AN AFFINE A_2 BUILDING FRAGMENT IN TIKZ

GREG KUPERBERG

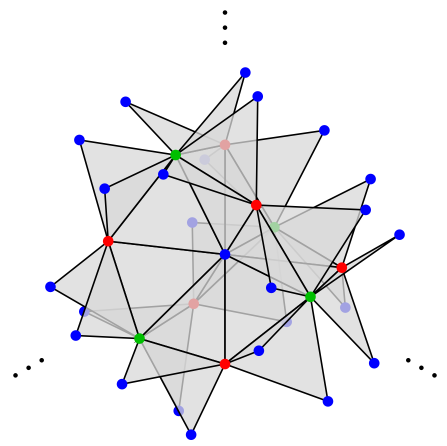

Figure 1. A fragment of an affine A_2 building

This is a fragment of an affine A_2 building in TikZ. As far as I know, the fragment embeds in every thick affine building of A_2 type. The TikZ code is generated from SAGE. I have included the SAGE code starting on the next page.

```
# Code to generate a fragment of an A_2 building

scale = 3           # Rescaling in 3D
dotrad = '.1'       # Size of the dots for vertices

def p(v): return '(%.3f,%.3f)' % (v[0],v[2])    # Planar projection
def cross(v,w): return v.cross_product(w)       # Just an abbreviation

# Set up the transformation for the eye position
degrees = RR(pi/180)
(phi,theta) = (15*degrees, -25*degrees)
r1 = matrix(RR,[[cos(phi),-sin(phi),0],[sin(phi),cos(phi),0],[0,0,1]])
r2 = matrix(RR,[[1,0,0],[0,cos(theta),-sin(theta)],[0,sin(theta),cos(theta)]])
r = scale*r2*r1

# Generate the rotation group of the tetrahedron
i = identity_matrix(RR,3)
a = matrix(RR,[[-1,0,0],[0,-1,0],[0,0,1]])
b = matrix(RR,[[1,0,0],[0,-1,0],[0,0,-1]])
c = matrix(RR,[[0,1,0],[0,0,1],[1,0,0]])
a4 = [i,a,b,a*b]
a4 = a4 + [c*g for g in a4] + [c*c*g for g in a4]

# Set up a table of named points
points = {}
def addpt(v):
    v = p(v)
    if not v in points: points[v] = '(p%02d)' % (len(points)+1)
    return points[v]

# Set up a table of triangles
triangles = []
def addtriangle(m,v1,v2,v3):
    (v1,v2,v3) = (m*v1,m*v2,m*v3)
    z = (v1+v2+v3)[1]
    (v1,v2,v3) = (addpt(v1),addpt(v2),addpt(v3))
    triangles.append((z,[
        r'\fill[lightgray,nearly opaque] %s -- %s -- %s;' % (v1,v2,v3),
        r'\draw[semithick] %s -- %s -- %s -- cycle;' % (v1,v2,v3),
        r'\fill[darkgreen] %s circle (%s);' % (v2,dotrad),
        r'\fill[red] %s circle (%s);' % (v3,dotrad),
        r'\fill[blue] %s circle (%s);' % (v1,dotrad),
        ]))

# The actual work to build the figure
v1 = vector(RR,[1,1,1])/RR(sqrt(3))
v2 = vector(RR,[1,0,0])
v3 = .8*(v1+v2)+.7*cross(v1,v2)
v4 = .8*(v1+v2)-.7*cross(v1,v2)
o = vector(RR,[0,0,0])
for g in a4:
    addtriangle(r*g,o,v1,v2)
    addtriangle(r*g,v3,v1,v2)
    addtriangle(r*g,v4,v1,v2)

# Arrange the triangles with a depth-first sort
done = {}
statements = []
for (y,t) in sorted(triangles):
    for x in reversed(t):
        if not x in done: statements.append(x)
        done[x] = 1
for x in points: print r'\coordinate %s at %s;' % (points[x],x)
for x in reversed(statements): print x
```