SCI Arc - Spring 2013
EST/m
instructor: Satoru sugihara
student: Ehsan jelveh moghaddam

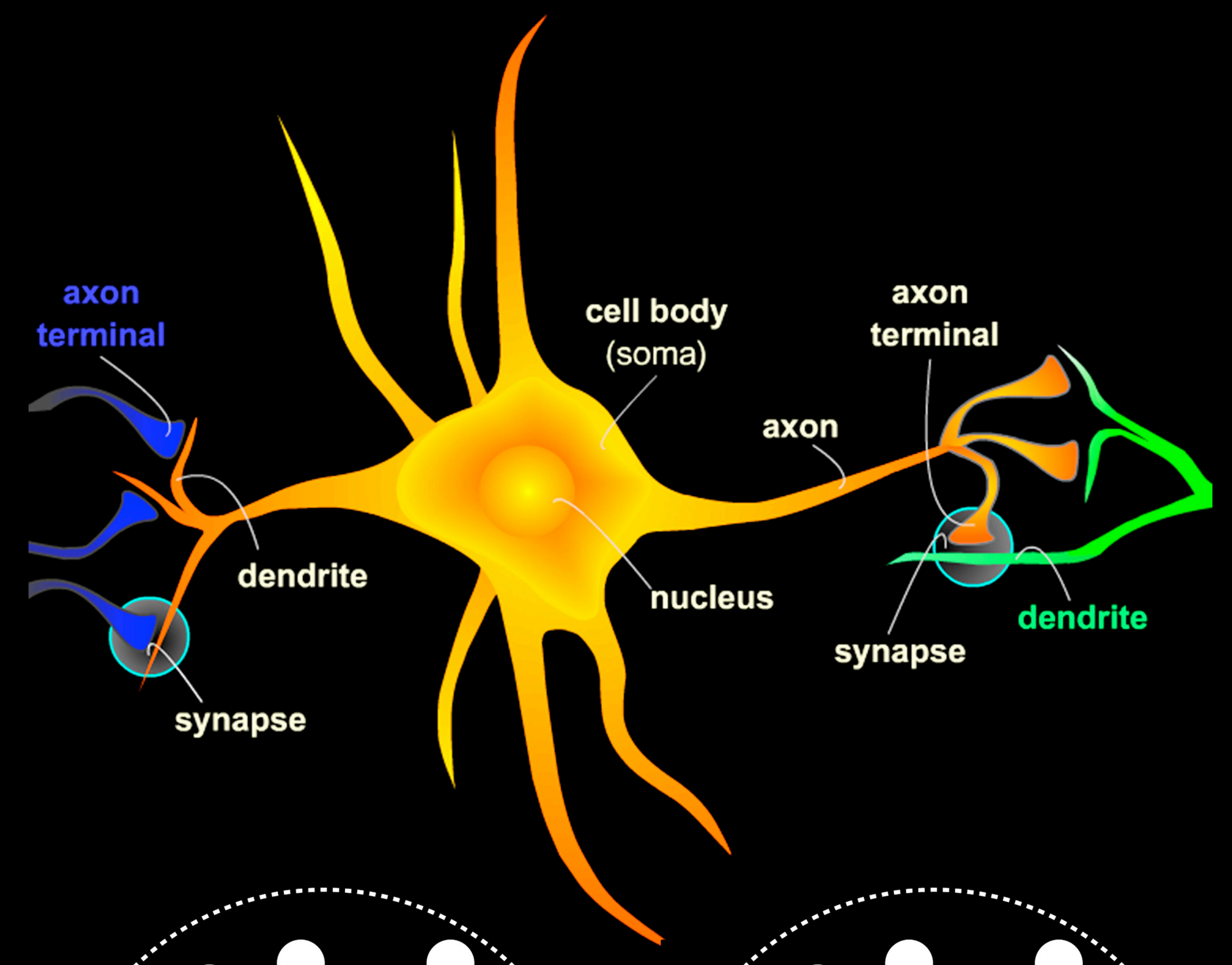
advance coding form

Neural Networks

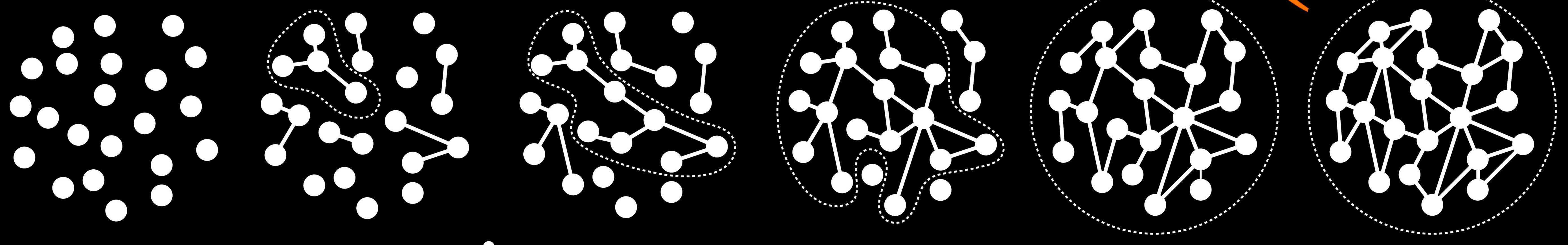
THE COMPUTER SIMULATION OF A COMPLEX SYSTEM

neuron structure

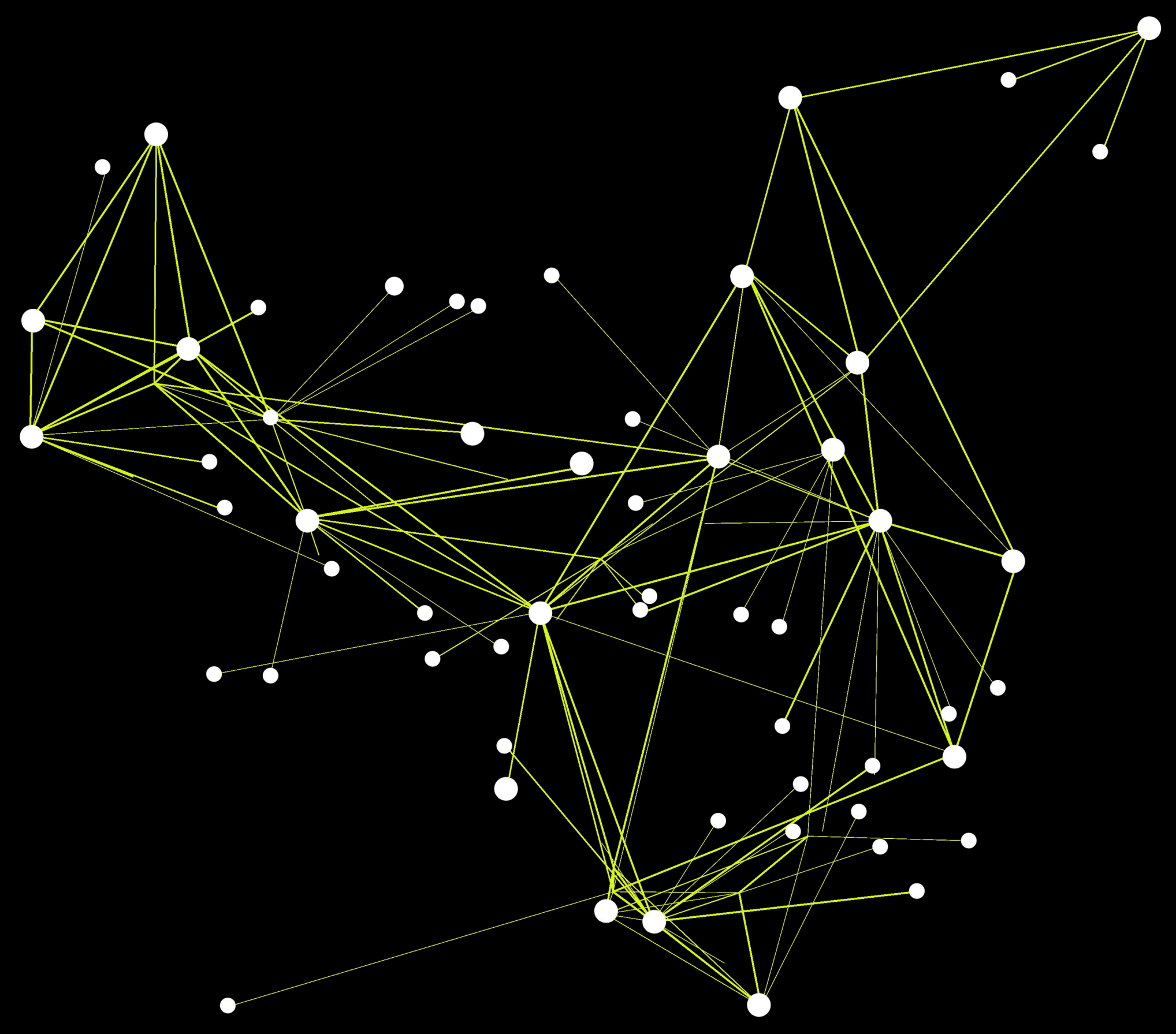
The human brain can be described as a biological neural network—an interconnected web of neurons transmitting elaborate patterns of electrical signals. Dendrites receive input signals and, based on those inputs, fire an output signal via an axon.



THE RELATION BETWEEN NEURONS AND AXONS IS BASED ON ATTRACTION POINTS AND TIME.WHEN NEURONS TRYING TO SHOOT THE AXONS.NEURONS ATTRACT THE NEAREST AXON.

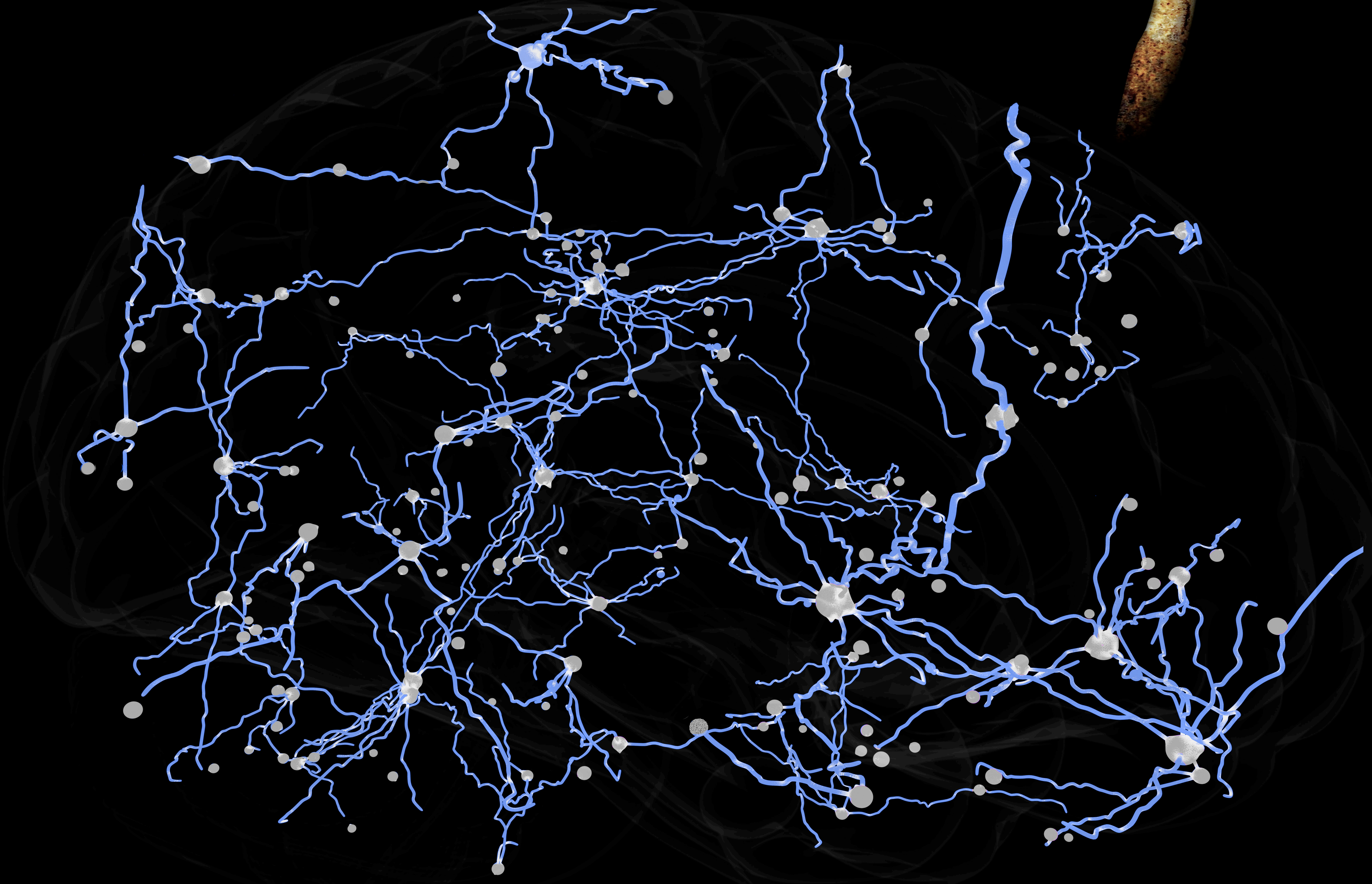


```
void interact(ArrayList<IDynamics> agents) {
  for (int i=0; i<agents.size(); i++) {
    if (agents.get(i) instanceof Axon) {
      Axon a = (Axon)agents.get(i);
      if (a.parent != this) {
        if ( a.pos().dist(pos) < 40 ) {
          a.trajectory().addCP(pos);
          a.makePipe();
          a.del(false);
          intensity += 1.0;

          if(a.prevPos!=null){
            deformSphere(a.prevPos.dif(pos));
          }
          else{
            deformSphere(a.pos().dif(pos));
          }
```



```
class Neuron extends IAgent {
  IVec pos, dir;
  IMesh sphere;
  double radius = 0.5; //0.1;
  double maxRadius = 1.2;
  double sphereGrowthRatio = 1.01;
  double intensity = 10;//100;
  boolean blink=false;
  Neuron(IVec p, IVec d) {
    pos = p;
    dir = d;
    sphere = IG.meshSphere(pos, radius, 15).clr(1.0);
  }
  void interact(ArrayList<IDynamics> agents) {
    for (int i=0; i<agents.size(); i++) {
      if (agents.get(i) instanceof Axon) {
        Axon a = (Axon)agents.get(i);
        if (a.parent != this) {
          if ( a.pos().dist(pos) < 3 ) {
            a.trajectory().addCP(pos);
            a.makePipe();
            a.del(false);
            intensity += 1.0;
            if(a.prevPos!=null){
              deformSphere(a.prevPos.dif(pos));
            }
            else{
              deformSphere(a.pos().dif(pos));
            }
            if(radius<maxRadius){
              sphere.scale(pos,sphereGrowthRatio);
              radius *= sphereGrowthRatio;
            }
            blink=true;
          }
        }
      }
    }
```

```
void makePipe(){
  double radius = 0.01;

  ICurve traj = trajectory();
  ArrayList<IVec> pts = new ArrayList<IVec>();
  pts.add(traj.cp(1));
  for(int i=1; i<traj.cpNum()-1; i++){
    if(!traj.cp(i).eq(pts.get(pts.size()-1)) && traj.cp(i).dif(pts.get(pts.size()-1)).angle(traj.cp(i+1).dif(traj.cp(i))) < PI*0.6){
      pts.add(traj.cp(i));
    }
  }
  if(!traj.cp(traj.cpNum()-1).eq(pts.get(pts.size()-1))){
    pts.add(traj.cp(traj.cpNum()-1));
  }
  if(pts.size()>1){
    IG.meshPolygonStick(new ICurveGeo(pts.toArray(new IVec[pts.size()])), 0.1, 6).clr(10.0,0,0.1); //
  }
 }
```



```
class Axon extends IParticleTrajectory {
  IPoint point;
  Neuron parent;
  IVec prevPos;

  Axon(IVec pos, IVec vel, Neuron par) {
    super(pos, vel);
    parent = par;
    fric(0.7);
    size(2);
  }
  void interact(ArrayList < IDynamics > agents) {
    Neuron closest=null;
    double minDist=0;
    for (int i=0; i < agents.size(); i++) {
      if (agents.get(i) instanceof Neuron) {
        Neuron a = (Neuron)agents.get(i);
        if(a!=parent){
        if (closest==null || a.pos.dist(pos()) < minDist) {
          minDist = a.pos.dist(pos());
          closest = a;
        }
        }
      }
    }

    if (closest!=null){
      IVec force = closest.pos.dif(pos());
      force.len(1000);
      push(force);
    }
  }
```