Issues found:

1. Many lists
2. Instatiate/Destroy
3. Scripting backend is Mono
4. Since we aren’t using physics, objects can be static
5. Reducing draw calls
6. Shadows are soft & not needed for this prototype.
7. Using Find function
8. Audiolistener not needed (no sound in game)
9. Instances of Vector3.magnitude
10. Only do Boids.AvoidCollisions() if Instance.m\_tempHits is larger than 0
11. Checking for what player presses (1~4) is in a for loop.

Attempted changes to issues:

1.Change to Arrays where possible (in some places a List is preferable since we want to Add).

Gave quite an increase in efficiency (up to 15%).

2.Object pooling

Made a new class & gameobject for object pooling instead of Destroy & Instantiate.

Gave quite an increase in efficiency (up to 20%).

3.Change to IL2CPP

Gave a slight increase in efficiency.

4.Changed to Static

Gave a slight increase in efficiency.

5.Change to dynamic batching

Dynamic Batching seems to give more efficiency here according to the profiler, I also tried GPU Instancing.

6.Shadows aren’t really needed here, change to no shadows on lightsource.

Gave a slight increase in efficiency.

7.Cache instead in classes.

Gave quite an increase in efficiency (up to 15%).

8.Removed Audiolistener.

Gave a tiny increase in efficiency.

9. Changed to .sqrMagnitude.

Gave a tiny increase in efficiency.

10. if case to check if necessary to avoid collisions.

Gave a tiny increase in efficiency.

11.Changed to an if case.

Tiny improvement.

12.Attempted to call the GC on reset & when changing camera angle, marginal difference.

All points above together gave an overall increase in fps of 55%.

Future suggestions:

Implement DOTS

Rework movement

Use physics for collisions

Camera summary:

Decided to go with a couple of fixed camera angles the player can switch between.

There are 4 points and the player shifts perspective by pressing LeftArrow or RightArrow.

This sets the camera's position and rotation to the next/previous point in the array.

These 4 points are opposite sides and above and below the Boids.

If we had better performance I would have gone with a camera you can move, but when tried performance dropped noticeably. This is more hidden when camera change is (almost) instant.