 sub {
                /i exch def
                points i get perspective_project lineto
        } for
        end
} def

/polygon {
        1 dict begin
        /points exch def
        points 0 get 0 get cvi 255 mod 255 div points 0 get 1 get cvi 255 mod 255 div points 0 get 2 get cvi 255 mod 255 div set
        newpath
        points create_polygon
```

```
        closepath
        (both) render_method eq { gsave fill grestore stroke } { (full) render_method eq { fill }{ stroke } ifelse } ifelse
        end
} def

/cube {
        2 dict begin
        /size exch def
        /pos exch def
        /x pos 0 get def
        /y pos 1 get def
        /z pos 2 get def
        /zSize size zMult mul def
        /tris [
                % top tris
                [[x y zSize] [x size add y zSize] [x y size add zSize]]
                [[x y size add zSize] [x size add y zSize] [x size add y size add zSize]]
                % bottom tris
                [[x y z] [x y size add z] [x size add y z]]
                [[x y size add z] [x size add y size add z] [x size add y z]]
                % front tris
                [[x y z] [x size add y z] [x y zSize]]
                [[x y zSize] [x size add y z] [x size add y zSize]]
                % left tris
                [[x y zSize] [x y size add z] [x y z]]
                [[x y zSize] [x y size add z] [x y size add zSize]]
                % right tris
                [[x size add y size add z] [x size add y zSize] [x size add y size add zSize]]
                [[x size add y z] [x size add y size add z] [x size add y zSize]]
                % back tris
                [[x y size add z] [x y size add zSize] [x size add y size add z]]
                [[x y size add zSize] [x size add y size add zSize] [x size add y size add z]]

        ] def
        0 1 tris length 1 sub {
                /i exch def
                % i 12 div i 12 div i 12 div setrgbcolor
                tris i get polygon
        } for
        end
} def

/square {
        2 dict begin
        /size exch def
        /pos exch def
        /x pos 0 get def
        /y pos 1 get def
        /z pos 2 get def
        /zSize size zMult mul def
        /points1 [[x y z] [x y size add z] [x size add y z]] def
        /points2 [[x y size add z] [x size add y size add z] [x size add y z]] def

        points1 polygon
        points2 polygon

        end
} def
% algorithm from http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html
/icosphere {
        2 dict begin
        /subdivisions exch def
        /radius exch def
        /center exch def

        /vertices 12 array def
        /faces 20 array def
        /to_unit_sphere {
```

```
        1 dict begin
        /point exch def
        /length point 0 get 2 exp point 1 get 2 exp add point 2 get 2 exp add sqrt def
        [point 0 get length div point 1 get length div point 2 get length div]
        end
} def

/middle {
        2 dict begin
        /point2 exch def
        /point1 exch def
        [point1 0 get point2 0 get add 2 div point1 1 get point2 1 get add 2 div point1 2 get point2 2 get add 2 div]
        end
} def

/t 5 sqrt 1 add 2 div def

vertices 0 [-1 t 0] to_unit_sphere put
vertices 1 [1 t 0] to_unit_sphere put
vertices 2 [-1 t neg 0] to_unit_sphere put
vertices 3 [1 t neg 0] to_unit_sphere put

vertices 4 [0 -1 t] to_unit_sphere put
vertices 5 [0 1 t] to_unit_sphere put
vertices 6 [0 -1 t neg] to_unit_sphere put
vertices 7 [0 1 t neg] to_unit_sphere put

vertices 8 [t 0 -1] to_unit_sphere put
vertices 9 [t 0 1] to_unit_sphere put
vertices 10 [t neg 0 -1] to_unit_sphere put
vertices 11 [t neg 0 1] to_unit_sphere put

faces 0 [vertices 0 get vertices 11 get vertices 5 get] put
faces 1 [vertices 0 get vertices 5 get vertices 1 get] put
faces 2 [vertices 0 get vertices 1 get vertices 7 get] put
faces 3 [vertices 0 get vertices 7 get vertices 10 get] put
faces 4 [vertices 0 get vertices 10 get vertices 11 get] put

faces 5 [vertices 1 get vertices 5 get vertices 9 get] put
faces 6 [vertices 5 get vertices 11 get vertices 4 get] put
faces 7 [vertices 11 get vertices 10 get vertices 2 get] put
faces 8 [vertices 10 get vertices 7 get vertices 6 get] put
faces 9 [vertices 7 get vertices 1 get vertices 8 get] put

faces 10 [vertices 3 get vertices 9 get vertices 4 get] put
faces 11 [vertices 3 get vertices 4 get vertices 2 get] put
faces 12 [vertices 3 get vertices 2 get vertices 6 get] put
faces 13 [vertices 3 get vertices 6 get vertices 8 get] put
faces 14 [vertices 3 get vertices 8 get vertices 9 get] put

faces 15 [vertices 4 get vertices 9 get vertices 5 get] put
faces 16 [vertices 2 get vertices 4 get vertices 11 get] put
faces 17 [vertices 6 get vertices 2 get vertices 10 get] put
faces 18 [vertices 8 get vertices 6 get vertices 7 get] put
faces 19 [vertices 9 get vertices 8 get vertices 1 get] put

0 1 subdivisions 1 sub {
        /faces2 faces length 4 mul cvi array def
        /faceIdx 0 def
        0 1 faces length 1 sub {
                /i exch def
                /face faces i get def
                /a face 0 get def
                /b face 1 get def
                /c face 2 get def
                /m1 a b middle def
                /m2 b c middle def
                /m3 c a middle def
```

```
                        faces2 faceIdx [a m1 m3] put
                        /faceIdx faceIdx 1 add def
                        faces2 faceIdx [b m2 m1] put
                        /faceIdx faceIdx 1 add def
                        faces2 faceIdx [c m3 m2] put
                        /faceIdx faceIdx 1 add def
                        faces2 faceIdx [m1 m2 m3] put
                        /faceIdx faceIdx 1 add def
                } for
                /faces faces2 [exch aload pop] def
        } for
        0 1 faces length 1 sub {
                /i exch def
                /tri faces i get def
                0 1 2 {
                        /j exch def
                        /point tri j get to_unit_sphere def
                        /newPoint [point 0 get radius mul center 0 get add point 1 get radius mul center 1 get add point 2 get r
                        tri j newPoint put
                } for
                enable_back_face_culling { tri back_face_cull true eq {tri polygon} if } { tri polygon } ifelse
                points midpoint perspective_project moveto
                points normal_tri perspective_project lineto
                stroke
        } for
        end
} def

/draw_axes {
        newpath
        [0 0 0] perspective_project moveto
        [1000 0 0] perspective_project lineto
        closepath
        1 0 0 setrgbcolor
        stroke
        newpath
        [0 0 0] perspective_project moveto
        [0 1000 0] perspective_project lineto
        closepath
        0 1 0 setrgbcolor
        stroke
        newpath
        [0 0 0] perspective_project moveto
        [0 0 -1000] perspective_project lineto
        closepath
        0 0 1 setrgbcolor
        stroke
} def
```

The next code is my working file that I use as a testbed. The current version of the code displays a 50x50 grid of a dual glider gun pattern in Conway's Game of Life.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 500 500
% import 3d lib
(3d.eps) run


realtime srand

/size 750 def
/cycles 1080 def

% disable back-face culling (because it doesn't work correctly yet)
/enable_back_face_culling false def
% full or wireframe (or both)
/render_method (full) def

size 3 div size 3 div translate

/gridsize 50 def
/cellsize size gridsize div def
cellsize 150 div setlinewidth

% used for GOL: one is used to display, one is used to calculate because all the cells have to be updated at the same time
/old [
        0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
        1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
] def
/current old [exch aload pop] def
/heatmap [gridsize gridsize mul {0} repeat] def

/getneighbors {
        1 dict begin
        /i exch def
        end
        /neighbors [
                i 1 gridsize add sub i gridsize sub i 1 gridsize sub sub
                i 1 sub                               i 1 add
                i 1 gridsize add add i gridsize add i 1 gridsize sub add
        ] def
        /l 0 def
        0 1 7 {
                /j exch def
                /c neighbors j get def
                /valid c 0 gt c gridsize gridsize mul lt and def
                valid {
                        old c get 1 eq {/l l 1 add def} if
                } if
        } for
        l
} def

/live {
        1 dict begin
        /i exch def end
        /l i getneighbors def

        /x i gridsize mod cellsize mul def
        /y i gridsize idiv cellsize mul def
        l 2 ne l 3 ne and {current i 0 put} {current i 1 put } ifelse
} def

/dead {
        1 dict begin
        /i exch def        end

        /l i getneighbors def

        /x i gridsize mod cellsize mul def
        /y i gridsize idiv cellsize mul def
        l 3 eq {current i 1 put} {current i 0 put} ifelse
} def
/cell {
        3 dict begin
        /state exch def
        /y exch def
        /x exch def
        state 1 eq { 55 255 div 63 255 div 81 255 div setrgbcolor [x y 0] cellsize cube } if
        end
} def

/wipe {
        /old_color [ currentrgbcolor ] def
        newpath
        size neg size neg moveto
        size size neg lineto
```

```
        size size lineto
        size neg size lineto
        closepath
        1 1 1 setrgbcolor
        fill
        old_color aload setrgbcolor
} def


% custom camera offsets for this example
/cPosStart [2 5 div size mul .001 add 2 5 div size mul .001 add 0.5] def
/cPos cPosStart def
/cAngle [-0.05 0.001 135.001] def

/cSpeed 1 def
/circleSize 500 def

/timeTaken 0 def

/points [[0 10 0.32] [30 -20 0.16] [0 0 0]] def
0 1 cycles {
        /timeTaken realtime def
        /theta exch def
        /cPos [theta cSpeed mul cos circleSize mul cPosStart 0 get add theta cSpeed mul sin circleSize mul cPosStart 1 get add c
        /cAngle [cAngle 0 get cAngle 1 get theta cSpeed mul 90 sub] def
        %draw_axes
        %0 1 0 setrgbcolor
        %[200 200 100] 100 0 icosphere
        %1 0 0 setrgbcolor
        [200 200 50] 200 2 icosphere
        %0 0 1 setrgbcolor
        %[200 200 0] 300 2 icosphere
        % [0 0 0] cellsize gridsize mul square
        %223 255 div 187 255 div 177 255 div setrgbcolor [0 0 0] cellsize gridsize mul square
        %0 1 gridsize 2 exp 1 sub {
        %       /i exch def
        %       /x i gridsize mod cellsize mul def
        %       /y i gridsize idiv cellsize mul def
        %       x y current i get cell
        %} for
        %theta 2 mod 1 eq {
        %       0 1 gridsize 2 exp 1 sub {
        %               /i exch def
        %               % recalculate
        %               old i get 0 eq {i dead}{i live} ifelse
        %       } for
        %       /old current [exch aload pop] def
        %} if
        /timeTaken realtime timeTaken sub def
        timeTaken 1000 div ==
        copypage
} for

%EOF
```