Boids

Grupo 01

Horacio Gomez (L : 50825) Juan Pablo Orsay (L : 49373)

Segundo Semestre 2018

72.25 Simulación de Sistemas
Instituto Tecnológico de Buenos Aires

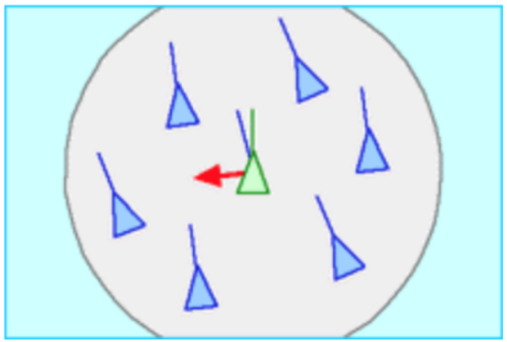
Índice

Boids: Algoritmo

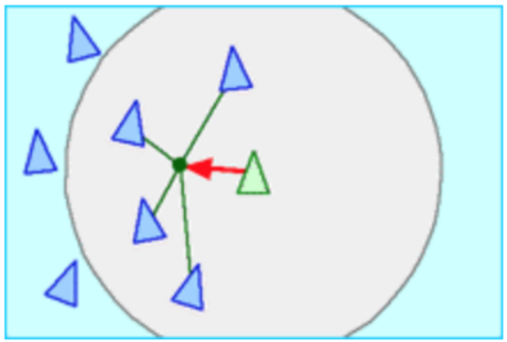
Algoritmo 1 Boids

```
1:  procedure Step(rules, factors, boids, dT)
2:      for boid ← boids do
3:          v = boid.velocity
4:          for rule ← rules do
5:              v += factor_i * rule(boid, boids)
6:          end for
7:          if magnitude(v) > MAX_SPEED then
8:              v = v/magnitude(v) * MAX_SPEED
9:          end if
10:         boid.velocity = v
11:         boid.position += boid.velocity * dT
12:     end for
13: end procedure
```

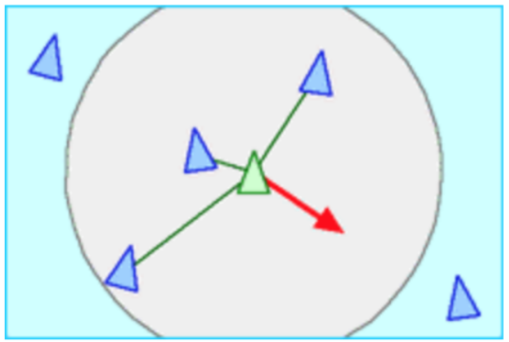
Boids: Reglas



Alineamiento



Cohesión

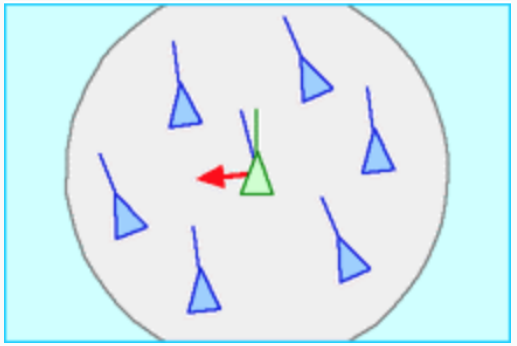


Separación

Regla: Alineamiento

Algoritmo 2 Alineamiento

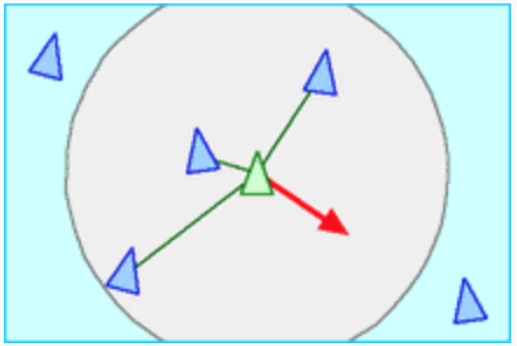
```
1: procedure Alineamiento(boid, boids)
2:     v_out = {0, 0, 0}
3:     for neighbour ← boids do
4:         v_out += neighbour.velocity
5:     end for
6:     v_out = v_out / length(neighbours)   ▷ Velocidad promedio de vecinos
7:     v_out = v_out − boid.velocity
8:     return v_out
9: end procedure
```



Regla: Separación

Algoritmo 4 Separación

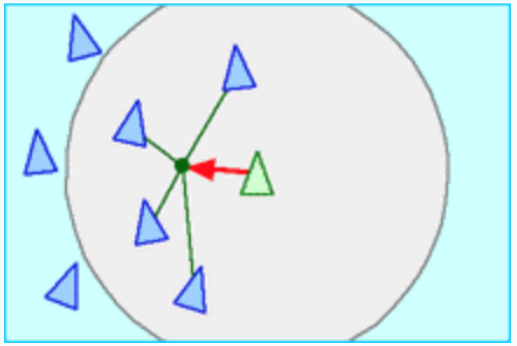
```
1:  procedure Separación(boid, boids)
2:      v_out = {0, 0, 0}
3:      for neighbour ← boids do
4:          if distance(neighbour, boid) > unsafeDistance then
5:              continue
6:          end if
7:          v_out −= boid.position − neighbour.position
8:      end for
9:      v_out = v_out/length(neighbours)        ▷ Posición promedio de vecinos
10:     v_out = v_out − boid.position
11:     return v_out
12: end procedure
```



Regla: Cohesión

Algoritmo 3 Cohesión

1: **procedure** Cohesión($boid, boids$)
2: $\quad \boldsymbol{v}_{out} = \{0, 0, 0\}$
3: $\quad$ **for** $neighbour \leftarrow boids$ **do**
4: $\quad\quad \boldsymbol{v}_{out}$ += $neighbour.position$
5: $\quad$ **end for**
6: $\quad \boldsymbol{v}_{out} = \boldsymbol{v}_{out}/length(neighbours)$ ▷ Posición promedio de vecinos
7: $\quad \boldsymbol{v}_{out} = \boldsymbol{v}_{out} - boid.position$
8: $\quad$ **return** $\boldsymbol{v}_{out}$
9: **end procedure**



Regla: Tendencia a

Algoritmo 5 TendencyTo

```
1: procedure TendencyTo(boid, neighbours, interactWith)
2:     v_out = {0, 0, 0}
3:     for neighbour ← neighbours do
4:         if neighbour.type != interactWith then
5:             continue
6:         end if
7:         v_out += (neighbour.position − boid.position)/10
8:     end for
9: end procedure
```

Regla: Frontera

Algoritmo 6 Boundary

1: **procedure** Boundary($boid, universe$)
2: $x_{min} \leftarrow universe.metadata.boundaries.minx$
3: $x_{max} \leftarrow universe.metadata.boundaries.maxx$
4: $y_{min} \leftarrow universe.metadata.boundaries.miny$
5: $y_{max} \leftarrow universe.metadata.boundaries.maxy$
6: $z_{min} \leftarrow universe.metadata.boundaries.minz$
7: $z_{max} \leftarrow universe.metadata.boundaries.maxz$
8: $speed \leftarrow CONST$
9: $\boldsymbol{v} \leftarrow \{0, 0, 0\}$
10: **if** $boid.x < x_{min}$ **then**
11: $\boldsymbol{v}_x = speed$
12: **end if**
13: **if** $boid.x > x_{max}$ **then**
14: $\boldsymbol{v}_x = -speed$
15: **end if**
16: **if** $boid.y < y_{min}$ **then**
17: $\boldsymbol{v}_y = speed$
18: **end if**
19: **if** $boid.y > y_{max}$ **then**
20: $\boldsymbol{v}_y = -speed$
21: **end if**
22: **if** $boid.z < z_{min}$ **then**
23: $\boldsymbol{v}_z = speed$
24: **end if**
25: **if** $boid.z > z_{max}$ **then**
26: $\boldsymbol{v}_z = -speed$
27: **end if**
28: **return** v
29: **end procedure**

Índice

Implementación

Lenguaje

Kotlin

Visualización

Ovito

Avance de simulación y animación

$\Delta t = \frac{1}{60}$ (s)

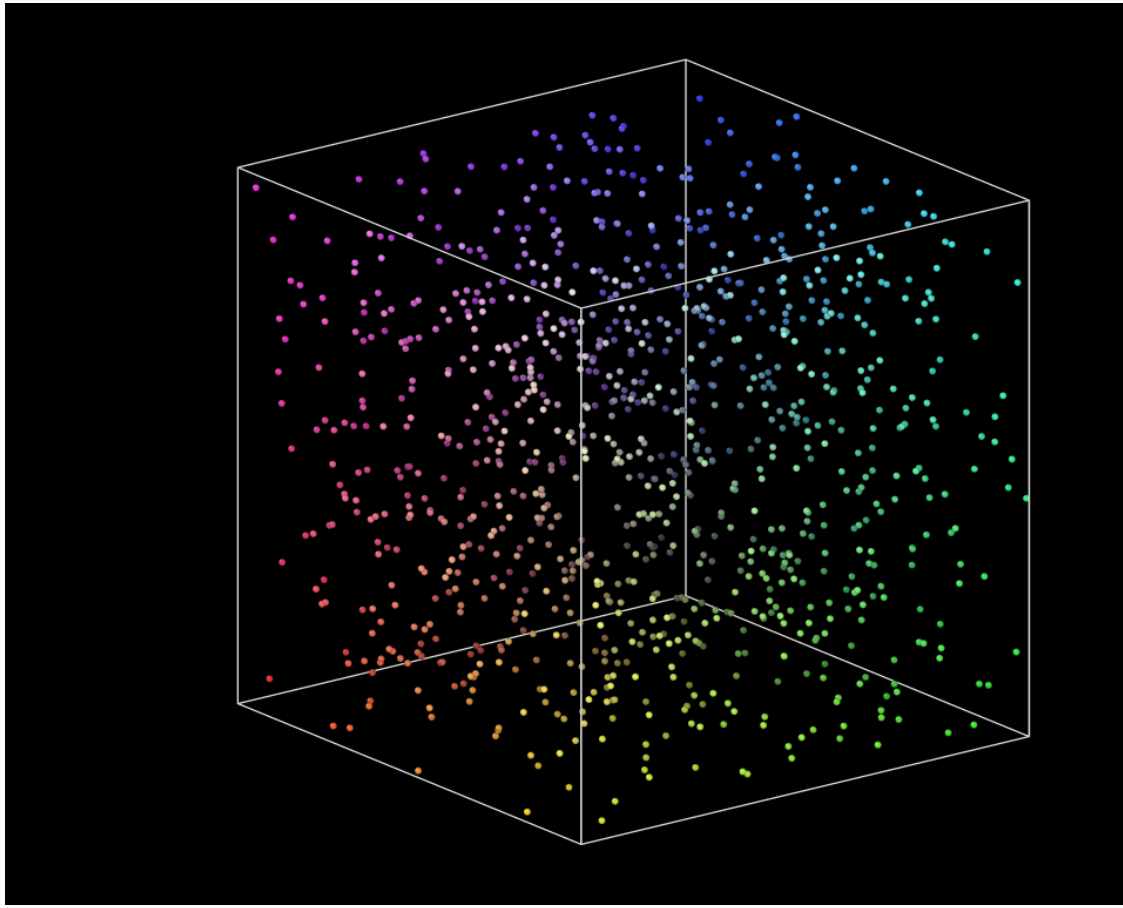
Método de actualización

Beeman modificado (fuerza depende de v_t)

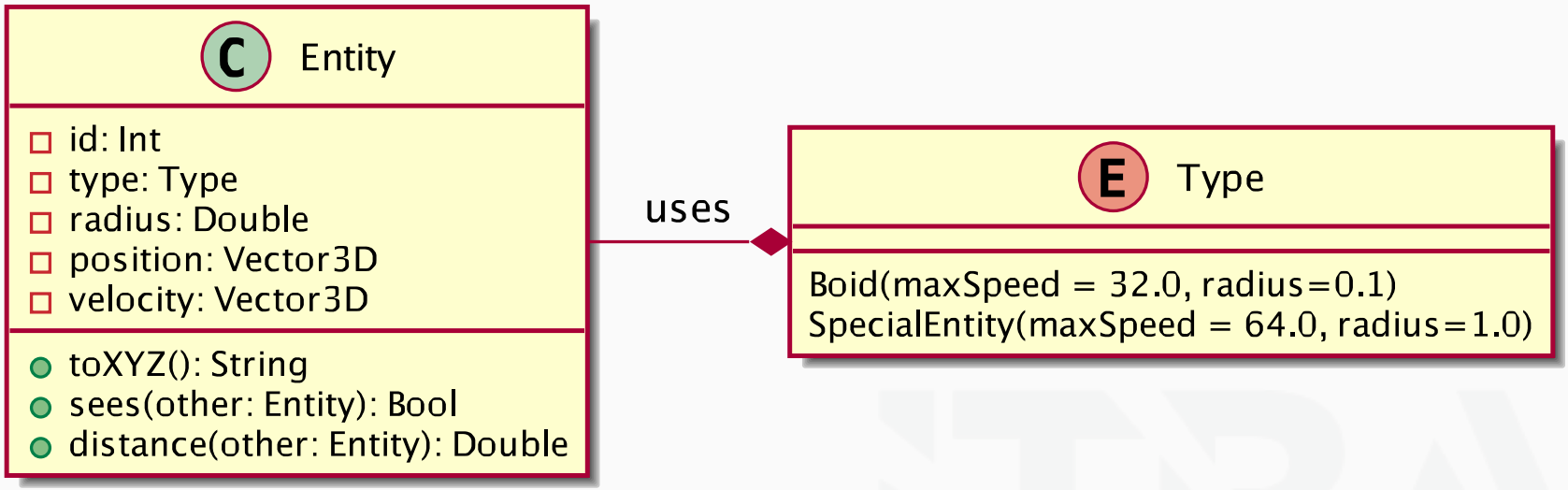
Visualización: Color

Color depende de la posición:

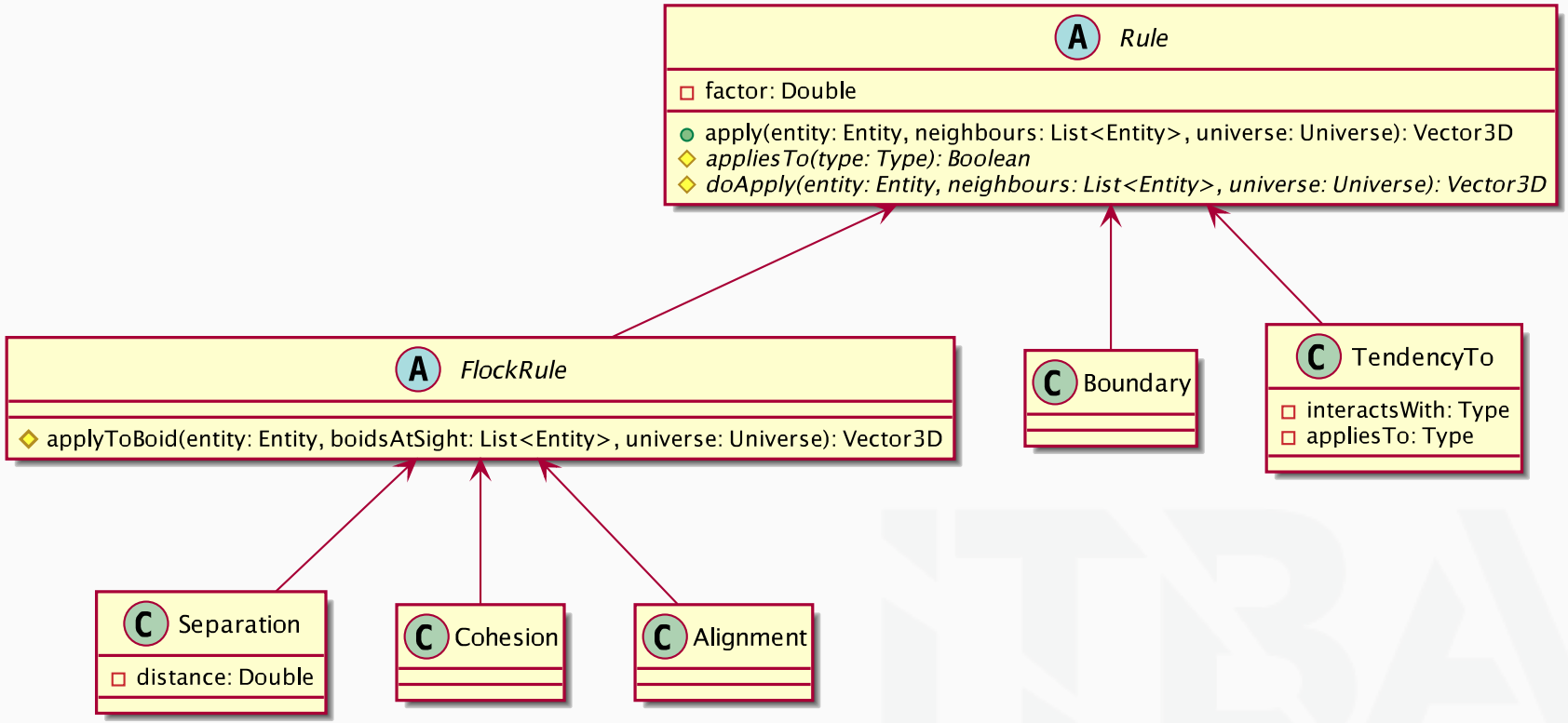
- $\mathbf{R} = 0{,}2 + (\mathit{position}.\mathbf{x}/\mathit{width}) * 0{,}8$
- $\mathbf{G} = 0{,}2 + (\mathit{position}.\mathbf{y}/\mathit{height}) * 0{,}8$
- $\mathbf{B} = 0{,}2 + (\mathit{position}.\mathbf{z}/\mathit{depth}) * 0{,}8$



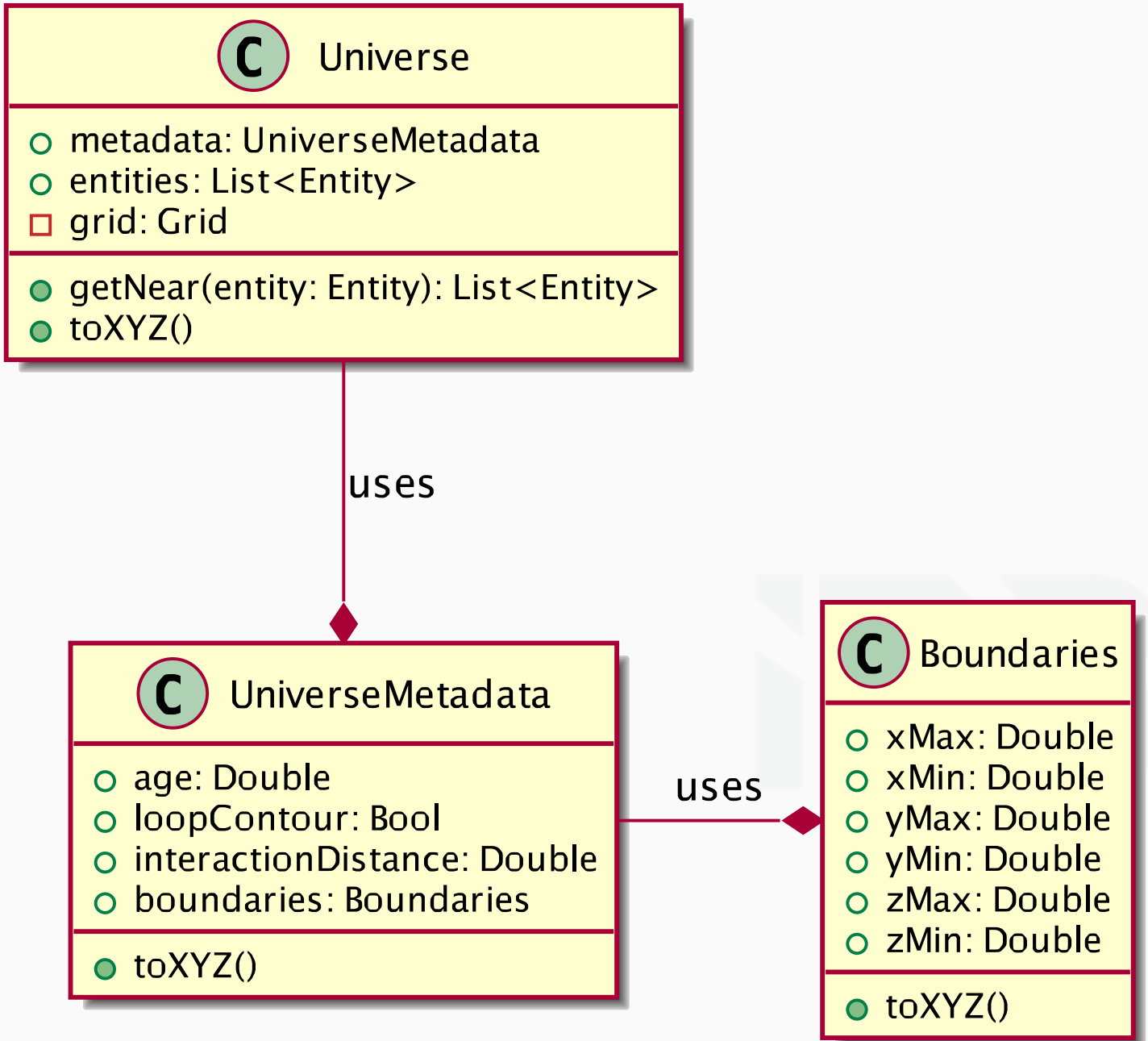
Entidades


C
Entity
id: Int
type: Type
radius: Double
position: Vector3D
velocity: Vector3D
toXYZ(): String
sees(other: Entity): Bool
distance(other: Entity): Double
uses
E
Type
Boid(maxSpeed = 32.0, radius=0.1)
SpecialEntity(maxSpeed = 64.0, radius=1.0)

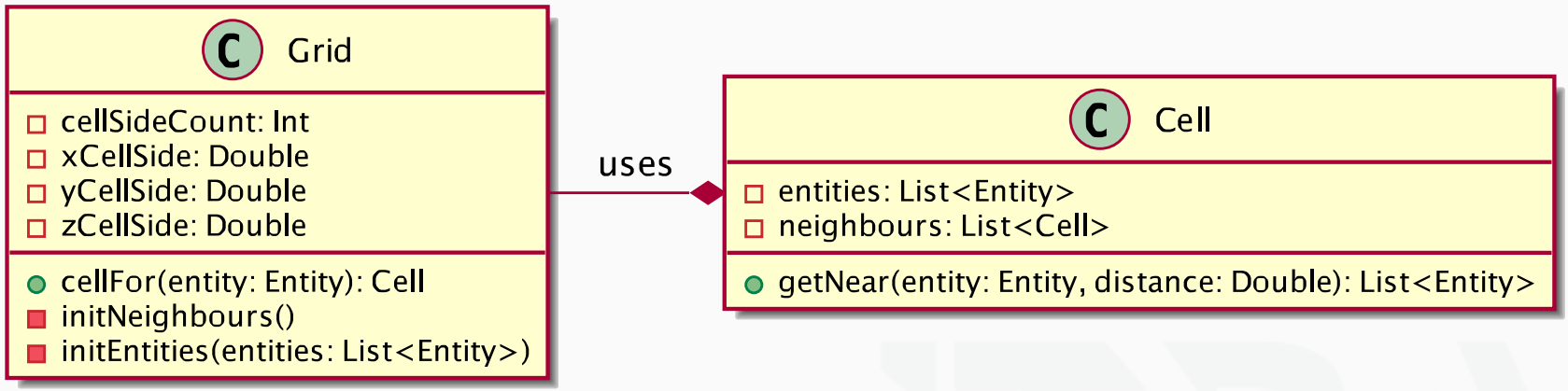
Reglas


Rule
factor: Double
apply(entity: Entity, neighbours: List<Entity>, universe: Universe): Vector3D
appliesTo(type: Type): Boolean
doApply(entity: Entity, neighbours: List<Entity>, universe: Universe): Vector3D
FlockRule
applyToBoid(entity: Entity, boidsAtSight: List<Entity>, universe: Universe): Vector3D
Boundary
TendencyTo
interactsWith: Type
appliesTo: Type
Separation
distance: Double
Cohesion
Alignment

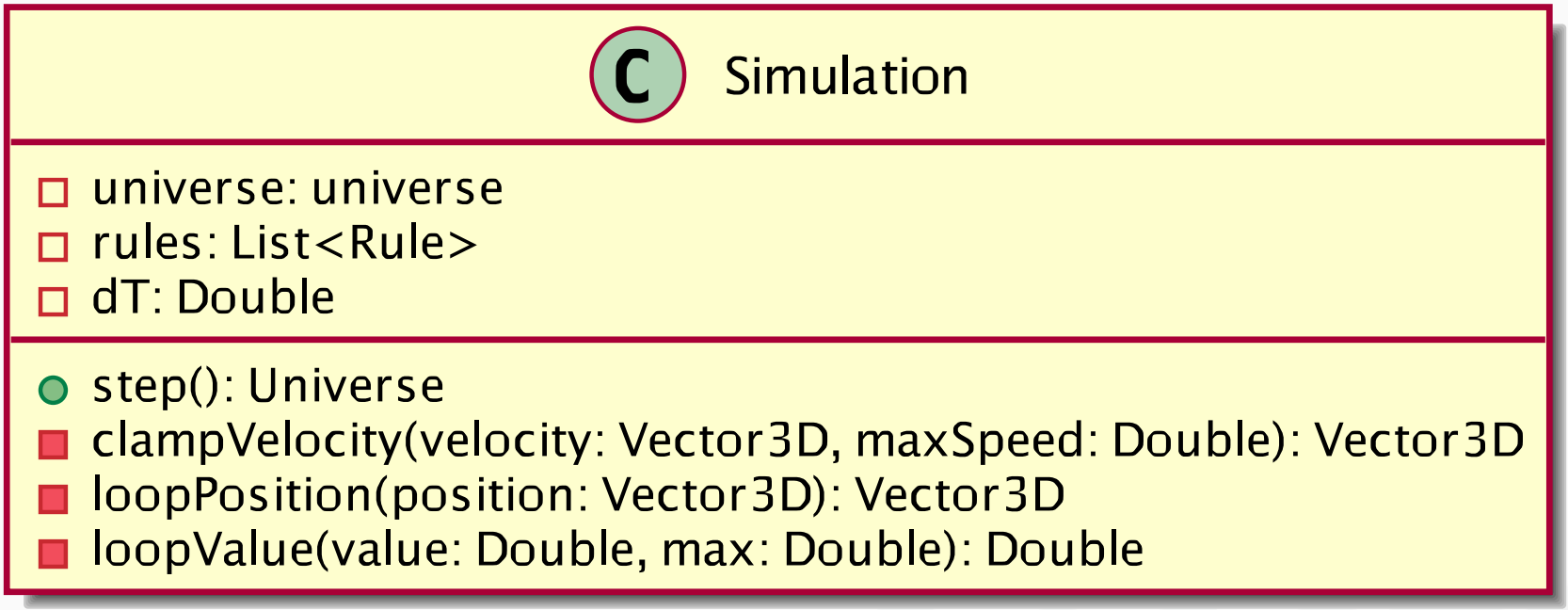
Universo


Universe
metadata: UniverseMetadata
entities: List<Entity>
grid: Grid
getNear(entity: Entity): List<Entity>
toXYZ()
uses
UniverseMetadata
age: Double
loopContour: Bool
interactionDistance: Double
boundaries: Boundaries
toXYZ()
uses
Boundaries
xMax: Double
xMin: Double
yMax: Double
yMin: Double
zMax: Double
zMin: Double
toXYZ()

Grilla


Grid
cellSideCount: Int
xCellSide: Double
yCellSide: Double
zCellSide: Double
cellFor(entity: Entity): Cell
initNeighbours()
initEntities(entities: List<Entity>)
uses
Cell
entities: List<Entity>
neighbours: List<Cell>
getNear(entity: Entity, distance: Double): List<Entity>

Simulación


Simulation
universe: universe
rules: List<Rule>
dT: Double
step(): Universe
clampVelocity(velocity: Vector3D, maxSpeed: Double): Vector3D
loopPosition(position: Vector3D): Vector3D
loopValue(value: Double, max: Double): Double

Índice

Parámetros de simulación

Tamaño del universo
32m x 32m x 32m

Condiciones de contorno
habilitadas

Cantidad de boids
1024

Cantidad de especiales
0

Distancia de interacción
2m

Parámetros de simulación

Radio: boids

0.1m

Radio: especiales

1.0m

Velocidad máxima: boids

32 m/s

Velocidad máxima: especiales

64 m/s

Reglas: Factores

Alineamiento
0.25

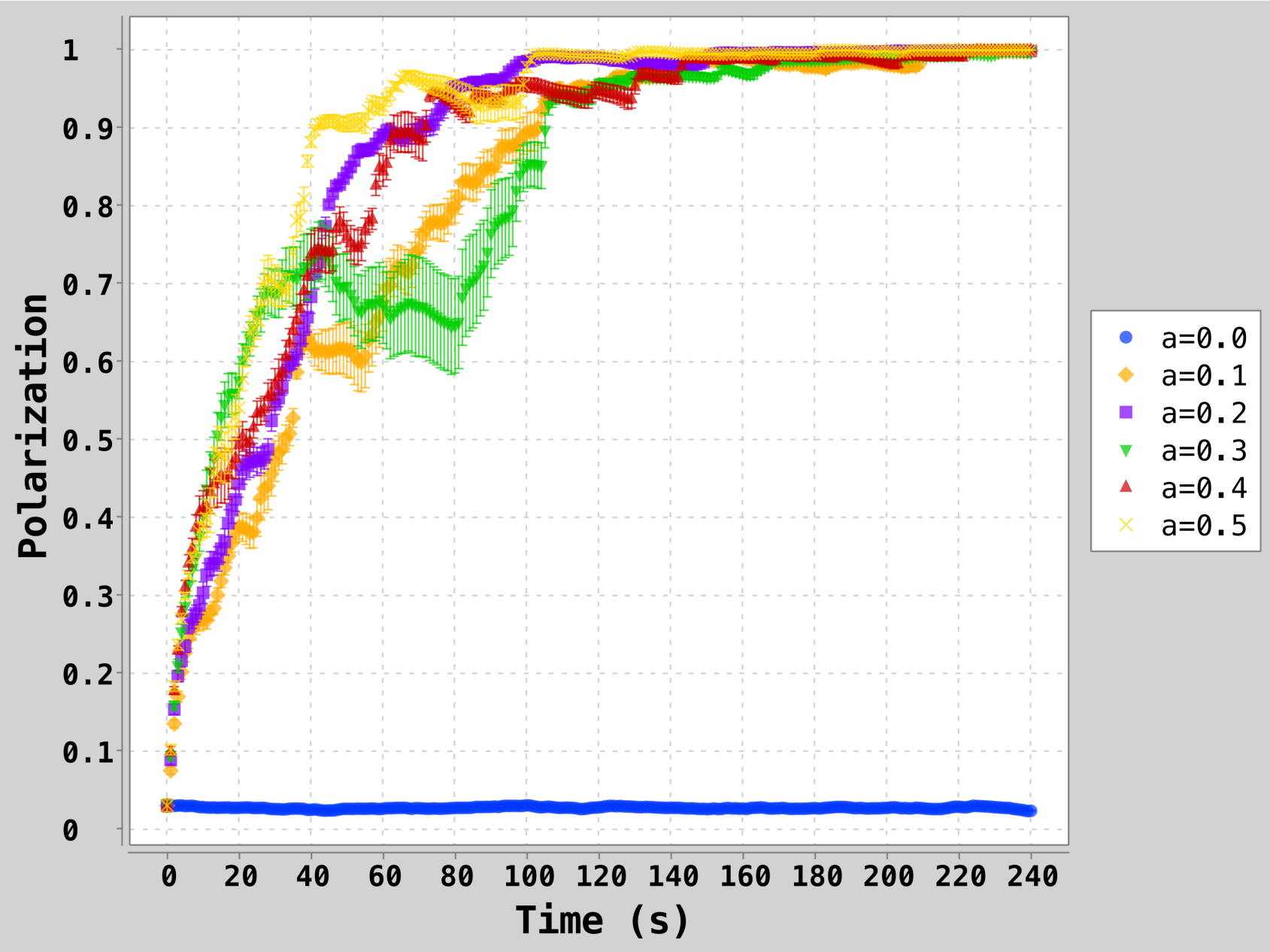
Cohesión
0.25

Separación
0.25

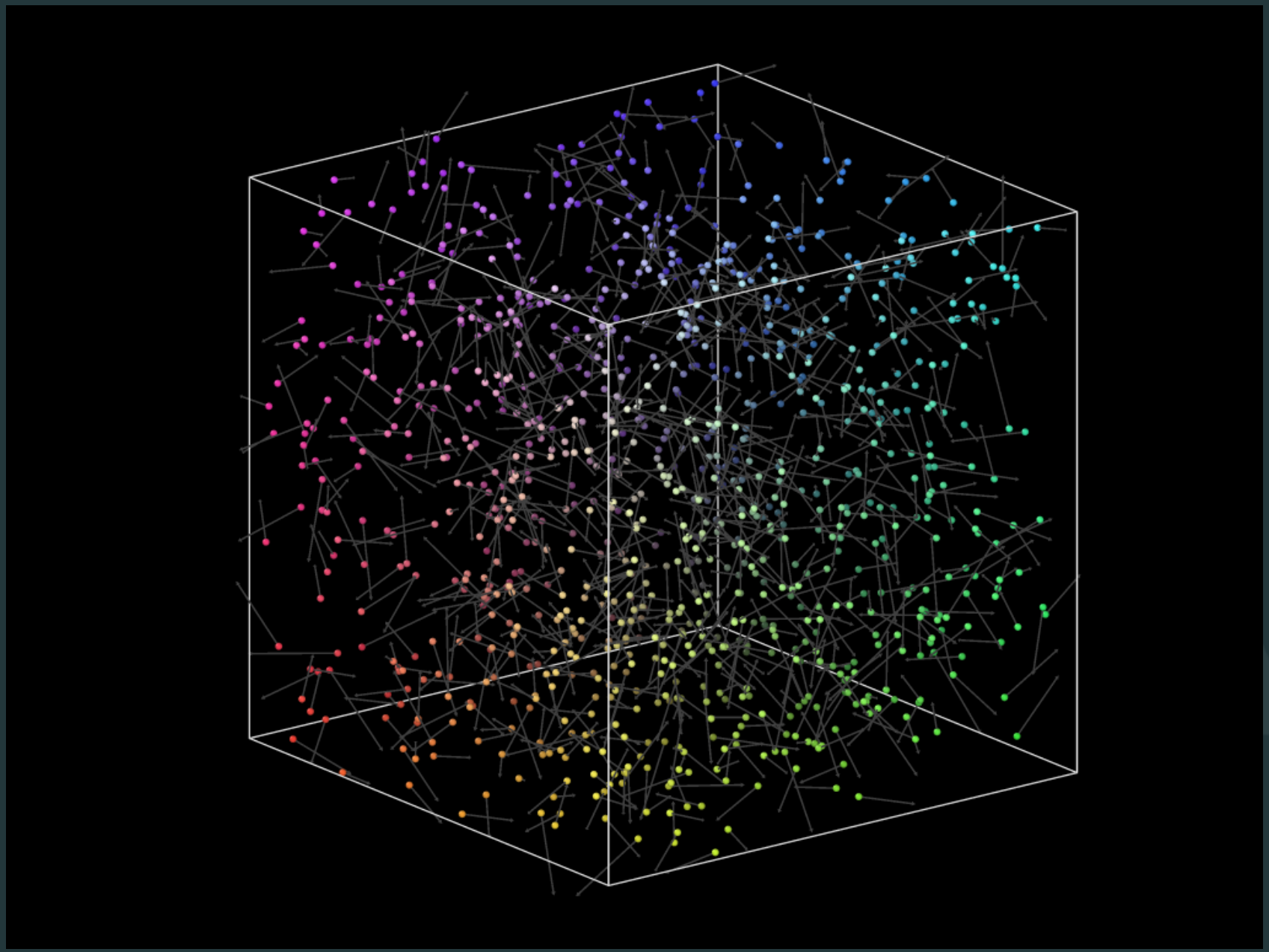
Tendencia: Boid a Especial
-0.25

Tendencia: Especial a Boid
0.25

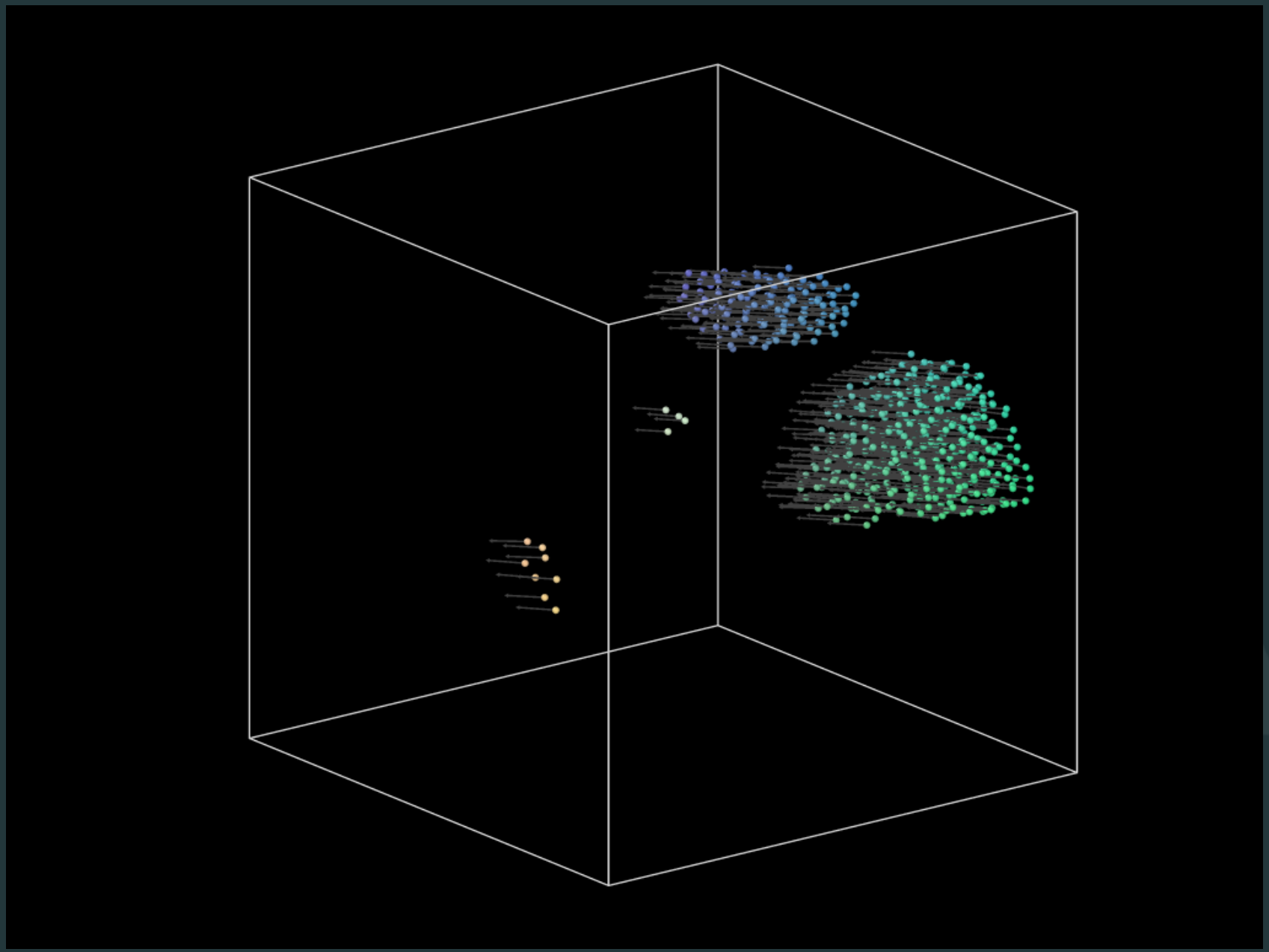
Polarización: Alineamiento



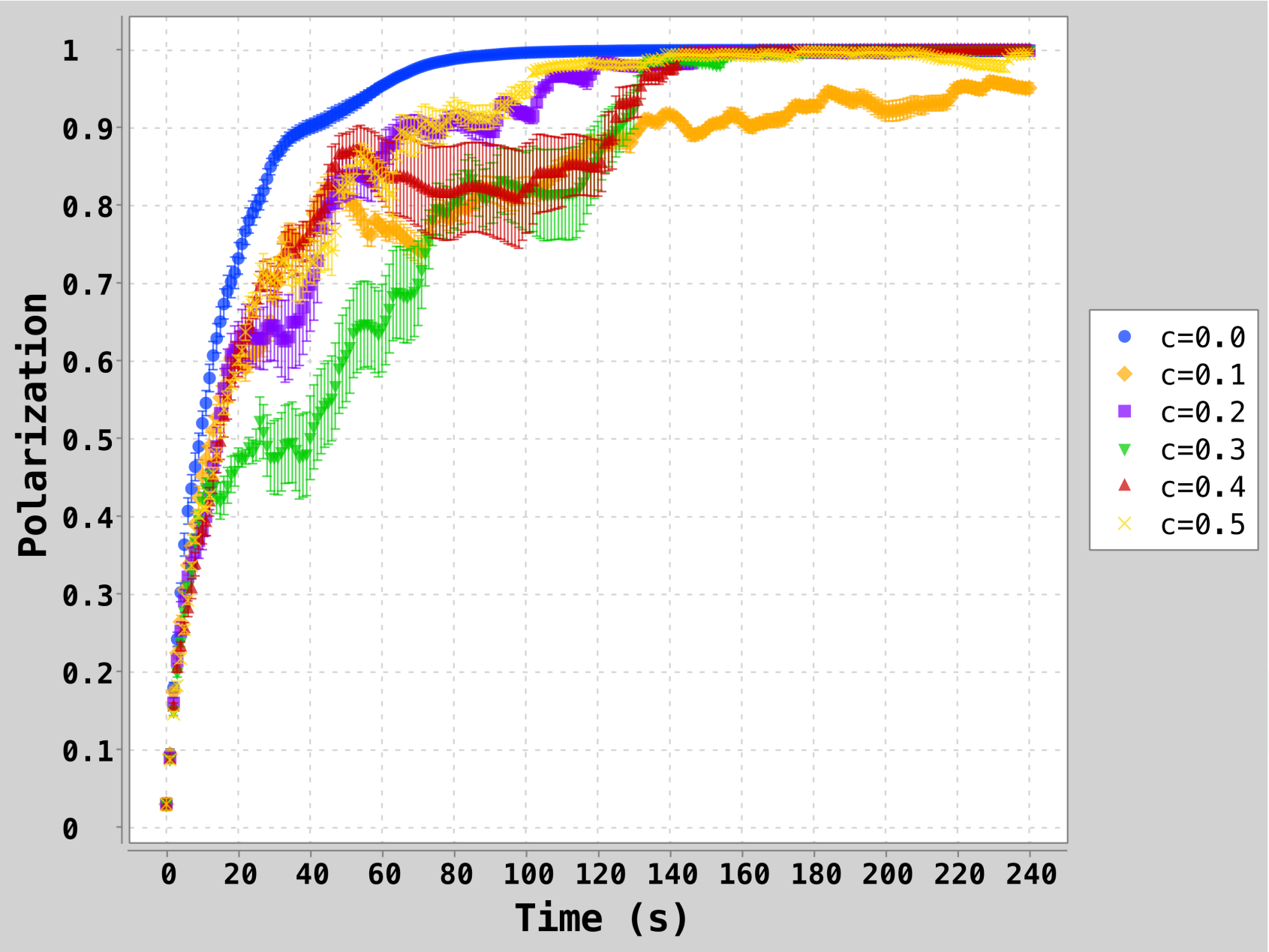
Alineamiento: a = 0.0



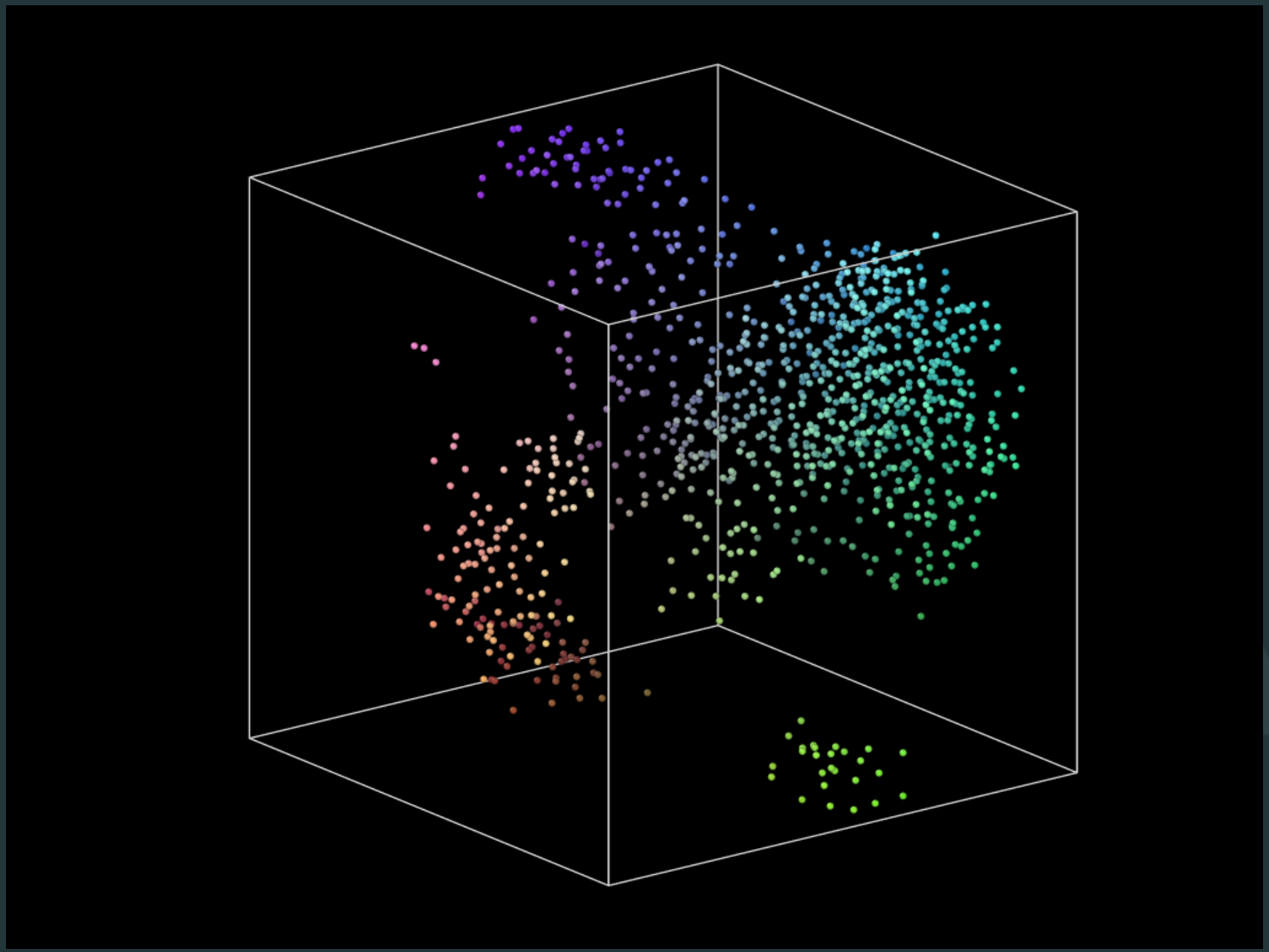
Alineamiento: a = 0.5



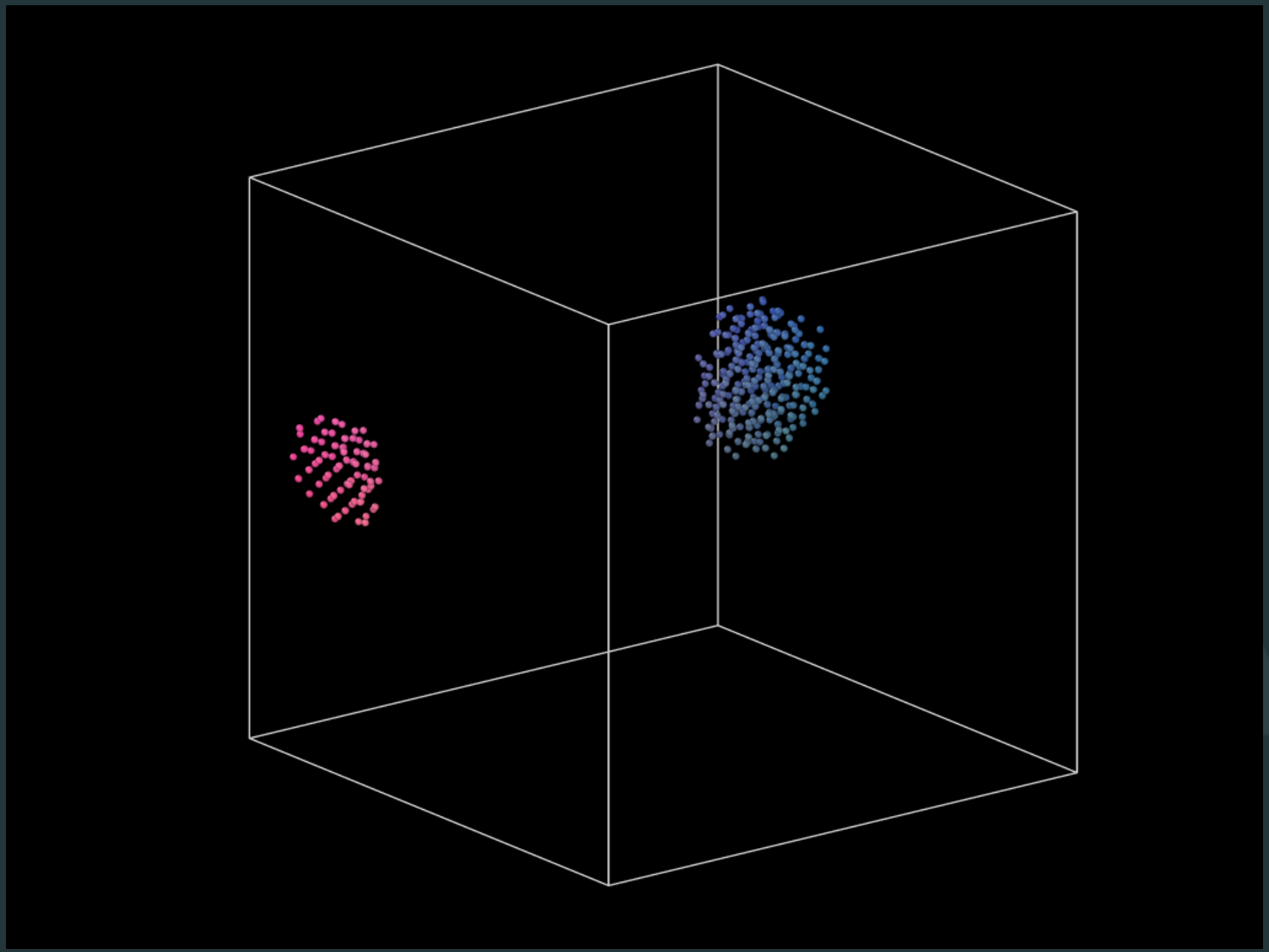
Polarización: Cohesión



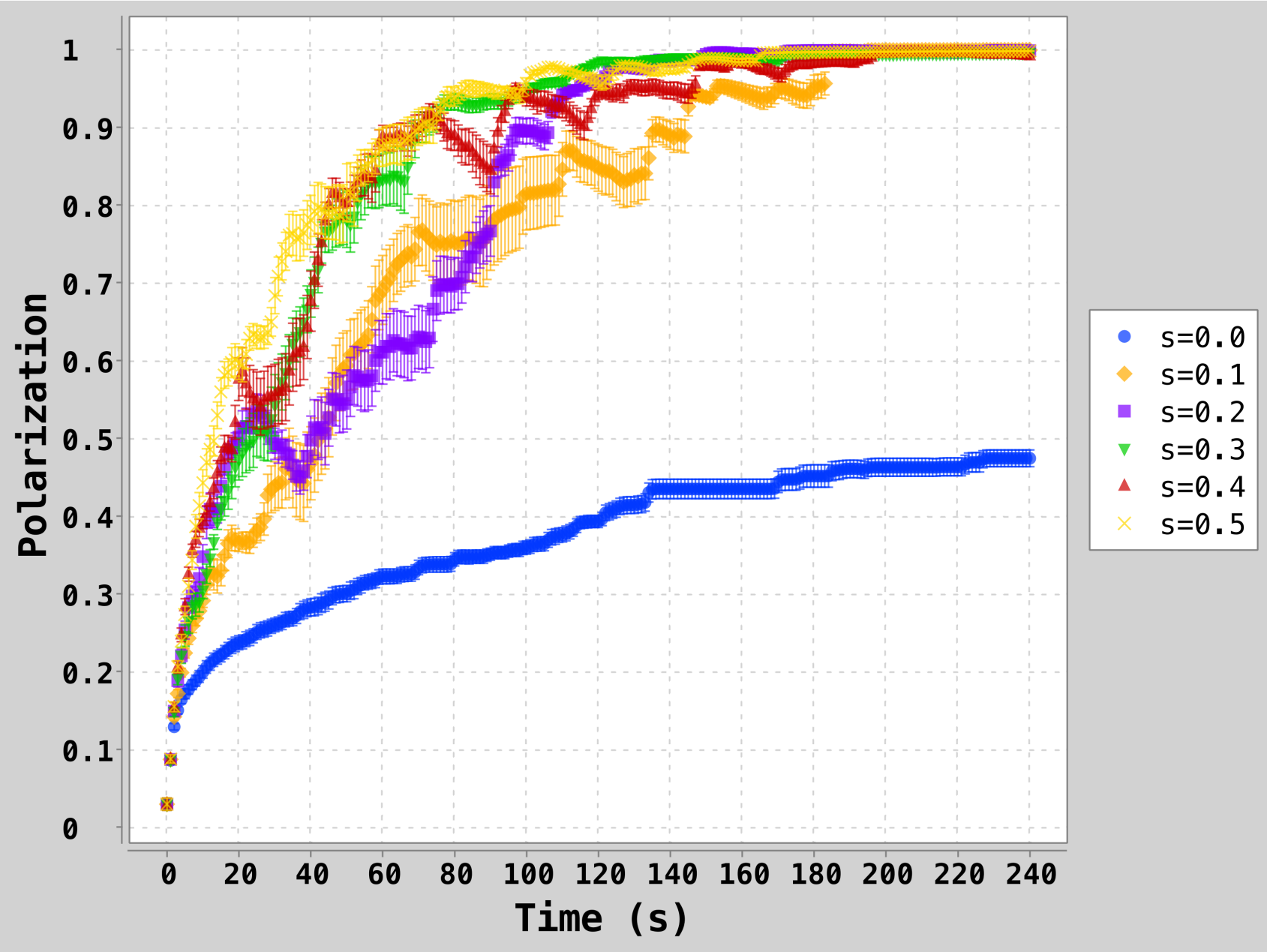
Cohesión: c = 0.0



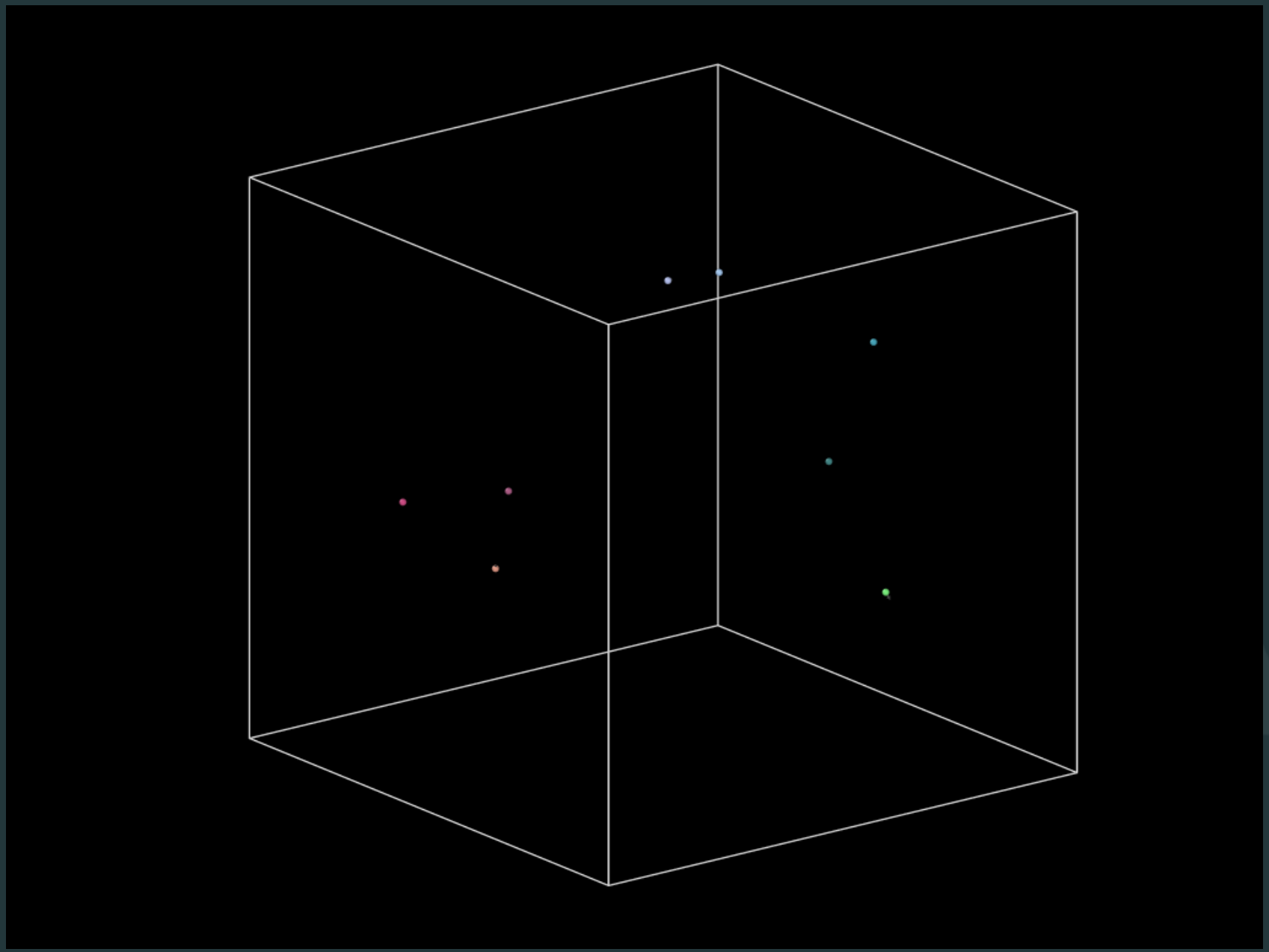
Cohesión: c = 0.5



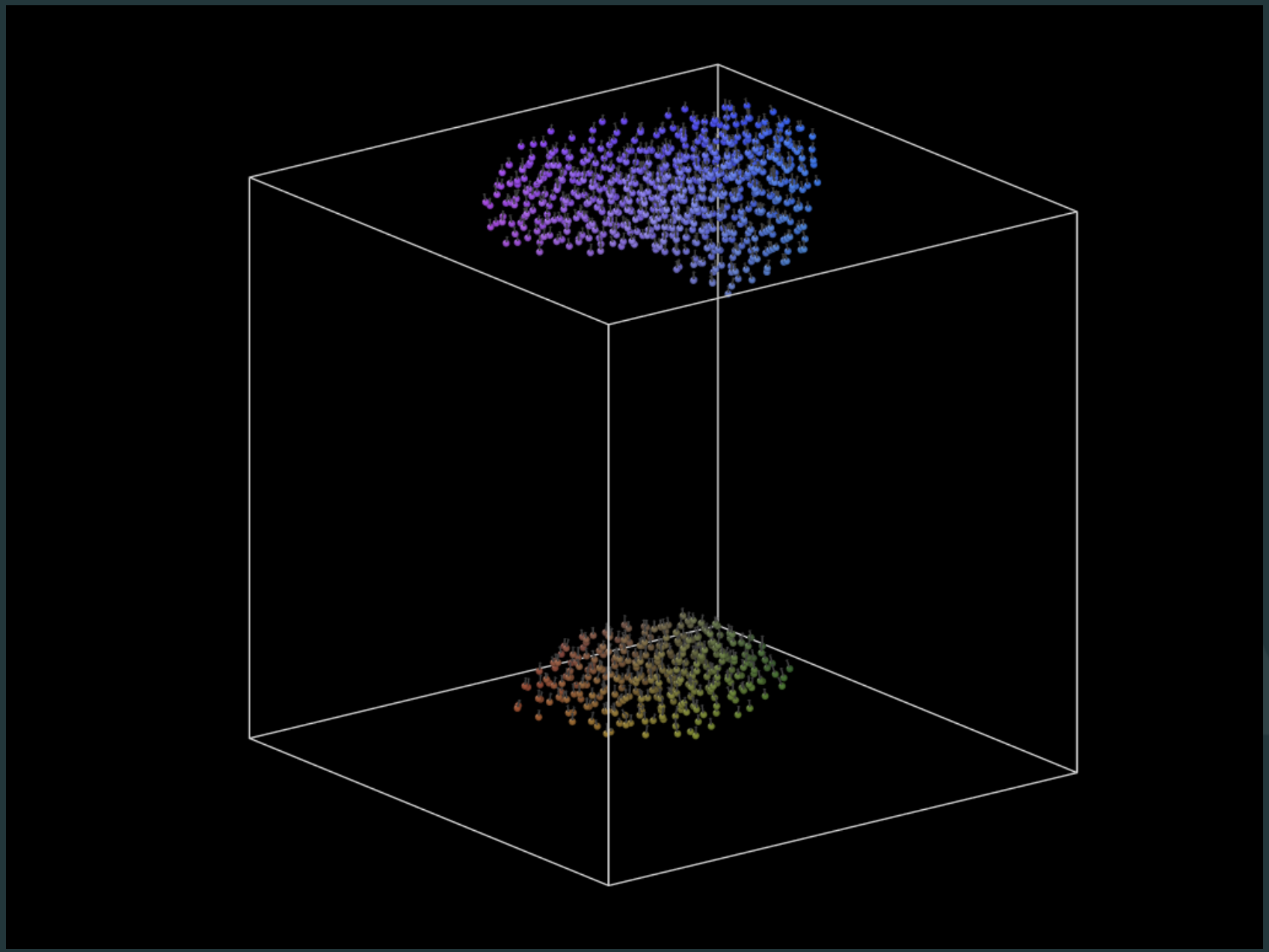
Polarización: Separación



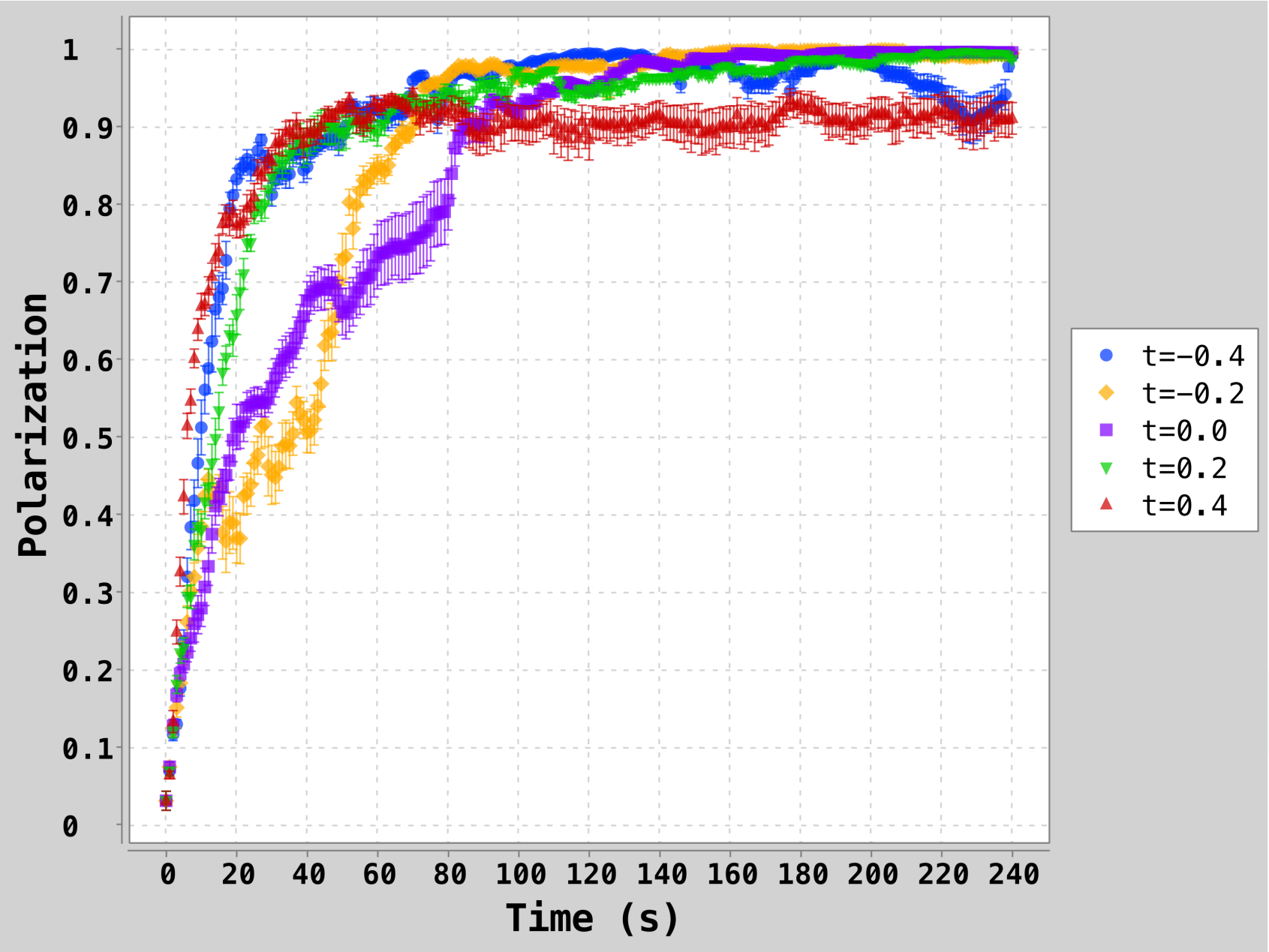
Separación: s = 0.0



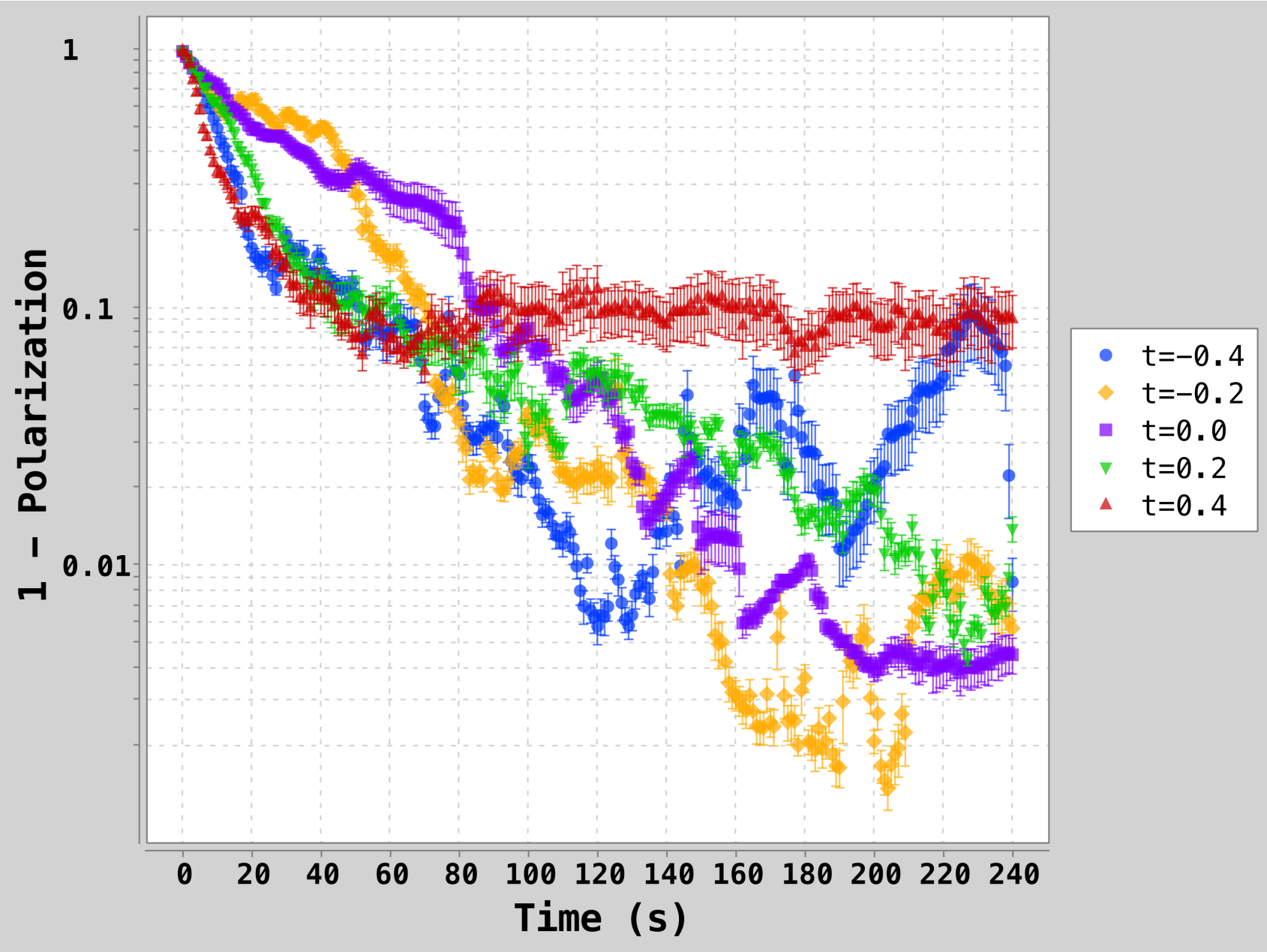
Separación: s = 0.5



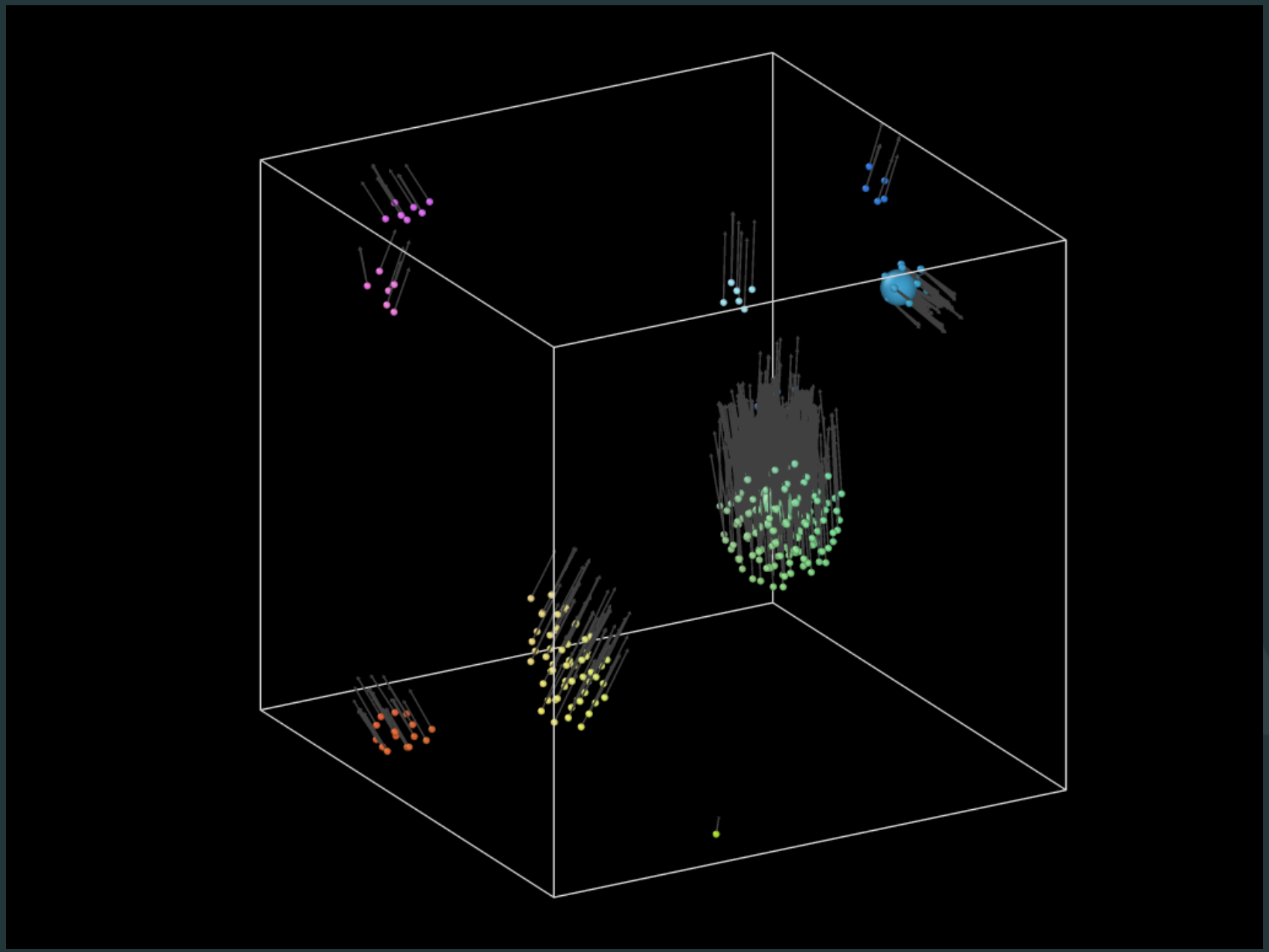
Polarización: Tendencia a



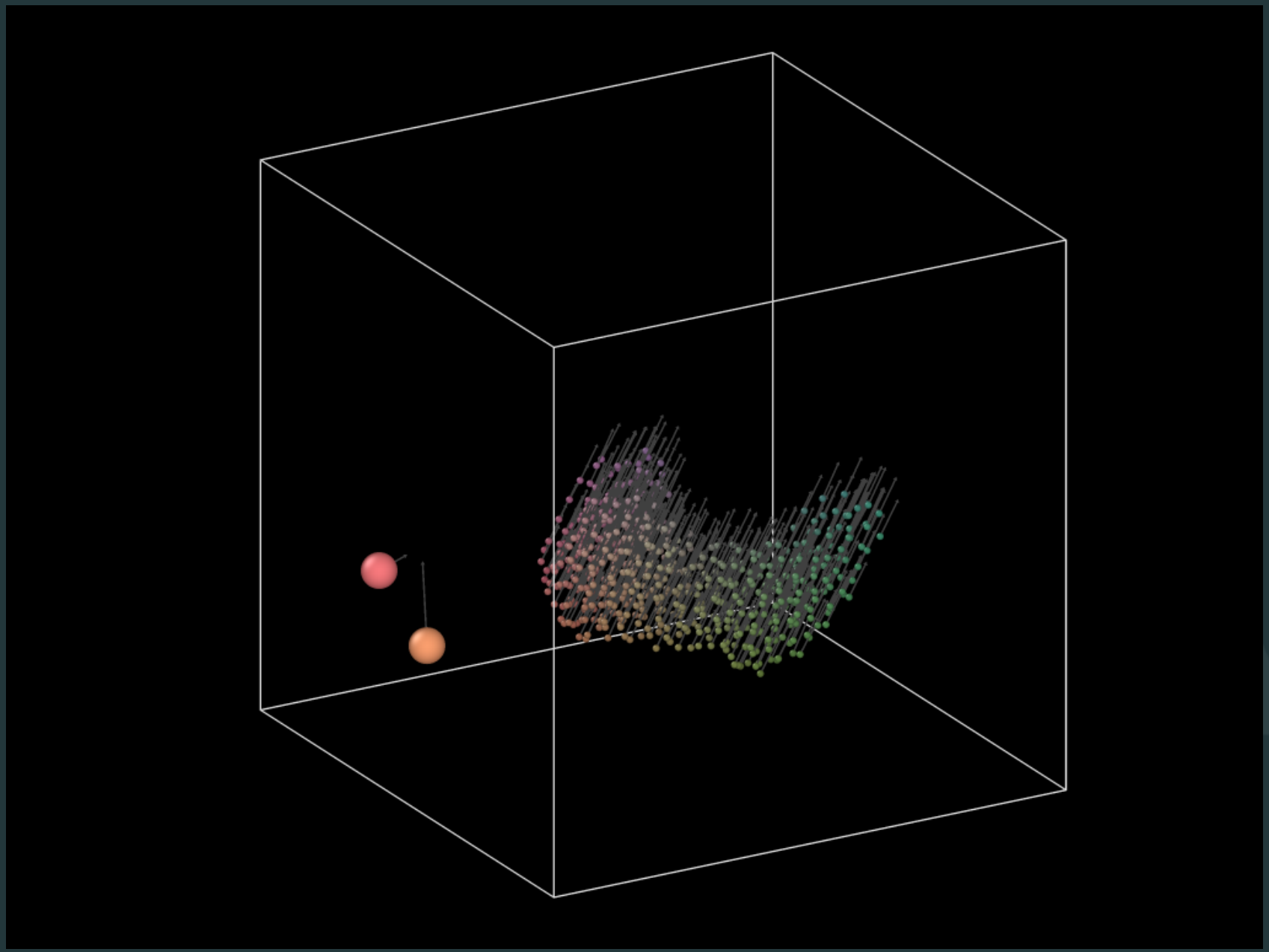
1 - Polarización: Tendencia a



Tendencia a: t = 0.4 (2 boids especiales)



Tendencia a: t = -0.4 (2 boids especiales)



Índice

Conclusiones

- La regla más importante para polarización: **Alineamiento**.
- Desactivar la regla de **Cohesión** acelera la polarización.
- Regla de **Tendencia a** cambia en gran medida el comportamiento de los *boids*.