Java Programming II
Lab9

514770-1
Fall 2024
11/27/2024

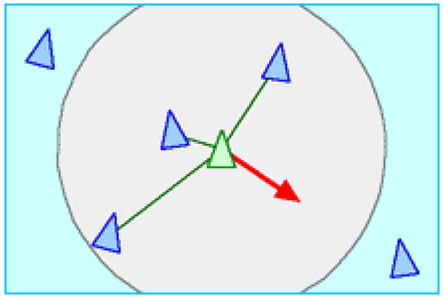
Kyoung Shin Park
Computer Engineering
Dankook University

Lab9

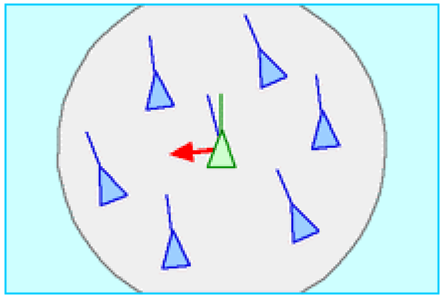
- Practice to write a program that **Flocking of Birds (Boids)** change its **State** based on its position and direction of their nearby neighbors using **State pattern.**
  - **BoidState<T> interface**
  - **WanderState, CohesionState, AlignmentState, SeperationState**

Lab9

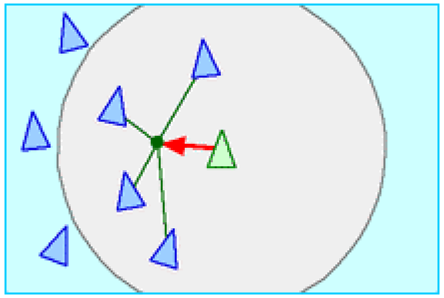
- Separation
  - Separation is a rule to **move away from nearby neighbors** to avoid crowding objects around them.



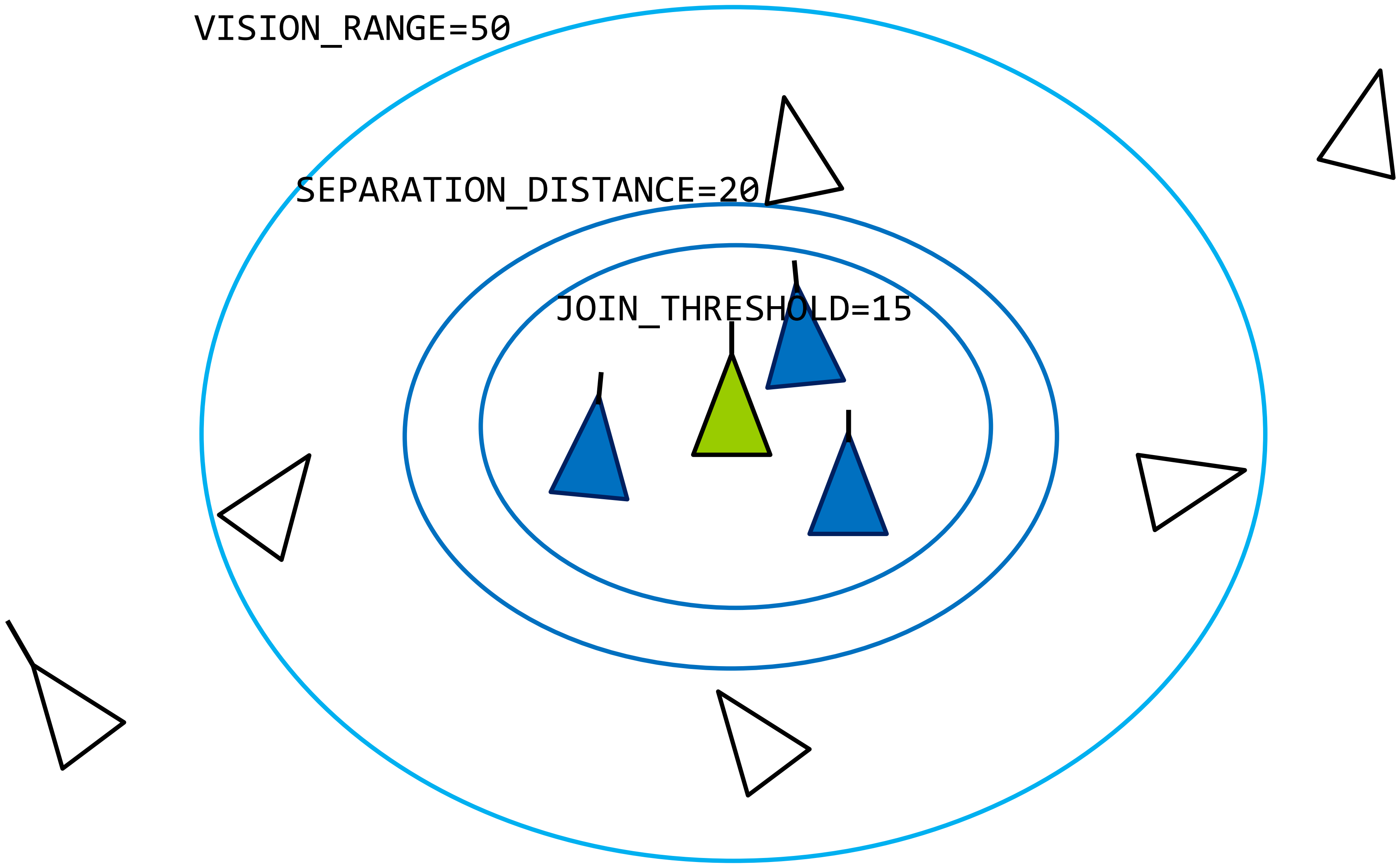
- Alignment
  - Alignment is a rule to **move toward the average direction** of neighboring objects



- Cohesion
  - Cohesion is a rule to find the midpoint (Average Position) between all neighbors and **move towards the midpoint**.

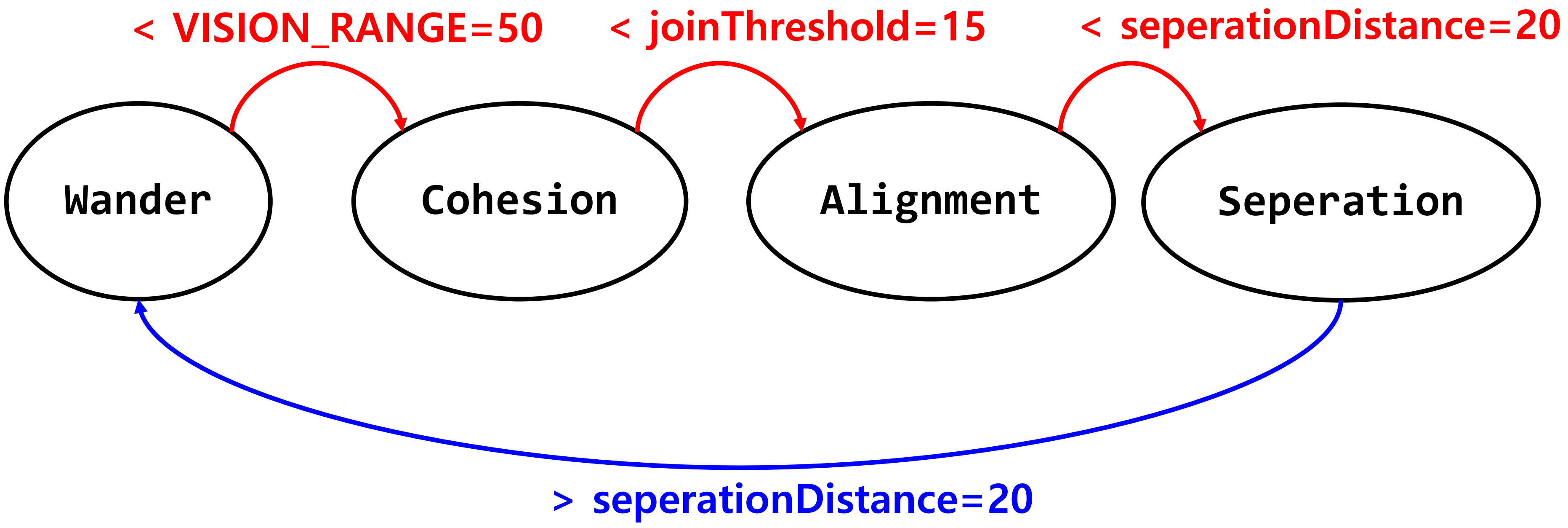


Lab9



Lab9

- **BoidState** Finite State Machine(FSM)



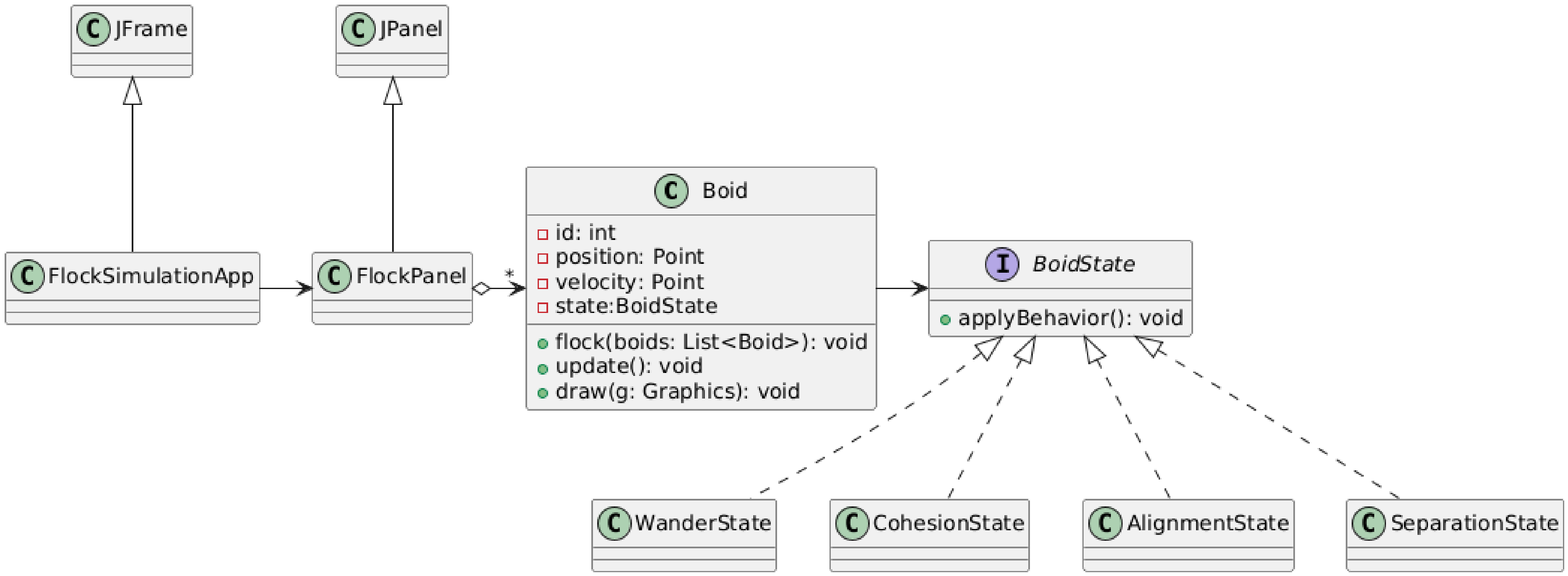
Lab9

- **WanderState**
  - Boid **moves randomly**
  - State transits to **CohesionState** if any nearby neighbors are found, i.e., boid.distance(other) < VISION_RANGE
- **CohesionState**
  - Boid **moves towards the center (Average Position)** of all neighbors (< VISION_RANGE)
  - State transits to **AlignmentState** if boid is close enough to midpoint, i.e., boid.distance(center) < JOIN_THRESHOLD

Lab9

- **AlignmentState**
  - Boid **moves towards the average direction (Average Velocity)** of all neighbors (< VISION_RANGE)
  - State transits to **SeperationState** if boid is close to neighbors, , i.e., boid.distance(other) < SEPARATION_DISTANCE
- **SeparationState**
  - Boid **moves away** from nearby neighbors to avoid crowding, i.e., boid.distance(other) < SEPARATION_DISTANCE
  - State transits to **WanderState** if boid is separated from neighbors, , i.e., boid.distance(other) > SEPARATION_DISTANCE

Lab9


JFrame
JPanel
FlockSimulationApp
FlockPanel
*
Boid
- id: int
- position: Point
- velocity: Point
- state:BoidState
+ flock(boids: List<Boid>): void
+ update(): void
+ draw(g: Graphics): void
BoidState
+ applyBehavior(): void
WanderState
CohesionState
AlignmentState
SeparationState

Lab9

```
public interface BoidState {
    void applyBehavior(Boid boid, List<Boid> boids);
}

public class WanderState implements BoidState {
    private static final int MAX_FORCE = 1;

    @Override
    public void applyBehavior(Boid boid, List<Boid> boids) {
        // random wandering behavior
        Random random = new Random();
        int dx = random.nextInt(2 * MAX_FORCE + 1) - MAX_FORCE;
        int dy = random.nextInt(2 * MAX_FORCE + 1) - MAX_FORCE;
        boid.applyForce(new Point(dx, dy));

```

Lab9

```
        // check for nearby boids to join a flock
        for (Boid other : boids) {
            if (other != boid && 
boid.getPosition().distance(other.getPosition()) 
< Boid.VISION_RANGE) {
                System.out.println("boid id=" + boid.getId() + " 
state=" + boid.getState() + " changed to CohesionState");
                boid.setState(new CohesionState());
                return; // transition to CohesionState if nearby 
boids are found
            }
        }
    }
    @Override
    public String toString() {
        return "WanderState";
    }
}
```

Lab9

```
public class Boid {
    private Point position;
    private Point velocity;
…
    private BoidState state;
    private int id; // id is automatically assigned by count
    private static int count = 0;

    public Boid(int panelWidth, int panelHeight) {
        …
        this.id = ++count;
        this.state = new WanderState();
    } …
    @Override
    public String toString() {
        // id, position, velocity, state
    }
}
```

Submit to e-learning

- Add your code (e.g., other class or design pattern, etc) in the Lab9 assignment.
- Submit the Lab9 assignment (JAVA24-2-Lab9-YourID-YourName.zip including the report) to e-learning due by 12/3.