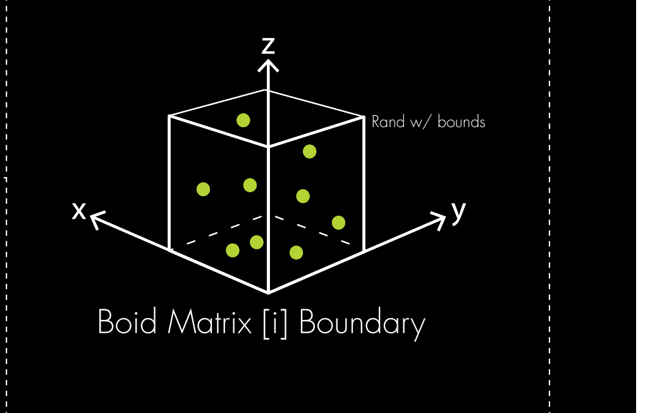
# (SM)

## Swarm[al] Morphology

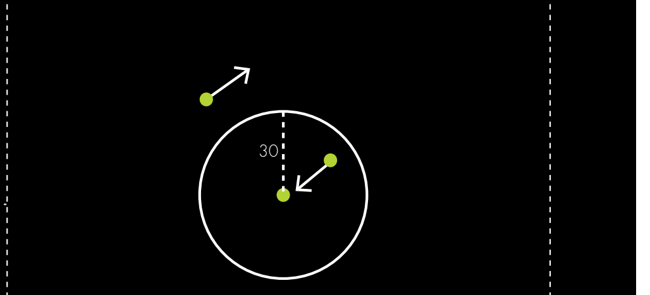
//Evan Emery
//VS Code Form
//Spring 2013

```
//SETUP/////////////////////////////////////////////////////////////////////////////////////////////

void setup() {
  size(1920, 1080, IG.GL);
  IG.duration(500);
  //IG.front();
  IG.bg(0);
  IConfig.syncDrawAndDynamics = true;
  IConfig.keyRotationSpeed = 1;

 //BOID AGENTS///////

  for (int i=0; i < 20 ; i++) {
    b1[i] = new MyBoid1(IRand.pt(10, 10, 10, 30, 30, 30), IG.v(0, 0, 0));
    b1[i].clr(179, 210, 52);
  }
  for (int i=0; i < 15 ; i++) {
    b2[i*2] = new MyBoid2(IRand.pt(30, 30, 30, 40, 40, 40), IG.v(0, 0, 0));
    b2[i*2].clr(179, 220, 52);
    b2[i*2+1] = new MyBoid2(IRand.pt(40, 40, 40, 50, 50, 50), IG.v(0, 0, 0));
    b2[i*2+1].clr(179, 230, 52);
  }

  //ATTRACTOR AGENTS///////

  for (int i=0; i < 15 ; i++) {
    a1[i] = new Attractor(IRand.pt( 0, 0, 0, 70, 70, 70), IRand.pt(-30, -30, 0, 30, 30, 30));
    a1[i].clr(155, 155, 155);
    a1[i].size(7);
  }
 //ANCHOR CLASS/////////////////////////////////////////////////////////////////////////////////////////

class Anchor extends IAgent {
  IVec pos;
  IPoint point;
  Anchor(IVec p) {
    pos = p;
    point = new IPoint(pos).clr(179, 210, 52).size(2);
    if (anchorSphere) {
      new ISphere(pos, 0.25).clr(179, 210, 52);
    }
    observe.anchors.add(this); // register to observer
  }

  void interact(ArrayList < IDynamics > agents) {
    if (time()==0) { // only when the first time
      for (int i=0; i < agents.size(); i++) {
        if (agents.get(i) instanceof Anchor) {
          Anchor a = (Anchor)agents.get(i);
          if (a!=this && a.time() > 0) { // exclude anchors just created
            if (a.pos.dist(pos) < 20) { // closer than
              ICurve line = new ICurve(a.pos, pos).clr(1.0, 0.1);
              if (pipeAnchor) {
                IG.meshRoundStick(a.pos, pos, .05);
              }
              observe.lines.add(line); // register to observer
            }
          }
        }
      }
    }
  }
}


//ATTRACTOR CLASS [PREY]//////////////////////////////////////////////////////////////////////////////

class Attractor extends IParticle {

  //ATTRACTION PROPERTIES
  double attraction = 15;
  double threshold = 30;
  //SPHERE SIZE [REPRESENTATION]
  double radius = 0.4;

  Attractor(IVec p, IVec v) {
    super(p, v);
    fric(0.05);
  }
  void interact(ArrayList < IDynamics > agents) {
    for (int i=0; i < agents.size(); i++) {
      if (agents.get(i) instanceof MyBoid1) {
        MyBoid1 b = (MyBoid1)agents.get(i);
        if (b.pos().dist(pos()) < threshold ) {
          double intensity = threshold - b.pos().dist(pos());
          IVec frc = b.pos().dif(pos).len(intensity * attraction);
          b.pull(frc);
        }
      }
      if (agents.get(i) instanceof MyBoid2) {
        MyBoid2 b = (MyBoid2)agents.get(i);
        if (b.pos().dist(pos()) < threshold ) {
          double intensity = threshold - b.pos().dist(pos());
          IVec frc = b.pos().dif(pos).len(intensity * attraction);
          b.pull(frc);
        }
      }
      if (agents.get(i) instanceof Attractor) {
        Attractor b = (Attractor)agents.get(i);

        //DRAW LINES BETWEEN ALL ATTRACTORS
        if (drawAttractorLines) {
          if (time()%80==0) {
            IG.crv(pos.cp(), b.pos().cp()).clr(0.8, 0.05);//ATTRACTOR LINES [BAKED]
            new ISphere(b.pos, radius).clr(1.0);
            new IPoint(b.pos).clr(1.0).size(4);
          }
        }
      }
    }
  }

  //UPDATE////////////////////////////////////////////////////////////////////////////////////////////

  void update() {
    if (IRand.pct(10)) {
      push(IRand.pt(-300, -300, 0, 300, 300, 300));//ATTRACTOR MOVEMENT
    }

    if (time()%15 == 0) {//HOW OFTEN ANCHOR IS CREATED
      new Anchor(pos().cp());
    }
  }
}
```

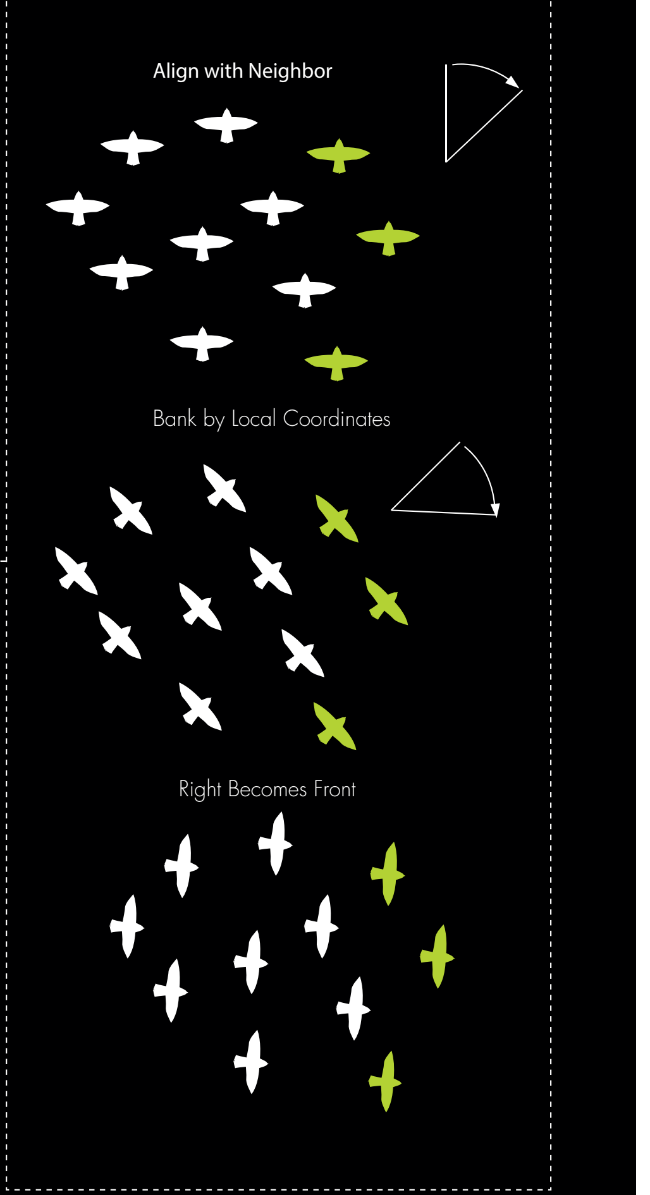
```
//BOID1 CLASS [FLOCK]/////////////////////////////////////////

class MyBoid1 extends IBoid {

 //SWARM RADIUS//
 int cohd = 65;
 int sepd = 50;
 int alid = 10;

 //LOCAL VARIABLES
 IVec prevPos;
 ISphere sphere;
 double radius;
 double size;
 int aln = 10 ;

 MyBoid1(IVec p, IVec v) {
   super(p, v);

   //SWARM RATIO//
   cohesionDist(cohd);
   cohesionRatio(10);
   separationDist(sepd);
   separationRatio(8);
   alignmentDist(alid);
   alignmentRatio(5);

   //SINGLE BOID ATTRIBUTES
   mass(IRand.get(1, 5));//RANDOM WEIGHT OF EACH BIRD
   size = IRand.get(0.4, 0.8);//SIZE BIRD 1
 }
  void interact(ArrayList < IDynamics > agents) {
   int count = 0;
   for (int i=0; i < agents.size(); i++) {
     if (agents.get(i) instanceof MyBoid1) {
       MyBoid1 b = (MyBoid1)agents.get(i);
       if (b != this) {
         if (b.pos().dist(pos()) < aln) {
           count++;
         }
       }
     }
   }

   //SELF REPRODUCTION//
   if (count == 0) {
     for (int i=0; i < 5 ; i++) {
       if (reproduceAlone) {
         new MyBoid1(pos().cp(), IG.v(0, 0, 0)).clr(clr());
       }
     }
   }
 }
 //BOID1 CURVE TRACE//
 void update() {
   IVec curPos = pos().cp();
   if (prevPos!=null) {
     if (boidTraceCurve) {
       IG.crv(prevPos, curPos).clr(1.0, 0.4);
     }
     if (pipeBoid) {
       IG.meshRoundStick(prevPos, curPos, .1);
     }
   }
   if (sphere!=null) {
     sphere.del();
   }
   prevPos = curPos;

   //BOID1 SWARM CHANGE [FLAPPING WINGS]
   if (time()%15==0) {
     cohd = 65;
     cohesionDist(cohd);
     sepd = 50;
     separationDist(sepd);
     alid = 10;
     alignmentDist(alid);
   }
   else if (time()%15==7) {
     cohd= 10;
     cohesionDist(cohd);
     sepd = 30;
     separationDist(sepd);
     alid = 5;
     alignmentDist(alid);
   }

   radius = sin(time()*2*PI/15)*size; //SCALE CHANGE OVER TIME

   //BOID1 SPHERE TRACE [FORM REPRESENTATION]
   sphere = new ISphere(curPos, radius).clr(179, 210, 52);
   if (time()%40==0) {
     if (makeSphereEverytime) {
       new ISphere(curPos, radius).clr(1.0);
     }
   }

   // BOID1 ANCHOR POINT
   if (makeAnchor) {
     if (time()%40 == 0) {
       new Anchor(pos().cp());
     }
   }
 }
}
```



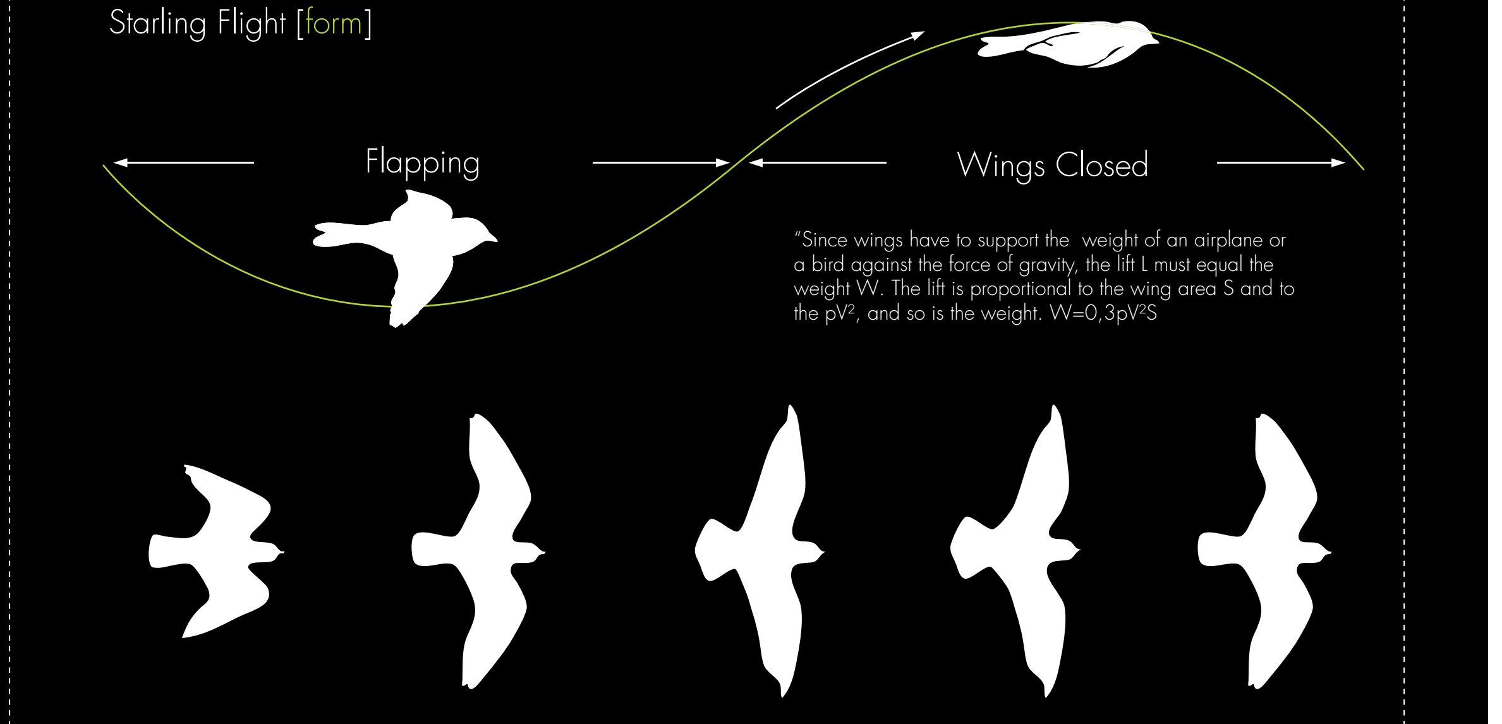
Boid Matrix [i] Boundary



Attractor Distance



Starling Flight [form]



"Since wings have to support the  weight of an airplane or a bird against the force of gravity, the lift L must equal the weight W. The lift is proportional to the wing area S and to the $pV^2$, and so is the weight. $W=0,3pV^2S$