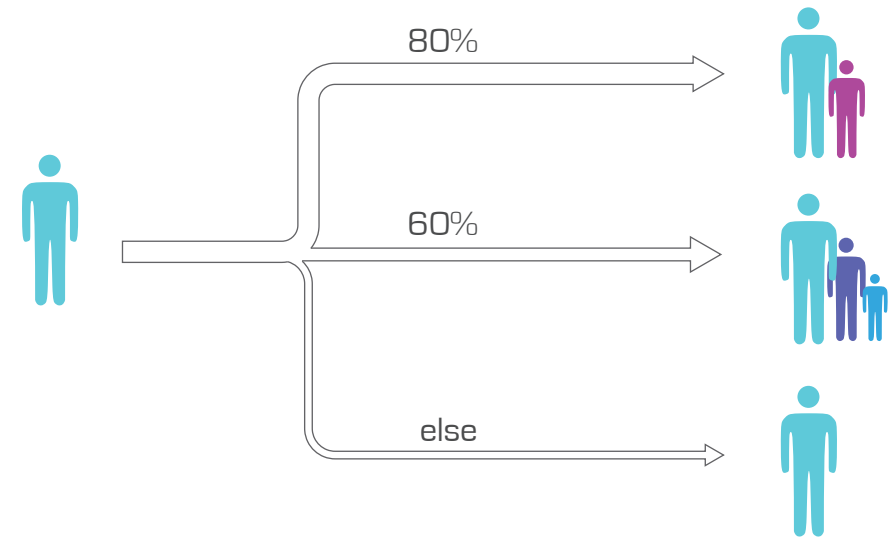


**A\_logic of child # percentage**

```
int childNum=0;
if (!Random.percent(80)) then # of child=1;
else if (!Random.percent(60)) then # of child = 2;
else then # of child=0;
```



**B\_types of directions**

```
double dir = IRandom.getInt(0, 7);
dir==0 nextPos[axis].len(nextSize);
dir==1 nextPos[axis].len(-nextSize);
dir==2 cross[Vec.zaxis].len(nextSize);
dir==3 cross[Vec.zaxis].len(-nextSize);
dir==4 cross[Vec.yaxis].len(nextSize);
dir==5 cross[Vec.yaxis].len(-nextSize);
dir==6 cross[Vec.xaxis].len(nextSize);
dir==7 cross[Vec.xaxis].len(-nextSize);
```

IVec nextAxis = axis.dup().rot[Vec.xaxis, PI/20];

**C\_geometry definition**

I[Surface] createTetrahedron (I[Vec pt, double size, double height])

I[Surface] createOctahedron (I[Vec pt, double size, double height])



**IGEO.CODE DEFINITION**

```
import processing.opengl.*;
import igeo.*;

void setup() {
  size(600, 600, IG.GL);
  IRandom.init(1);
  IG.duration(100);
  int num = 1;
  for (int i=0; i < num; i++) {
    new MyAgent(IRandom.pt(30, 30, 0), 10, 5, 5, new IVec(1, 1, 1)).clr(1.25);
  }
  IG.transparent[0]; //transparent graphic mode
}

static class MyAgent extends IAgent {
  IVec pos;
  IVec axis;

  double size, width1, height1;
  I[Surface] polyhedronSrf;
  boolean changed=true;

  MyAgent(I[Vec pt, double w, double h, double sz, IVec axis] {
    pos = pt;
    size = sz;
    width1=w;
    height1=h;
    axis = axis;
  }

  void update() {
    super.update();

    if [changed] {
```

```
if [polyhedronSrf!=null] {
  for [ISurface srf.polyhedronSrf] {
    srf.del();
  }
}

//polyhedronSrf = createTetrahedron(pos, size, size);
if [!Random.percent(20)] {
  polyhedronSrf = createTetrahedron(pos, size, size);
}
else {
  polyhedronSrf = createOctahedron(pos, size, size*sqrt(2));
}

for [ISurface srf.polyhedronSrf] {
  srf.clr(clr());
  srf.rot(pos, axis, PI/6);
}

changed=false;

//child number
int childNum=0;

if [!Random.percent(80)] {
  childNum=1;
}
else if [!Random.percent(60)] {
  childNum = 2;
}
else {
  childNum=0;
}

for [int i=0; i<childNum; i++) {
```

```
// next unit size = child sizes
double nextSize = IRandom.getDouble(.95, 1) * size;
IVec nextPos = pos.dup();
double dir = IRandom.getInt(0, 7);

if [time==0] //delayed to create the next agent til time==0
if [dir==0] nextPos.add[axis.dup().len(nextSize)];
else if [dir==1] nextPos.add [axis.dup().len(-nextSize)];
else if [dir==2] nextPos.add [axis.cross[Vec.zaxis].len(nextSize)];
else if [dir==3] nextPos.add [axis.cross[Vec.zaxis].len(-nextSize)];
else if [dir==4] nextPos.add [axis.cross[Vec.yaxis].len(nextSize)];
else if [dir==5] nextPos.add [axis.cross[Vec.yaxis].len(-nextSize)];
else if [dir==6] nextPos.add [axis.cross[Vec.xaxis].len(nextSize)];
else {
  nextPos.add [axis.cross[Vec.xaxis].len(-nextSize)];
}

IVec nextAxis = axis.dup().rot[Vec.xaxis, PI/20];

int r = clr().getRed() + IRandom.getInt(-100, 100);
int g = clr().getGreen() + IRandom.getInt(-100, 100);
int b = clr().getBlue() + IRandom.getInt(-100, 100);
new MyAgent(nextPos, nextSize, nextSize, nextSize,
nextAxis).clr(r, g, b);
}

I[Surface] createTetrahedron
(I[Vec pt, double size, double height] {
  // tetrahedron
  IVec vertexDir = new IVec(size, 0, 0);
  IVec v1 = pt.cp[vertexDir];
  vertexDir.rot[Vec.zaxis, PI*2/3];
  IVec v2 = pt.cp[vertexDir];
```

```
vertexDir.rot[Vec.zaxis, PI*2/3];
IVec v3 = pt.cp[vertexDir];
IVec v4 = pt.cp[0, 0, height];

I[Surface] triangles = new I[Surface]4;
triangles[0] = new I[Surface]v1.dup(), v2.dup(), v3.dup();
triangles[1] = new I[Surface]v1.dup(), v2.dup(), v4.dup();
triangles[2] = new I[Surface]v1.dup(), v3.dup(), v4.dup();
triangles[3] = new I[Surface]v2.dup(), v3.dup(), v4.dup();

return triangles;
}

I[Surface] createOctahedron
(I[Vec pt, double size, double height] {
  // tetrahedron
  IVec v1 = pt.cp[size/2, size/2, 0];
  IVec v2 = pt.cp[-size/2, size/2, 0];
  IVec v3 = pt.cp[-size/2, -size/2, 0];
  IVec v4 = pt.cp[size/2, -size/2, 0];
  IVec v5 = pt.cp[0, 0, height/2];
  IVec v6 = pt.cp[0, 0, -height/2];

  I[Surface] triangles = new I[Surface]8;
  triangles[0] = new I[Surface]v1.dup(), v2.dup(), v5.dup();
  triangles[1] = new I[Surface]v2.dup(), v3.dup(), v5.dup();
  triangles[2] = new I[Surface]v3.dup(), v4.dup(), v5.dup();
  triangles[3] = new I[Surface]v4.dup(), v1.dup(), v5.dup();
  triangles[4] = new I[Surface]v4.dup(), v3.dup(), v6.dup();
  triangles[5] = new I[Surface]v3.dup(), v2.dup(), v6.dup();
  triangles[6] = new I[Surface]v2.dup(), v1.dup(), v6.dup();
  triangles[7] = new I[Surface]v1.dup(), v4.dup(), v6.dup();

  return triangles;
}
}
```



```
# of child
int childNum=0;
if (!Random.percent(80))
  childNum=1;
else if (!Random.percent(60))
  childNum = 2;
else
  childNum=0;

nextSize
double nextSize =IRandom.getDouble(.9, 1.1) * size;

nextAxis
IVec nextAxis = axis.dup().rot[Vec.xaxis, PI/20];
```

```
# of child
int childNum=0;
if (!Random.percent(60))
  childNum=1;
else if (!Random.percent(50))
  childNum = 2;
else
  childNum=0;

nextSize
double nextSize =IRandom.getDouble(.95, 1) * size;

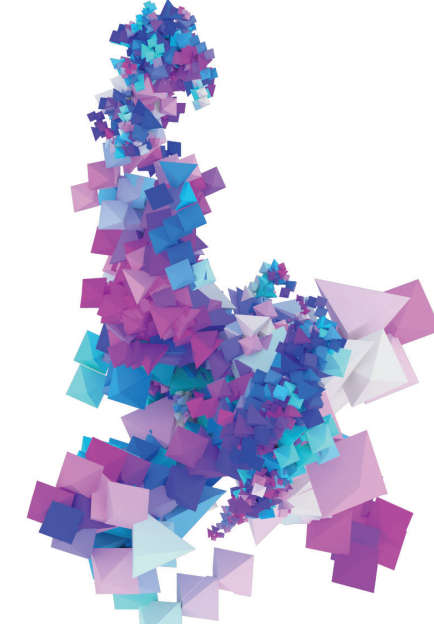
nextAxis
IVec nextAxis = axis.dup().rot[Vec.xaxis, PI/10];
```

```
# of child
int childNum=0;
if (!Random.percent(80))
  childNum=1;
else if (!Random.percent(60))
  childNum = 2;
else
  childNum=0;

nextSize
double nextSize =IRandom.getDouble(.95, 1) * size;

nextAxis
IVec nextAxis = axis.dup().rot[Vec.xaxis, PI/20];
```

**PARAMETER CHANGES**



**TOP VIEWS**



**AXONOMETRICS**

