Boids VFX · Flocking, Swarm Navigation & Behaviour · URP

• • •

Support & Feedback

Email: care@skulmovski.studio • Discord: [Join our community]

If you encounter any issues, please contact me via Discord or Email before submitting a review.

• • •

Table of Contents

⚠️ IMPORTANT: Known Constraints & By-Design Behavior 2
⚠️ IMPORTANT: First Time Setup (Read Before Starting) 6
Project Requirements 6
Fast Integration (Quick Start / Demo Scene) 7
Using Setup Assistant 8
Setting Up Boids VFX in a Custom Scene 9
⚠️ How the Components Fit Together 9
Swarm Control Center 11
Playback, Population & Spawning 13
Flocking, Motion & Presets 15
Volumes, Boundaries & Follow 17
Spline Navigation 19
World, Obstacles & Influence Zones 20
Death Zones, Triggers & Events 23
Visual Effects & Custom Mesh Preparation 26
Secondary VFX 28
Material Colours to Texture Baker 31
Runtime API 33
Performance & Troubleshooting 39
Swarm System — Parameter Reference 44
World & Components — Parameter Reference 49
Fish VFX — Parameter Reference 54
Bird / Wing VFX — Parameter Reference 55
Non-Deforming Mesh VFX — Parameter Reference 57
Quad VFX — Parameter Reference 58
Thank You! 60

⚠️ IMPORTANT: Known Constraints & By-Design Behavior

Before using the asset, please review the following simulation, rendering and playback behaviours.

Some follow from the GPU architecture; others define how the supplied effects and tools work. They may affect mesh preparation, scene setup and gameplay integration.

Detailed instructions and the relevant settings are covered in the following chapters.

Boids Are Not Individual GameObjects

Boid movement is calculated on the GPU and rendered through Unity VFX Graph. Individual boids do not have their own GameObject, Transform, Animator or Rigidbody.

The swarm is configured through SwarmSystem, habitat volumes and environment zones. Trigger events provide aggregate boid statistics rather than references to individual scene objects.

Navigation Uses Steering Forces and Local Avoidance

The system combines flocking, wandering, habitat volumes, splines and local environment influences.

Obstacle avoidance does not perform global pathfinding through NavMesh or A*. Complex mazes, narrow passages and overlapping obstacles may require additional scene setup. Test these areas with your chosen speeds, agent sizes and avoidance settings.

Obstacles Use Primitive Shapes

SwarmObstacleZone represents obstacles using mathematical primitives. A supported Unity Collider can supply the shape, but boid interactions are not calculated through standard Rigidbody physics.

Adding the component to a building does not automatically reproduce its visible geometry, windows, doors or small details. Represent complex objects with a suitable primitive or several zones.

Avoidance uses the boid's position and configured Agent Radius. The visible mesh does not automatically become an exact collision shape.

Dead Boids Do Not Automatically Respawn

Capacity defines the maximum number of boids that can spawn during one simulation cycle. It is not a continuously maintained living population.

Once a boid dies, its slot is not reused. To create a new population, call Reinitialize or use Stop → Play. Changing Capacity also recreates the population.

Main VFX Particles Have a Technical Lifetime

In the supplied graphs, the main visual particles have a technical lifetime of **9999 seconds of VFX time — 2 h, 46 m and 39 s**.

This is separate from the GPU Lifetime setting in SwarmSystem. It starts when the visual particles are created during graph initialisation and includes time spent waiting for delayed boid spawning.

For finite effects, call Stop when they are no longer needed. To run beyond this limit, start a new cycle through Reinitialize or Stop → Play. This discards the existing population.

When the technical lifetime expires, rendering may end while the GPU simulation continues. IsPlaying does not automatically become false. Calling Play again on an already running system does not renew this lifetime.

Population Limit and Performance

One **SwarmSystem supports up to 1000 boids**. This limit applies to each system separately.

In the current implementation, each living boid checks the other living boids in its system to calculate neighbour interactions. Simulation cost therefore rises quickly as the population increases. Performance also depends on the number and complexity of zones, mesh geometry, animation settings and secondary effects.

You can use multiple SwarmSystems, but their workloads add up. The maximum supported population does not guarantee the same performance on every device. Test the final configuration on your target platform.

Mesh Animation Is Designed for Ambient Use

The supplied shader animations are general-purpose deformations intended to bring ambient swarms to life. They do not automatically produce convincing, species-specific motion for every fish or winged model.

The result depends on mesh shape, vertex placement, the animation mask and deformation settings. These animations are best suited to ambient creatures rather than close-up shots that require detailed, carefully authored motion.

For hero creatures or cinematic close-ups, consider individually animated models with a flocking or movement system designed to support that workflow.

Stop Ends the Effect Immediately

Stop immediately removes the main population and secondary effects. It does not wait for secondary particles to finish and does not generate On Die bursts for the stopped boids.

Pausing while preserving the simulation state is not supported. Starting again after Stop creates a new population.

Reinitialize also recreates the population and resets related state, including trigger history and attraction-zone selection. Calling Reinitialize on a stopped system does not start it.

Animated Meshes Require Preparation

Fish body bending and wing flapping deform mesh vertices in the shader. The result depends on the model's shape, vertex placement, mesh orientation and animation mask.

The animation mask uses the **red vertex colour channel — Vertex Color.R**. A value of 0 excludes the vertex from the mask's influence, 1 applies its full influence, and intermediate values provide a gradual transition.

Prepare the mask before using the model for predictable results. Without suitable vertex colour data, parts that should remain still may deform along with the tail or wings.

Mesh orientation, mask preparation and export are covered in the custom mesh chapter.

The Supplied Animations Use Specific Motion Types

Fish Body Bending produces lateral body deformation. It suits fish models designed for this motion, but does not automatically reproduce the vertical tail movement of dolphins or orcas.

Wing Flapping produces procedural wing motion. Suitable models can represent birds or insects, but the result depends on their geometry, mask and deformation settings.

The supplied graphs do not include an animation state system with transitions between flight, landing, takeoff or other actions. A separate VFX option is provided for meshes that do not need deformation.

Temporary Volume Exit Depends on the Balance of Forces

With Allow Temporary Exit enabled, boids can leave their habitat volume, but a return force acts on them outside its boundary.

How far they travel and when they return depend on the combined boundary, attraction, speed and other force settings. There is no guaranteed pursuit distance or delay before returning.

Soft Margin does not define how far boids are allowed to travel outside the Volume. It contributes to the transition into boundary steering and affects the return behaviour when temporary exit is enabled.

Attract and Repel are also steering influences. Repel, in particular, does not replace a solid obstacle.

Reaching the End of an Open Spline Does Not End the Simulation

Reaching the end of an open spline does not automatically remove boids or stop the effect. Boids may continue moving beyond the route.

If the sequence needs an ending, configure it separately. For example, place a Death Zone at the exit, use boid lifetime, or switch the system to Volume mode through an event handler.

Trigger Events Are Asynchronous

Trigger crossings and statistics are calculated on the GPU. Results reach the CPU through asynchronous readback at the configured polling rate, so an event may arrive after the crossing occurs.

One SwarmSystem supports **up to three unique trigger zones**. Each zone can belong to only one system at a time.

A trigger responds when its configured threshold is reached. It is not an individual OnTriggerEnter callback for every boid. Repeat activation depends on the count mode, Trigger Once and rearming through Rearm.

Boids May Become Trapped in Tight or Conflicting Geometry

Narrow gaps, overlapping obstacles and obstacles placed close to a Volume boundary can leave too little room for boids to turn or escape.

Flocking, boundary return, influence forces and obstacle avoidance may pull in competing directions. In these situations, boids can bunch up, turn repeatedly or appear pressed against a surface.

Local avoidance does not guarantee recovery from every trapped configuration. Leave sufficient clearance for the configured Agent Radius and Wall Margin, and check that the habitat provides space around obstacles. If needed, adjust the geometry, reduce crowding or rebalance the movement forces.

Secondary VFX Have Fixed Roles

The supplied graphs contain two specialised branches:

- **Continuous** emits secondary particles while a boid is alive.
- **On Die** creates secondary particles when a boid dies.

Both enables the two branches together. Each branch has its own visual settings, but the emission mode does not reassign their roles.

Secondary particles do not participate in flocking or automatically use the main simulation's obstacle avoidance.

After a boid dies normally, its secondary particles can remain until their own lifetime ends. Calling Stop removes both branches immediately.



⚠️ IMPORTANT: First Time Setup (Read Before Starting)

If you are importing this asset into your project for the first time, we **highly recommend** opening the Setup Assistant before running the demo scene or adding prefabs to your custom scenes.

If the Assistant window did not open automatically after import, you can launch it manually via the top menu:

Tools > Skulmovski Studio > Boids VFX · Flocking, Swarm Navigation & Behaviour · URP > VFX Setup Assistant

Project Requirements

Ensure your project meets the following requirements before using the asset.

- **Unity Version:** Unity 6+. Tested with **Unity 6000.0.77f1**.
- **Universal Render Pipeline:** This package is intended for URP projects.
- **Visual Effect Graph:** Must be installed through the Unity Package Manager.
- **Compute Shader Support:** The boid simulation uses GPU Compute Shaders. Your target platform and graphics API must support them.

The Setup Assistant checks the active Render Pipeline and whether the Visual Effect Graph package is installed.

First-Time Compilation

When importing the asset for the first time, the boid compute shader and VFX Graphs may take some time to compile. Wait for importing and compilation to finish before testing the demo scene.

If the effect renders incorrectly after compilation has finished, use **Recompile VFX Graphs** in the Setup Assistant and wait for the reimport to complete.

Fast Integration (Quick Start / Demo Scene)

To get the most out of the Demo Scene without encountering any rendering errors, please follow this exact sequence:

Step 1: Check the Project with Setup Assistant

The **Setup Assistant** window should open automatically after importing the asset. If it does not, open it via:

`Tools > Skulmovski Studio > Boids VFX · Flocking, Swarm Navigation & Behaviour · URP > VFX Setup Assistant`

Click **Refresh** and review **Project-Wide Settings**. Ensure the project uses URP and the Visual Effect Graph package is installed.

Wait for importing and compilation to finish.

Step 2: Recompile VFX Graphs and Open the Demo

Under **Useful Tools** in the Assistant, click **Recompile VFX Graphs** and wait for recompilation to finish.

Then click **Open Demo Scene**.

Step 3: View the Demo

For convenience, the simulation also runs in **Edit Mode**, so entering Play Mode is not required to view the demo.

The demo camera has no movement script, so free-flight controls are not available in **Game View**. Use **Scene View** to explore freely.

Using Setup Assistant

The **VFX Setup Assistant** checks the basic project requirements and provides quick access to the demo scene and editor tools.

Open it via:

```
Tools > Skulmovski Studio > Boids VFX · Flocking, Swarm Navigation & Behaviour · URP > VFX Setup Assistant
```

After making changes to the project configuration, click **Refresh** to update the checklist.

Project-Wide Settings

These checks apply to the project rather than the currently open scene.

- **Active Pipeline is URP:** Verifies that the active Render Pipeline is URP. Pipeline configuration is handled manually.
- **Visual Effect Graph Package Installed:** Checks whether the required package is installed. **Open Package Manager** provides access to package installation.

Scene Settings

This asset does not require any additional scene-specific configuration through the Setup Assistant.

Useful Tools

The Assistant provides quick access to:

- **Recompile VFX Graphs:** Forces recompilation of the package's VFX Graph assets. Use this if freshly imported effects do not render correctly, and wait for compilation to finish.
- **Open Demo Scene:** Opens the included demo scene.
- **Open Control Center:** Opens Swarm Control Center to manage swarm setup, movement, space, population and triggers.
- **Open Baker:** Opens Material Colors to Texture, which combines solid material colours into a texture and creates a single-submesh mesh copy using the model's existing UV0.

Additional Notes

The Setup Assistant covers the **essential project checks** and provides access to the asset's tools, but it is not intended to manage every possible custom project architecture.



Setting Up Boids VFX in a Custom Scene

For a basic effect, add **SwarmSystem** to the scene and link it to one of the supplied VFX Graphs.

Setup Steps

1. Create an empty GameObject and add the **SwarmSystem** component.
2. Select a **Swarm Preset** in the component and click **Apply**.
3. Drag the desired VFX Graph into the scene. You can also parent the resulting VFX object to the system.
4. Assign its **Visual Effect** component to the corresponding field in SwarmSystem.

Boids should now be moving with the selected preset's settings. Entering Play Mode is not required.

The supplied graphs use preconfigured meshes or quads. If the selected mesh effect has no model assigned, assign one to **Source Mesh**. For a custom mesh, enable **Read/Write** in the model's import settings.

If the changes are not reflected in the effect, click **Open in Swarm Control Center** in the system's Inspector, then **Refresh Swarm** in the top-right corner of the window.

Further configuration of movement, habitat volumes, environment interactions and appearance is covered in the corresponding chapters.

⚠️ How the Components Fit Together

SwarmSystem connects the simulation, visual effect and movement space. Additional components control spawning, environment interactions and events.

A basic effect requires only **SwarmSystem and a compatible Visual Effect**. Add other components as needed.

SwarmSystem — Simulation Control

SwarmSystem stores population and movement settings, runs the GPU simulation and passes the result to the linked Visual Effect.

Each system calculates its own population. Boids from different SwarmSystems do not share flocking calculations or treat one another as neighbours.

Visual Effect — Rendering

The **Visual Effect** component uses a compatible VFX Graph to render the boids.

Appearance, textures, size, shader animation and the visual settings of secondary particles are configured here. Movement and population are configured in SwarmSystem.

Assign the Visual Effect to the corresponding field in the system. Placing a VFX object near the controller does not connect them.

Volume and Spline — Movement Space

The system uses one of two spatial modes:

- **Volume** defines a habitat where boids move freely. It can use a single shape or several combined primitives.
- **Spline** defines a route with a movement corridor. Steering forces guide boids along it.

In Spline mode, the regular Volume boundary is not applied alongside the route. These are separate spatial modes, rather than two required parts of the same setup.

Spawn Source — Spawn Location

The **Spawn Source** determines where boids appear and their initial direction.

It does not keep the swarm nearby or replace Volume or Spline. For example, boids can spawn inside a small sphere and then move throughout a much larger habitat.

Without an active Spawn Source, the system uses the current Volume or valid Spline to place the population.

Swarm World — Obstacles and Influences

Swarm World holds the lists of obstacles and influence zones available to the system.

For a zone to affect the swarm:

1. Assign the World to SwarmSystem.
2. Add the zone to the corresponding list in that World.
3. Ensure both the World and the zone are enabled.

Placing an Obstacle or Influence in the scene alone is not enough. Parenting it under the system does not replace registration in the World.

The creation tools in the Environment category create and connect these objects through Control Center.

Multiple SwarmSystems can share one World. They use the same obstacles and influences while keeping separate populations.

Obstacles — Avoidance

SwarmObstacleZone represents an obstacle with a primitive shape. Boids use this shape to calculate avoidance and limit penetration.

The shape is configured separately from the visible geometry. Complex objects may require several primitives.

Influences — Movement Forces

SwarmInfluenceZone supports three types of influence:

- **Attract** draws interested boids towards the zone.
- **Repel** pushes boids away within its working range.
- **Flow** directs boids inside its shape along a configured flow.

Influences combine with other movement forces. They do not replace the habitat or route. For example, Attract can pull boids out of a Volume when temporary exit is allowed, but the Volume's return force continues to act.

Death Zones and Triggers — Separate Assignments

These zones are assigned directly to **SwarmSystem**, rather than to the World's lists.

- **Death Zone** removes boids that cross its area.
- **Trigger** tracks visits or occupancy and fires an event when the configured threshold is reached.

A zone created in the scene must be assigned to the relevant system.

Presets — Behaviour Settings

A preset stores movement and obstacle avoidance response settings. It does not create a World, zones, route or habitat, and does not change the appearance.

Assign a preset and click **Apply** to load its values into the system.

Swarm Control Center

Swarm Control Center is the main window for configuring SwarmSystem. It brings together references, movement settings, space, population and zone creation tools.

Open it via:

```
Tools > Skulmovski Studio > Boids VFX · Flocking, Swarm Navigation & Behaviour · URP > Swarm Control Center
```

To open it with a specific system selected, click **Open in Swarm Control Center** in the SwarmSystem component's Inspector.

Selecting a System

The **Controller** field specifies which SwarmSystem the window edits. **Find in Scene** helps locate a system in the open scene.

Selecting other objects, such as obstacles or Volume parts, keeps the current system assigned. To edit another swarm, change the Controller reference.

Categories

- **Overview** — a summary of the system, population, assigned VFX and movement space.
- **Setup** — main references, automatic playback, mesh selection and orientation.
- **Behaviour** — presets, flocking, speed, Wander and obstacle avoidance response.
- **Space** — Volume or Spline, composite habitats, Follow and scene guides.
- **Environment** — Swarm World, obstacles, Attract, Repel and Flow zones.
- **Population** — Capacity, spawning, lifetime, secondary emission mode and Death Zones.
- **Triggers** — trigger assignments, creation tools and polling settings.
- **Visuals** — access to the linked Visual Effect for appearance settings.

Hints and Appearance

Hints: On / Off shows or hides setting descriptions. Hints are enabled on first use.

The window supports light and dark themes. Your theme and hint preferences are saved between sessions.

Swarm Control Center can be used as a floating window or docked into the Unity Editor layout.

Refresh Swarm

Refresh Swarm recreates the selected system's population. Use it if changes to initial settings or the visual effect are not reflected in the result.

This restarts the population rather than simply refreshing the interface. Existing boids and related state are reset. If the system is stopped, Refresh Swarm starts a new cycle.

Creating Connected Objects

The relevant categories provide tools to create and connect Worlds, obstacles, influences, Composite Volume parts, splines, spawn sources, death zones and triggers.

After creating an object, check its position, shape and dimensions. Zones do not automatically fit themselves to surrounding geometry.

Appearance and shader animation settings are located on the **Visual Effect** component. Control Center manages the simulation and its references.

Playback, Population & Spawning

A population is created for one simulation cycle. Boids can appear all at once or gradually, then continue moving until they die or the system stops.

Starting and Stopping

Play On Enable determines whether the system starts when the component is enabled. Changing this flag does not start or stop the current simulation.

The following operations are available through code:

- **Play()** starts a new cycle after Stop. Calling it again while the system is running does not restart it.
- **Stop()** immediately removes the main population and secondary effects.
- **Reinitialize()** recreates the population on the next simulation step. It does not start a stopped system.

The component and its GameObject must be enabled. Disabling the system releases its resources; enabling it again follows the Play On Enable setting.

Pausing while preserving the simulation state is not supported.

Capacity and Performance

Capacity is the maximum number of boids that can spawn during one cycle. One SwarmSystem supports **1 to 1000 boids**.

Capacity does not maintain a constant living population. Dead slots are not reused. To populate the system again, use Reinitialize or Stop → Play.

Changing Capacity recreates the population rather than adding slots while preserving existing boids.

Neighbour interactions require checking the other living boids in the same system. Increasing the population can therefore raise the workload significantly. Multiple SwarmSystems are calculated separately, but their workloads add up.

See **Performance & Troubleshooting** for population sizing and performance testing recommendations.

Population Counters

- **CAPACITY** — the capacity of the current cycle.
- **SPAWNED** — the number of boids that have spawned, including those that have died.
- **ALIVE** — the living population according to the latest available data.

Living population data is obtained through asynchronous GPU readback, so ALIVE may update with a short delay.

Spawn Modes

- **All At Once** — the entire population appears together.
- **Fixed Batches** — boids appear in batches of a fixed size at a fixed interval.
- **Random Batches** — batch size and interval are chosen within the specified ranges.

The first batch appears after the system is ready, without waiting for the interval. The final batch is limited to the remaining Capacity.

Gradual spawning does not create an endless stream. Once all slots have spawned, no more boids are added.

Spawn Source

The **SwarmSpawn** component defines where boids appear. Supported shapes are **Point**, **Box** and **Sphere**.

Box and Sphere choose positions inside their shape. The source uses its Transform's position, rotation and scale. You can move it between batches so subsequent boids appear at the new location.

Initial direction can be random or follow the source's Forward direction. **Initial Speed** sets the speed at spawn; simulation forces determine subsequent movement.

A Spawn Source is optional. Without an active source, the system uses the current Volume or valid Spline to place the population.

The source controls spawning, but does not keep the swarm nearby. Volume or Spline defines the subsequent movement space.

Do not place the entire source inside an impassable obstacle. The system cannot guarantee a free spawn position in that configuration.

Lifetime and Completion

With **GPU Lifetime** enabled, each boid receives a lifetime within the Min/Max range. Its age starts when it actually spawns, including when spawning in batches.

A dead boid no longer moves or participates in neighbour interactions. Depending on the secondary emission mode, its death can produce an On Die burst.

The death of the entire population does not automatically stop or restart the system. Use **AllBoidsDead** to respond to completion. Details are covered in the death zones and events chapter.

The technical lifetime of the main VFX particles is separate from GPU Lifetime and is described in **Known Constraints & By-Design Behavior**.

Flocking, Motion & Presets

Movement combines neighbour interactions, Wander, spatial constraints, environment influences and obstacle avoidance.

The same visual effect can behave as a tightly packed flock, a loose group or mostly independent particles. Choosing a fish or bird model does not restrict the available movement styles.

Flocking

Flocking uses three main forces:

- **Cohesion** steers a boid towards the average position of its neighbours.
- **Alignment** aligns its movement with the neighbours' average velocity.
- **Separation** pushes it away from neighbours that are too close.

Neighbor Radius determines which neighbours are included. **Separation Radius** defines the separation distance and cannot exceed Neighbor Radius.

Stronger Cohesion gathers the group, Alignment makes movement more coordinated, and Separation helps spread individuals apart and reduce crowding.

Separation is not physical mesh collision. Adjust its distance and strength to suit the size of the visible models.

Speed and Force Response

Max Speed sets the base speed limit. **Speed Multiplier Min/Max** adds stable per-boid variation: each boid's multiplier is not randomly chosen again every frame.

Set both multipliers to 1 to disable this variation.

Max Force limits individual steering calculations. It is not an absolute cap on the final acceleration after all forces are combined.

Acceleration Response scales the response to flocking, Wander and environment influences. Volume boundaries, Spline guidance and obstacle avoidance have their own settings and are not directly scaled by this parameter.

Use Stop to stop the effect rather than reducing Max Speed to zero.

Wander

Wander adds a smoothly changing individual movement direction.

- **Strength** controls the force.
- **Frequency** controls how quickly the direction changes.
- **Persistence** controls directional smoothing.

Wander produces less coordinated movement. Its effect depends on flocking strength and spatial constraints: strong grouping forces can suppress free wandering.

Wander settings do not control wingbeat frequency or body bending. Those settings are located on the Visual Effect component.

Balancing Forces

Attract, Repel and Flow add their own forces, while Volume, Spline and obstacles guide or constrain movement within the scene.

If changing one setting does not produce the expected result, check the other active forces. For example, Volume return can limit pursuit of an Attract zone, while strong Cohesion can keep individuals close to the group.

See **Swarm System — Parameter Reference** for detailed field descriptions and dependencies.

Movement Presets

Presets store movement and obstacle avoidance response settings. Select a **Swarm Preset**, then click **Apply** in the Inspector or **Apply Preset** in Control Center.

Assigning the reference without clicking Apply does not change the current values. You can edit the settings manually after applying a preset.

Presets do not include:

- meshes, textures or shader animation;
- Capacity, spawn sources or spawn schedules;
- Volume, Follow or Spline;
- World, zones, triggers or lifetime.

The same preset can therefore be used with different visual effects and scenes.

Saving and Sharing

Save As saves the current movement settings to a **.swarm** file.

To transfer a preset, copy the file into another project's Assets folder with Boids VFX installed, assign it to the system and apply it.

Reset Movement restores the default movement settings. It does not reset the entire system or replace spatial and visual settings.

Volumes, Boundaries & Follow

In **Volume** mode, boids move freely within a defined habitat. Its shape sets the movement space, while **Boundary & Return** controls how boids react near the boundary and outside it.

Primitive Shapes

Available shapes are **Box**, **Sphere**, **Capsule** and **Cylinder**.

Position and orientation come from the system or a separate **Volume Root**. For a primitive Volume, **Use Rotation** determines whether rotation is used.

Volume dimensions are specified in world units and are not multiplied by Transform Scale. Use Size, Radius and Height to change the shape.

This differs from Spawn Sources and environment zones, which account for Transform scale.

Composite Volume

Composite combines several primitive shapes into one habitat. A boid is considered inside if it is within at least one active part.

Up to **32 active assigned parts** are supported. Each uses a **SwarmVolumePrimitive** component and must be added to the **Volume Primitives** list. Parenting an object under the system alone does not register it.

Composite combines shapes. Subtraction and intersection are not available as separate operations.

Overlap adjacent parts to leave a passage between them. Separate, disconnected regions do not automatically create a migration route.

For example, two large regions can be connected by a Cylinder part that overlaps both. Place a Flow zone inside the connection to guide boids that enter it towards the other region. The Flow zone must be registered in the World assigned to the system.

This creates a passage and a local force, but does not guarantee that the entire population will move from one region to the other.

Boundary & Return

These settings help boids turn before reaching the edge and return after leaving the habitat.

- **Soft Margin** — the width of the inner band where boids begin responding smoothly to the boundary. Increasing it generally allows turns to begin earlier. It does not add space outside the Volume.
- **Steering Strength** — how strongly boundary steering affects movement.
- **Max Boundary Force** — the base boundary force. When temporary exit is allowed, the return force outside the Volume can become stronger than this value.
- **Return Growth** — how quickly the return force increases with distance outside the Volume.



- **Look Ahead** — how far ahead in time the system checks the predicted position using the current velocity. Higher values allow an earlier response to an approaching boundary.

For example, if boids react too late and turn sharply at the edge, check Soft Margin and Look Ahead. If they travel too far outside with temporary exit enabled, check the return force, its growth and any external influences.

These parameters work together. Soft Margin also contributes to the outside return calculation, so changing it can affect behaviour beyond the Volume boundary.

See **Swarm System — Parameter Reference** for detailed descriptions and interactions.

Temporary Exit

With **Allow Temporary Exit** disabled, the boid's centre is additionally constrained to remain inside the habitat. The visible mesh can extend beyond the boundary.

With Allow Temporary Exit enabled, a boid can leave, but the return force acts immediately. There is no configurable delay before returning or fixed permitted exit distance.

Temporary pursuit of an Attract zone requires balancing attraction against return. Consider Strength, Outer Distance, Falloff and target speed. Very strong attraction with a short working range does not provide long-distance pursuit.

Follow Target

Follow moves the habitat with a selected **Transform**. The target does not require a special component.

Boids keep their world positions and catch up with the moving habitat through steering forces. They are not teleported with the target. When the target stops, they continue moving inside the Volume.

Offset uses the target's local coordinates and rotates with it. **Follow Rotation** separately determines whether the habitat itself rotates.

A target that moves too quickly can leave the swarm behind. With temporary exit disabled, rapid boundary movement can cause abrupt position corrections.

Following with a Composite Volume

For a moving Composite, use a separate **Volume Root** parented under SwarmSystem and place the habitat parts beneath it.

Follow Target must not be the system itself, the Volume Root or a descendant of that Root.

Follow controls the habitat. Separately placed Spawn Sources, Splines and World zones do not automatically move with it.

Volume Preview

Composite preview modes are **Wireframe**, **Primitives** and **Off**.

Wireframe displays an approximation of the combined surface. **Resolution** controls preview detail, and **Opacity** controls transparency.

These settings affect Scene View only and do not change the GPU simulation geometry. Narrow connections and small details may require a higher preview resolution.

Spline Navigation

In **Spline** mode, boids follow a route within a soft corridor. Flocking, Wander, obstacles and environment influences remain active.

Creating a Route

Create a **SwarmSplinePath** through the **Space** category in Control Center, or add the component manually. Select Spline mode and assign the path to the system.

The path Inspector provides:

- **Add Point** — adds a control point.
- **Remove Last** — removes the last point.
- **Reverse** — reverses the point order and route direction.

With the path object selected, move its points in **Scene View**.

An open route requires at least two points; a loop requires three. Two points form a straight section. Additional points allow a smooth curve.

Open and Looped Routes

Loop closes the route. There is no need to repeat the first point at the end of the list.

On a looped route, boids continue following the circuit.

On an open route, spline guidance ends for each boid after it passes the endpoint. The boid continues moving under the remaining forces.

There is no automatic reversal, removal, respawn or switch to Volume mode. Use a Death Zone or Lifetime to end the sequence. To move into a habitat, switch spatial modes from a trigger event handler.

Corridor and Guidance

- **Corridor Radius** — the radius of the movement corridor around the route. This is a soft constraint, not an impassable tube.
- **Soft Margin** — the inner band near the corridor edge where return steering increases. It cannot exceed Corridor Radius.
- **Forward Strength** — steering strength along the route.
- **Return Strength** — steering strength back towards the route after a sideways deviation.
- **Look Ahead** — the distance ahead along the route used to guide movement.

Unlike Volume Look Ahead, Spline Look Ahead is a **distance, not a time**.

If boids spread too far sideways, check Return Strength, corridor width and other active forces. If they struggle to follow turns, check speed, Look Ahead and the curve's shape.

Route Detail

Samples Per Span controls curve detail between control points. Higher values represent bends more accurately but increase the segment count.

The GPU route is limited to **256 segments**. Complex curves are automatically sampled within this budget.

Switching Spatial Modes

Switching between Volume and Spline does not recreate the population. Boids keep their positions and begin responding to the new movement space.

To place them again using the new initial conditions, use **Refresh Swarm** or Reinitialize.

If the path is missing, disabled or invalid, the system falls back to Volume. Check the Path reference and control points before running the effect.

World, Obstacles & Influence Zones

Swarm World groups the obstacles and influences used by a system.

Assign the World to SwarmSystem and add zones to its **Obstacles** and **Influences** lists. Placing the objects in the scene alone does not connect them.

The **Environment** category in Control Center provides tools to create and connect these objects.

Sharing a World

Multiple systems can use one World. They share the environment but keep separate populations.

Changing a shared World's lists affects every system using it. Assign separate Worlds when systems need independent environments.

Disabled zones are excluded from calculations. Disabling a World also disables its effect on the systems using it.

Obstacle Shapes

SwarmObstacleZone supports:

- **Box**
- **Sphere**
- **Capsule**
- **Cylinder**

Set the shape manually or obtain it from a supported BoxCollider, SphereCollider or CapsuleCollider. Cylinder is configured manually.

MeshCollider, TerrainCollider and exact collision against the visible mesh's triangles are not used.

Represent buildings and other complex objects with one suitable primitive or several zones.

Avoidance Settings

Response settings are located under **Behaviour** → **Obstacle Avoidance**:

- **Agent Radius** — clearance around the boid's centre.
- **Detection Distance** — how far ahead obstacles are checked.
- **Probe Angle** — the angle used to test alternative avoidance directions.
- **Max Turn Rate** — the maximum turning speed during avoidance.
- **Min Speed Factor** — the minimum speed multiplier when braking near an obstacle.

Wall Margin on the obstacle adds clearance to Agent Radius.

The visible mesh size does not set these values automatically. When increasing model size, check Agent Radius and passage widths.

Avoidance is local. Test complex obstacle intersections and narrow channels at the intended movement speeds.

Open Walls

For Box obstacles, **Open Axes** opens both opposing sides along the selected local axes.

For example:

- **None** — a regular closed Box.
- **X** — opens the X sides, leaving the Y and Z walls.
- **XZ** — leaves only the pair of Y walls.

For Cylinder, **Open Ends** opens both end caps while retaining the side wall.

Use these options to create passages, channels and boundary surfaces. The remaining walls work from both sides.

Passages must account for Agent Radius and Wall Margin. A gap that looks open may still be too narrow for the configured agent.

Influence Zones

SwarmInfluenceZone supports **Attract**, **Repel** and **Flow**. These calculations do not require a Collider.

Available shapes are Sphere, Box, Capsule and Cylinder. The object's position, rotation and scale affect the zone.

Attract

Attract draws interested boids towards its shape.

- **Inner Distance** — the distance at which the current interest in the zone ends.
- **Outer Distance** — the outer working range.
- **Attract Release Distance** — the distance a boid must move beyond after completing its interest before it can select this zone again.
- **Strength** — attraction strength.
- **Attract Interest Chance** — the chance of taking an interest when selecting a zone.

Distances are measured **from the shape's surface**, not necessarily from its centre. A Sphere with Radius = 0 provides a point target.

A boid can have one active attraction target at a time. Interest Chance does not represent a fixed percentage of the swarm, and it is not rerolled every frame without regard to the previous decision.

After rejecting a zone, the boid remembers that decision until it moves beyond Outer Distance. After reaching Inner Distance, renewed interest is blocked until it moves beyond Release Distance.

Falloff

For Attract and Repel, Falloff controls how the force changes across the working range:

- **Constant** — a constant weight.
- **Linear** — linear weakening from Inner to Outer.
- **Smooth** — a smooth transition.

Strength does not set pursuit speed. The result depends on the movement settings and other active forces.

Attraction Outside a Volume

Enable **Allow Temporary Exit** to let boids leave their habitat in response to Attract.

The result depends on attraction range and strength, Falloff, target speed, flocking and Volume return.

If boids turn back too soon, first check Outer Distance and target speed. Increasing Strength will not help if the zone is already out of range.

Repel

Repel pushes boids away from its shape within Outer Distance.

Attract Interest Chance and **Attract Release Distance** are not used in this mode.

Repel is useful for gently parting a swarm around a character or moving object. Use an Obstacle when boids should not pass through a surface.

Flow

Flow acts **inside its shape**, directing movement along the object's local **+Z** axis.

- **Strength** — flow strength.
- **Edge Falloff Distance** — the width of the inner band where the flow weakens near the boundary.
- **Falloff** — the transition within that band.

Inner Distance, Outer Distance and Attract Release Distance do not define Flow's working area.

Turbulence adds smooth changes to the flow:

- **Strength** — the strength of deviations.
- **Scale** — the noise's spatial frequency.
- **Speed** — how quickly the noise changes over time.

Multiple Flow zones can be combined, but they do not automatically form a sequence of waypoints. Each zone acts where the boid is located.

Death Zones, Triggers & Events

Death Zones remove boids. Triggers let you respond when a specified portion of the swarm has visited an area or is currently inside it.

Both zone types are assigned directly to **SwarmSystem**, separately from World.

Death Zones

SwarmDeathZone supports **Box** and **Sphere**. Its shape uses the object's position, rotation and scale.

A boid dies immediately when it crosses the zone. It no longer moves or participates in neighbour interactions. Death does not apply a gradual fade or mesh shrink.

Use these zones to end an open route or remove boids in a particular part of the scene.

Secondary Effects on Death

On Die emission depends on the global **Secondary Emission Mode**.

Override Secondary VFX can suppress On Die emission in that zone. It cannot enable On Die when the global mode excludes it, and it does not change the secondary particles' appearance.

AllBoidsDead

AllBoidsDead fires when **the entire Capacity has spawned and every boid has died**.

A temporary absence of living boids between earlier batches does not count as completion of the whole population.

Data arrives through asynchronous GPU readback, so the event may be delayed. **Polls Per Second** controls the checking rate.

The event does not automatically stop or restart the system. Perform those actions in your own handler if required.

Trigger Zones

One system supports **up to three unique SwarmTriggerZones**. Each zone can belong to only one system at a time.

Triggers use a Box area. Their calculations do not require a Collider.

Create a zone through the **Triggers** category or assign a prepared component manually.

Count Modes

- **Unique Entered** — the number of distinct living boids that have visited the zone since the last Rearm. Repeated entries by the same boid do not increase its contribution.
- **Currently Inside** — the number of living boids inside at the time of counting.

Dead boids are excluded from living statistics, including Unique Entered.

Activation Threshold

Threshold Percent sets the portion of the living population required to fire the event.

The threshold uses the living population in the GPU snapshot, not Capacity. The required count is rounded up.

For example, with 40 living boids and a 25% threshold, 10 are required. With 41 living boids, 11 are required.

Repeat Activation

With **Trigger Once** enabled, the zone fires once until rearmed.

With Trigger Once disabled:

- **Currently Inside** rearms after the count falls below the threshold. Reaching the threshold again fires another event.
- **Unique Entered** requires an explicit Rearm for a fresh count. Boids leaving the zone do not clear its visit history.

Use **trigger.Rearm()** or **swarm.RearmTrigger(trigger)** to rearm one zone. **ResetTriggerHistory()** resets the history of all triggers in the system.

Boids already inside after Rearm can be counted again.

Connecting a Response

The zone component provides a **UnityEvent** that fires when the threshold is reached. For code subscriptions, use **ThresholdReached** or **ThresholdReachedWithData**.

A handler can change an object's colour, start another effect or switch the movement space.

ThresholdReachedWithData also supplies statistics: visits, occupancy, the snapshot's living population, percentage and count mode.

Choose one subscription method for a single action.

Polling and Edit Mode

Polls Per Second controls how often statistics are retrieved, not the simulation rate.

An event is not guaranteed to arrive in the frame of crossing. Do not use it as a substitute for precise per-frame physical contact.

Events can also run in **Edit Mode**. If a handler changes serialised scene settings, those changes can remain after it fires. Keep this in mind when testing mode switches and demo handlers.

Visual Effects & Custom Mesh Preparation

SwarmSystem calculates movement. The linked **Visual Effect** component controls appearance.

The asset provides four visual types:

- **Fish** — lateral body bending.
- **Bird / Wing** — procedural wing flapping, with Simple and Expensive variants.
- **Non-Deforming Mesh** — a mesh without specialised deformation.
- **Quad** — a textured quad.

All variants use the same movement system.

Where to Change Settings

Select Source Mesh and its orientation in Control Center's **Setup** category.

Configure size, textures, colour, material, animation and secondary particle appearance on the **Visual Effect** component.

Do not treat system-controlled properties as ordinary visual settings. SwarmSystem may overwrite their values.

Complete field lists are provided in the references for each graph.

Mesh Orientation

A prepared mesh uses these directions:

- **+Z** — forward, towards the nose.
- **+Y** — up.
- **±X** — the sides; for wing animation, the left and right wings.

If the source model uses different directions, specify its **Forward** and **Up** axes in Setup. These axes must be perpendicular.

The editor can create a rotated, prepared mesh copy while preserving the source model. Enable **Read/Write** in the import settings so its geometry can be read.

After changing the model or axes, check the prepared mesh before building the project. Copy preparation runs in the editor, not automatically in the built application.

Size and Geometry

BaseSize and **ScaleVariation** change visual size. They do not adjust Agent Radius, Volume or environment zones.

Smooth deformation requires vertices where the mesh bends. A large, flat face cannot bend like a more detailed surface.

Assigning a rigged model does not run an Animator for each boid. The supplied animations deform mesh vertices in the shader.

Vertex Colour Mask

Fish and Wing animation use **Vertex Color.R — the red vertex colour channel**.

- **0, black** — no mask influence.
- **1, white** — full influence.
- **Intermediate values** — partial influence and smooth transitions.

You can paint the mask in greyscale for convenience. The shader reads the red channel, not texture brightness or material colour.

The mask controls each vertex's participation in deformation. The final motion also depends on animation settings and vertex position.

Preserve vertex colours during export. Do not assume a model without a prepared mask will automatically receive a suitable deformation pattern.

Fish Mask

Keep stationary parts closer to black and increase the influence on areas that should bend. A smooth gradient is usually useful between the head and the moving body.

The supplied Fish graph bends the model sideways along local X, with the body's length along Z.

It does not automatically produce the vertical tail movement of a dolphin or orca. That swimming style requires a different deformation.

After assigning the mesh, adjust HeadZ/TailZ and the wave settings to fit the model's dimensions.

Wing Mask

The body and other stationary parts are usually black, while the wings are light. Use a gradual transition near the roots where needed.

Wings must lie on opposite sides of local X. Their shape, Shoulder/Elbow Pivot positions and mask must work together.

Simple and Expensive use different deformation methods. Neither is cloth simulation or skeletal animation.

Suitable models can represent birds, butterflies, moths, bees and other winged creatures. Adjust the settings to the geometry of each model.

Preparing a Mask in Blender

Interface placement may differ between Blender versions.

1. Import and select the mesh.
2. Under **Object Data Properties** → **Color Attributes**, create a colour attribute and make it active.
3. Switch to **Vertex Paint**.
4. Fill the model with black using **Set Vertex Colors**.
5. Paint moving areas white or grey. Check both sides and hidden surfaces.
6. When exporting FBX, include **Vertex Colors** rather than selecting None. If there are several attributes, use **Prioritize Active Color** where available.
7. In Unity, enable Read/Write, assign the mesh and check the mask with the appropriate animated VFX.

Vertex Paint stores colours in a Color Attribute rather than a texture.

Checking the Result

If the model moves correctly but deforms incorrectly, check:

- the appropriate animated VFX is selected;
- the mesh axes are correct;
- the red vertex colour channel survived export and import;
- deformation settings match the model's dimensions;
- the mesh has enough vertices for the intended bend.

Use Non-Deforming Mesh VFX for meshes that do not need deformation. It does not require a fish or wing animation mask.

Secondary VFX

The supplied graphs contain two secondary branches:

- **SC — Secondary Continuous** emits particles while a boid is alive.
- **SD — Secondary Die** creates particles when a boid dies.

The branches have independent visual settings but fixed roles.

Emission Mode

Secondary Emission Mode is configured in SwarmSystem:

- **Disabled** — both branches are off.
- **Continuous** — only SC is active.
- **On Die** — only SD is active.
- **Both** — both branches are active.

The mode does not reassign SC to death emission or SD to continuous emission.

Rate and Count

Continuous Rate / Alive Boid sets the continuous emission rate per living boid.

For example, 100 living boids at a rate of 2 target approximately 200 events per second. The visible result also depends on graph settings and capacity.

The corresponding **SecondaryContinuousRate** field on the Visual Effect is controlled by the system and overwritten during updates.

SecondaryDieCount on the Visual Effect sets the number of secondary particles created for one permitted boid death.

Where to Change Appearance

Emission mode and Continuous Rate are set in the controller. Other settings are located on the **Visual Effect** component.

Properties prefixed with **SC_** belong to Continuous. Properties prefixed with **SD_** belong to On Die.

The following reference applies to both branches. For example, **SC_Lifetime** and **SD_Lifetime** serve the same purpose but have independent values.

Initial Settings

- **Lifetime** — the secondary particle's lifetime range in seconds. This is separate from the main boid's GPU Lifetime.
- **Size** — the initial size range.
- **StartSpeed** — the initial speed range.
- **InheritVelocity** — the proportion of the boid's velocity passed to the secondary particle.
- **Offset** — the initial position offset from the event position.
- **AngleFrom / AngleTo** — the lower and upper bounds for random initial Euler angles. These control particle orientation, not Flow or route direction.

Movement

- **Gravity** — an acceleration vector. Use it for falling, rising or acceleration in a chosen direction.
- **Drag** — resistance to motion. Higher values slow particles more strongly.

These settings affect secondary particles independently of the main swarm's movement.

Turbulence

- **TurbulenceIntensity** — the strength of the noise force.
- **TurbulenceRequency** — the noise's spatial frequency. The current property name retains the spelling Requency.
- **TurbulenceDrag** — drag used by the Turbulence block.
- **TurbulenceOctaves** — the number of noise layers. Additional octaves increase detail and calculation cost.
- **TurbulenceRoughness** — the relative contribution of finer noise detail.

- **TurbulenceLacunarity** — the increase in spatial frequency between octaves.

Roughness and Lacunarity are most noticeable when using multiple octaves.

Colour, Size and Texture

- **Texture** — the secondary particle texture.
- **ColorOverLife** — colour and transparency throughout the particle's lifetime.
- **SizeOverLife** — size changes throughout its lifetime.
- **AlphaThreshold** — the alpha clipping threshold.
- **SoftParticleDistance** — the distance used to soften intersections with scene geometry based on depth. This is visual softening, not obstacle collision.
- **Metallic** — the metallic appearance of Lit particles.
- **Smoothness** — the surface smoothness of Lit particles.

For ColorOverLife and SizeOverLife, the start of the range represents particle birth and the end represents the end of its lifetime.

Death and Stop

On Die can fire when a boid dies from Lifetime or a Death Zone, provided the global emission mode allows it.

A Death Zone can suppress On Die within its area, but cannot enable it against the Secondary Emission Mode setting.

After a boid dies, secondary particles already created can remain until their own Lifetime ends.

Stop immediately removes both secondary branches and does not generate On Die bursts for the stopped population.

Limitations and Performance

Secondary particles do not participate in flocking or automatically use the main system's World obstacles.

The workload depends on living population, Continuous Rate, SecondaryDieCount, Lifetime and visual settings. At the same emission rate, a longer Lifetime leaves more particles alive at once.

If emission appears to be cut short, check the capacity of the corresponding VFX Graph branch. Increasing Rate or DieCount alone does not increase its Capacity.

Material Colours to Texture Baker

Material Colors to Texture transfers solid material colours into one texture and creates a mesh copy with a single submesh.

This is useful for models whose parts use separate coloured materials. For example, a fish's body, eyes and fins may use different material slots. The baker stores their base colours in a texture that can be assigned to the VFX.

Opening the Tool

Open the baker via:

```
Tools > Skulmovski Studio > Boids VFX · Flocking, Swarm Navigation & Behaviour · URP > Material Colors to Texture
```

It is also available through the **Tools** menu in Swarm Control Center and **Open Baker** in Setup Assistant.

Model Requirements

- One selected Renderer with a mesh.
- An existing **UV0 layout** within the 0–1 range.
- Triangle geometry.
- Material slots matching the number of submeshes.
- Materials using opaque, solid base colours.

Both **MeshRenderer with MeshFilter** and **SkinnedMeshRenderer** are supported. For SkinnedMeshRenderer, the source sharedMesh is used rather than the current animated pose.

If the tool cannot read the geometry or UVs, enable **Read/Write** in the model's import settings and click Apply.

Workflow

1. Assign a model, prefab or scene object to **Source Object**.
2. If several Renderers are found, select the required one.
3. Set **Texture Size** to 256, 512, 1024 or 2048.
4. Choose a **UV Overlap** mode.
5. Set an **Output Folder** inside Assets and an **Output Name**.
6. Click **Create Texture & Mesh**.

One mesh is processed per bake. Separate Renderers are not automatically combined.

Overlapping UVs

If areas with different material colours share the same UV coordinates, both colours cannot be stored in the same texture location.

Two modes are available:

- **Warn and continue** — continues baking, keeps the first colour written and reports conflicting pixels.
- **Stop on overlap** — stops baking when a conflict is detected.

Overlaps with the same colour are not treated as conflicts.

Checks are performed at the selected texture resolution. Very small UV regions may require a larger Texture Size.

Output

The tool creates two files:

- **_BaseColor.png** — the base colour texture.
- **_Mesh.asset** — a mesh copy with one submesh.

Source models and materials are not overwritten. If the name is already in use, the tool adds an available suffix.

Assign the generated mesh to the system's **Source Mesh** and the texture to **MeshTexture** on the Visual Effect component.

Use **Refresh Swarm** after assignment if needed.

What the Baker Does Not Transfer

The tool is designed for solid base colours. It does not bake:

- source material textures;
- the final Shader Graph result;
- lighting or shadows;
- Normal, AO, Emission or a complete PBR texture set;
- the current skeletal animation pose.

The baker does not create a UV layout. Materials with base textures or transparency require preparation before baking.

Vertex Colours and Animation

The mesh copy preserves existing vertex colours, but the baker does not create an animation mask.

The resulting colour texture does not replace **Vertex Color.R** for Fish Body Bending or Wing Flapping. Prepare that mask separately.

Runtime API

The Runtime API lets you control the system from your own code: start and stop effects, change behaviour, switch movement space and connect prepared zones.

All methods are called on **SwarmSystem**. No separate API component is required.

Getting a Reference

Store a SwarmSystem reference in your script or obtain it through GetComponent.

The main namespace is:

```
using SkulmovskiStudio.VFX_Boids.Swarm.Scripts.Swarm;
```

For parameter transition modes:

```
using SkulmovskiStudio.VFX_Boids.Swarm.Scripts.Swarm.Utilities;
```

In the examples below, **swarm** is a reference to an existing SwarmSystem.

Starting and Stopping

```
swarm.Play();
swarm.Stop();
swarm.Reinitialize();
```

- **Play()** starts a new cycle after Stop. Calling it again while running does not restart the system.
- **Stop()** immediately removes the main population and secondary effects.
- **Reinitialize()** recreates the population on the next simulation step without starting a stopped system.

To recreate the population with a different capacity:

```
swarm.Reinitialize(300);
```

Capacity must be **1–1000**. The existing population is discarded.

If playback should be controlled entirely by your code, disable **Play On Enable** beforehand. The component and GameObject must remain enabled.



System State

- **IsPlaying** — whether the system is set to run.
- **SpawnCapacity** or **GetCapacity()** — the configured capacity.
- **ActiveBoidCount** or **GetSpawnedCount()** — the number of boids spawned, including those that have died.
- **AliveBoidCount** or **GetAliveCount()** — the latest living population estimate.
- **IsSpawnComplete** — whether every slot in the current population has spawned.
- **AllBoidsDeadFired** — whether the completion event has fired during the current cycle.

IsPlaying does not guarantee that GPU resources are ready or that living boids remain.

The living count comes from asynchronous GPU readback and may lag behind the simulation.

Changing Behaviour

Numeric settings provide Get and Set methods. For example:

```
float speed = swarm.GetMaxSpeed();

swarm.SetMaxSpeed(4f);
swarm.SetCohesionWeight(1.5f);
swarm.SetWanderStrength(0.8f);
```

Available settings cover:

- Max Speed and Max Force;
- Acceleration Response;
- Cohesion, Alignment and Separation;
- Neighbor Radius and Separation Radius;
- Wander Strength, Frequency and Persistence;
- per-boid speed variation;
- obstacle avoidance response.

These changes do not recreate the population.

Smooth Transitions

Numeric setters that accept **duration** and **ease** can change a value gradually:

```
swarm.SetMaxSpeed(6f, 2f, SwarmEase.Smooth);
```

This transitions the speed setting from its current value to 6 over two seconds.

- **duration = 0** — applies immediately.
- **SwarmEase.Linear** — a linear transition.
- **SwarmEase.Smooth** — a transition with a smooth start and finish.

Calling the setter again for the same parameter replaces its current transition.

```
swarm.CancelAllAnimations();
```

CancelAllAnimations() stops numeric transitions at their current values. It does not stop the simulation or shader mesh animation.

Related Values and Argument Validation

Some parameters depend on one another. Paired methods allow them to be changed together:

```
swarm.SetFlockingRadii(3f, 0.5f);
swarm.SetSpeedVariation(0.9f, 1.1f);
swarm.SetSplineCorridor(2f, 0.5f);
```

- **SetFlockingRadii** sets Neighbor Radius and Separation Radius.
- **SetSpeedVariation** sets the minimum and maximum speed multipliers.
- **SetSplineCorridor** sets the corridor radius and its Soft Margin.

These calls apply both values immediately and cancel their current transitions.

The API checks ranges, dependencies and argument validity. Invalid values can throw exceptions. Do not pass NaN, Infinity, negative durations or incompatible values.

Presets

```
swarm.ApplyPreset();
swarm.ResetMovementToDefaults();
```

ApplyPreset() applies the assigned preset. **ApplyPreset(preset)** accepts another prepared SwarmPresetAsset.

Applying a preset does not recreate boids, but it cancels current numeric transitions.

ResetMovementToDefaults() restores default movement and avoidance settings. It does not reset the entire system.

Volume and Composite

Change the shape through:

- **SetVolumeBox(size)**
- **SetVolumeSphere(radius)**
- **SetVolumeCapsule(radius, height)**
- **SetVolumeCylinder(radius, height)**
- **SetVolumeComposite()**

For prepared Composite parts:

- **AddVolumePart(part)** — registers a part.
- **RemoveVolumePart(part)** — removes it from the list.
- **GetVolumeParts()** — returns the assigned parts.



AddVolumePart does not create an object or automatically switch the shape to Composite. Up to 32 unique parts are supported.

Set the volume origin and rotation behaviour through **SetVolumeRoot(root)** and **SetVolumeUseRotation(enable)**.

Boundary and return settings are available through:

- **SetVolumeAllowTemporaryExit(enable)**
- **SetVolumeReturnGrowth(...)**
- **SetVolumeMargin(...)**
- **SetVolumeWeight(...)**
- **SetVolumeMaxForce(...)**
- **SetVolumeLookAhead(...)**

Changing the shape does not recreate the population. However, shrinking a volume with temporary exit disabled can immediately correct the positions of boids left outside.

Follow

The following methods control a habitat that follows a target:

- **SetVolumeFollowTarget(target)**
- **SetVolumeFollowEnabled(enable)**
- **SetVolumeFollowOffset(offset)**
- **SetVolumeFollowRotation(enable)**

The corresponding Get methods return the current settings.

For a moving Composite, first prepare a separate child Volume Root and place the parts under it. The API checks the required hierarchy constraints.

Spline and Spatial Mode

```
swarm.SetSplinePath(path);
swarm.SetSplineLoop(true);
swarm.SetSpatialMode(SwarmSpatialMode.Spline);
```

path is a prepared SwarmSplinePath.

Assigning a Path does not switch the system to Spline automatically. Set the mode separately through SetSpatialMode.

To return to Volume:

```
swarm.SetSpatialMode(SwarmSpatialMode.Volume);
```

Route settings are available through:

- **SetSplineCorridor(radius, softMargin)**
- **SetSplineCorridorRadius(...)**

- **SetSplineSoftMargin(...)**
- **SetSplineForwardStrength(...)**
- **SetSplineReturnStrength(...)**
- **SetSplineLookAhead(...)**

Changing the mode or route preserves boid positions. Progress along the spline is recalculated.

The SwarmSystem API supports assigning a prepared route and configuring its guidance. It does not expose control-point editing through SwarmSystem.

Loop is stored on the Path itself. If several systems share a route, changing Loop modifies that shared object.

World and Prepared Zones

- **GetWorld() / SetWorld(world)** — gets or assigns a World.
- **AddObstacle(zone) / RemoveObstacle(zone)** — registers or unregisters an obstacle.
- **AddInfluence(zone) / RemoveInfluence(zone)** — registers or unregisters an influence.
- **GetObstacles() / GetInfluences()** — returns the registered zones.
- **ResetAttractionState()** — clears current attraction choices and interest history without recreating boids.

Assign a World before calling Add/RemoveObstacle or Add/RemoveInfluence.

These methods accept references to existing components. They do not create zones or configure their shapes.

Remove unregisters a zone without destroying its GameObject. Pass the same reference to remove a specific zone; no separate identifier is required.

If several systems share the World, changing its lists affects all of them.

Triggers

- **SetTriggers(...)** — replaces the trigger list.
- **AddTrigger(trigger)** — registers a prepared zone.
- **RemoveTrigger(trigger)** — unregisters a zone without destroying its object.
- **GetTriggers()** — returns the assigned zones.
- **RearmTrigger(trigger)** — rearms one zone.
- **ResetTriggerHistory()** — resets the history of all zones.

Up to three unique triggers are allowed. A zone must not belong to another system.

Changing the trigger list resets its history. Use RearmTrigger to rearm one zone without resetting the others.

Lifetime and Secondary Effects

```
swarm.SetLifetime(true, 5f, 10f);
swarm.SetSecondaryEmission(SecondaryEmissionMode.Both, 2f);
```

SetLifetime changes the lifetime range without resetting existing boid ages. Enabling Lifetime or reducing the range can cause boids to die on the next step. Dead boids remain dead.

SetSecondaryEmission sets the shared secondary emission mode and Continuous Rate per living boid. It does not change textures, colours or other branch appearance settings.

Completion Event

Use **AllBoidsDead** to respond when the population completes:

```
swarm.AllBoidsDead += OnAllBoidsDead;
```

The handler receives the system reference:

```
private void OnAllBoidsDead(SwarmSystem completedSwarm)
   {
       completedSwarm.Stop();
   }
```

Unsubscribe when the subscriber is no longer in use:

```
swarm.AllBoidsDead -= OnAllBoidsDead;
```

The event arrives after the entire Capacity has spawned and all boids have died. Calling Stop does not raise it.

Trigger events are connected on the relevant SwarmTriggerZone and are covered in **Death Zones, Triggers & Events**.

What Recreates the Population

Forces, boundary settings, Volume dimensions, Follow, routes, Worlds and prepared zone registrations can be changed without recreating the population.

Related state may still reset, such as route progress or attraction choice history.

Reinitialize and Capacity changes recreate the population. Stop → Play also starts a new cycle.

Not every Inspector field has a SwarmSystem setter. In particular, the API does not provide separate methods for changing the spawn schedule or Death Zone list.

Lists returned by GetVolumeParts, GetTriggers, GetObstacles and GetInfluences are intended for reading. Retrieve the list again after changing its contents.

Performance & Troubleshooting

The workload depends on population, environment and visual settings. Test the final configuration in a build on your target platform. Performance in the Unity Editor, particularly with Scene View and previews enabled, can differ from the built application.

Boid Count

One SwarmSystem supports up to **1000 boids**. This is a supported limit, not a recommended count for every device.

In the current implementation, each living boid checks the other living boids in its system. Neighbour-checking cost grows approximately quadratically: doubling the living population can produce roughly four times as many neighbour checks.

This does not mean the entire frame becomes four times more expensive. Rendering and other calculations have their own costs.

Reducing **Neighbor Radius** changes behaviour but does not eliminate the iteration over other boids.

Multiple systems are calculated separately, and their workloads add up. Splitting a population also changes behaviour: boids from different systems do not treat one another as neighbours.

Meshes and Animation

Consider both the source mesh's vertex count and the number of visible instances.

Shader deformation runs on the rendered geometry. Large numbers of detailed animated models can cost noticeably more than simple meshes.

Wing Flapping provides **Simple** and **Expensive** variants. Compare their appearance and cost on your models. The more complex variant may not be necessary for small or distant creatures.

Use **Non-Deforming Mesh VFX** when deformation is not needed. Quads are another option for suitable effects, but their cost still depends on screen size, material and particle overlap.

Obstacles and Influences

Each active zone adds simulation work. Keep only the obstacles and influences needed by the systems using the assigned World.

Represent complex geometry with a reasonable number of primitives. Many small obstacles can increase the workload and make avoidance more difficult.

Avoid passages with little clearance left after accounting for **Agent Radius** and **Wall Margin**.

Narrow gaps between obstacles and Volume boundaries can cause crowding, repeated turns and trapping. Check geometry and available space first, then adjust the balance of forces.

Secondary Particles

Secondary effect cost depends on:

- living boid count;
- Continuous Rate;
- SecondaryDieCount;
- secondary particle Lifetime;
- Turbulence and octave count;
- particle screen size and overlap.

At the same emission rate, longer Lifetime leaves more particles alive at once. A high Rate combined with long Lifetime can create far more secondary particles than main boids.

Increase secondary branch Capacity only when needed. More capacity does not replace tuning the emission rate and lifetime.

Triggers and Monitoring

Polls Per Second controls how often statistics are retrieved from the GPU. Increase it only when the sequence needs a faster response.

A higher polling rate does not make events synchronous or change the main simulation rate.

Editor Previews

Detailed Composite previews and large numbers of Gizmos can add editor overhead.

Use enough preview Resolution for the task rather than always selecting the maximum. Disable unnecessary previews when checking performance and compare against the target build.

Preview settings do not change the GPU simulation geometry.

Measuring Performance

Use the **Unity Profiler** and **GPU Profiler** where supported by the target platform and graphics API.

Change one factor at a time: population, mesh complexity, animation, zone count or secondary emission. This makes it easier to identify the limiting factor.

Test the effect in the working scene, including its lighting, materials and other systems, rather than only in isolation.

Boids Are Not Visible

Check that:

- importing and compilation have finished;
- SwarmSystem and its GameObject are enabled;
- the system has started through Play On Enable or Play;
- a compatible Visual Effect is assigned;
- the compute shader is assigned;
- a mesh is available for a mesh-based effect;
- boids are not dying immediately from Lifetime or a Death Zone.

If a freshly imported VFX Graph renders incorrectly, use **Recompile VFX Graphs** in Setup Assistant and wait for compilation to finish.

After changing initial settings, try **Refresh Swarm**. This recreates the population.

An Obstacle or Influence Has No Effect

Ensure that:

- the correct **Swarm World** is assigned to the system;
- the zone is registered in the corresponding World list;
- both the World and the zone are enabled;
- the shape, dimensions and working range fit the scene.

For Flow, check the local **+Z** direction. For Attract and Repel, remember that distances are measured from the shape's surface.

Death Zones and Triggers are assigned directly to the system, not to World.

Boids Become Trapped Against a Wall or Boundary

Check whether there is enough room to turn between the obstacle and Volume.

Account for Agent Radius and Wall Margin. Then check movement speed, detection distance and avoidance turn rate.

If boids appear pressed against a surface, several forces may be pulling in competing directions. Simplify the area, increase clearance or rebalance the forces. Local avoidance does not guarantee recovery from every enclosed configuration.

Boids Return from Attract Too Soon

Check **Outer Distance**, Falloff and target speed. High Strength does not help beyond the working range.

Ensure **Allow Temporary Exit** is enabled, then check Boundary & Return and the other swarm forces.

Soft Margin does not define a permitted distance outside the Volume.

Boids Fly Away After the Spline Ends

This is expected for an open route: spline guidance ends after the boid passes the endpoint.

Add a Death Zone, use Lifetime or switch the system to Volume through an event handler.

The Mesh Moves Sideways or Upside Down

Check **Forward** and **Up** in Setup. The prepared mesh should use +Z forward and +Y up.

Enable Read/Write on the source model to prepare a rotated copy. After changing the mesh or axes, check the result and use Refresh Swarm if needed.

Fish or Wings Deform Incorrectly

Check that:

- the appropriate animated graph is selected;
- the **Vertex Color.R** mask has been preserved;
- the mesh is oriented correctly;
- HeadZ/TailZ or Wing Pivot/Blend settings match the model's dimensions;
- there are enough vertices for smooth deformation.

Material colour and textures do not replace the vertex mask.

Follow Does Not Move a Composite Correctly

Check the separate child **Volume Root**, the placement of parts beneath it and their assignment in Volume Primitives.

Follow Target must not be the system, the Root or a descendant of the Root.

If the habitat moves correctly but the swarm falls behind, check target speed and the movement and return settings.

A Trigger Does Not Fire Again

Check Count Mode, Trigger Once and whether **Rearm** is required.

In Unique Entered mode, boids leaving the zone do not clear visit history. Call Rearm to begin a new count.

Allow for polling delay and remember that the threshold uses the living population, not Capacity.

Secondary Particles Are Missing or Emission Is Cut Short

Check:

- Secondary Emission Mode;
- Continuous Rate or SecondaryDieCount;
- whether a death actually occurs for On Die;
- whether a Death Zone suppresses the burst;
- particle Lifetime, size, transparency and texture;
- the corresponding graph branch's Capacity.

Stop does not generate On Die bursts and immediately removes secondary effects.

The Population Decreases After Deaths and Does Not Recover

Dead slots are not automatically reused. Use **Reinitialize** or **Stop** → **Play** to create a new population.

Rendering Disappears After a Long Run

The main visual particles have a technical lifetime of **9999 seconds of VFX time**.

After it expires, the compute simulation can continue without rendering. For longer sequences, plan a new cycle that recreates the population.

See **Known Constraints & By-Design Behavior** for details.

Swarm System — Parameter Reference

Settings are grouped by **Swarm Control Center** category. Some fields appear only for the selected mode or shape.

Distances use Unity world units unless stated otherwise. Instance values may differ from the initial defaults after applying a preset.

Setup — Connections

- `Visual Effect` — the component with a compatible VFX Graph that renders the system's population.
- `World` — the Swarm World supplying obstacles and influences. Multiple systems can share it. It is optional for basic movement without those zones.
- `Play On Enable` — starts a new cycle when the component is enabled. Changing the flag does not start or stop the current cycle.

The Inspector also provides **Compute Shader**, the simulation shader reference. In the editor, the system finds the supplied compute shader automatically. Ensure the reference is saved before building.

Setup — Mesh Orientation

- `Source Mesh` — the main effect's mesh. When assigned, it overrides Mesh in a compatible graph. An empty reference leaves the graph's mesh unchanged.
- `Forward` — the axis pointing towards the source model's nose.
- `Up` — the source model's upward axis. It must be perpendicular to Forward.

The prepared mesh uses **+Z forward and +Y up**. Enable Read/Write on the source mesh to create a rotated copy.

Source Mesh applies to mesh-based graphs. Quad does not use this field.

Behaviour — Movement Preset

- Preset — a movement settings file. Assigning the reference does not apply its values.
- Apply Preset — loads the selected preset's settings.
- Save As... — saves the current movement and obstacle avoidance response settings to a separate .swarm file.
- Reset Movement — restores default movement settings. It does not reset space, population or appearance.

Behaviour — Flocking

- Neighbor Radius — the neighbour search radius for flocking. Higher values include more distant individuals. Minimum: 0.01.
- Separation Radius — the close-range separation distance. It cannot exceed Neighbor Radius. Minimum: 0.01.
- Cohesion Weight — the strength of steering towards the neighbours' average position. Higher values gather the group more tightly. Zero disables this contribution.

- Alignment Weight — the strength of matching the neighbours' average velocity. Higher values produce more coordinated directions. Zero disables this contribution.
- Separation Weight — the strength of repulsion from nearby neighbours. It reduces crowding but does not create physical mesh collision.

Neighbor Radius determines which neighbours contribute, but reducing it does not eliminate iteration over the population.

Behaviour — Motion

- `Max Speed` — the base speed limit in units per second, multiplied by the individual Speed Multiplier. Minimum: 0.01.
- `Max Force` — the base limit for individual steering calculations. It is not a cap on the final acceleration after all forces are combined. Minimum: 0.01.
- `Acceleration Response` — a shared response multiplier for flocking, Wander and influences. It does not directly scale Volume, Spline or obstacle avoidance.
- `Speed Multiplier Min` — the lower limit of individual speed variation, from 0.5 to 1.
- `Speed Multiplier Max` — the upper limit of individual speed variation, from 1 to 1.5.

Each boid's multiplier remains stable rather than being rerolled every frame. Set both values to 1 to disable variation.

Behaviour — Wander

- `Wander Strength` — the strength of wandering. Higher values make individual movement deviate more strongly from other forces. Zero disables the Wander contribution.
- `Wander Frequency` — how quickly the wandering phase changes. Higher values produce faster changes; this is not the shader animation frequency.
- `Wander Persistence` — how long the previous direction is retained and smoothed, in seconds. Higher values produce more gradual direction changes. Zero disables filtering.

Behaviour — Obstacle Avoidance

- `Obstacle Agent Radius` — clearance around the boid's centre during avoidance. It does not automatically follow mesh size.
- `Obstacle Detection Distance` — the base obstacle-checking distance. Actual look-ahead can increase to account for movement and the space needed to turn.
- `Obstacle Probe Angle` — the spread of tested avoidance directions, from 10 to 85 degrees.
- `Obstacle Max Turn Rate` — the maximum avoidance turning speed, from 1 to 1080 degrees per second.
- `Obstacle Min Speed Factor` — the minimum speed multiplier when braking near an obstacle, from 0.05 to 1. Lower values allow stronger slowing.

Agent Radius is combined with the obstacle's Wall Margin. Excessive clearance can make a narrow passage inaccessible.

Space — Spatial Mode

- `Mode` — Volume or Spline. Volume defines a habitat; Spline guides movement along a route.

If the assigned spline is missing, disabled or invalid, the system falls back to Volume.

Space — Volume Shape

- `Shape` — Box, Sphere, Capsule, Cylinder or Composite.
- `Use Rotation` — uses the volume origin's rotation for primitive shapes. Rotation does not change a Sphere's geometry; Composite uses each part's Transform.
- `Size` — the Box's full dimensions along X, Y and Z.
- `Radius` — the radius of a Sphere, Capsule or Cylinder.
- `Height` — the full height of a Capsule or Cylinder. A Capsule's effective height cannot be less than its diameter.
- `Primitive Parts` — the Composite Volume's parts. Up to 32 active assigned parts are used.

Volume dimensions are specified in world units and are not multiplied by Transform Scale.

Space — Scene Preview

These settings apply to Composite:

- `Display` — Wireframe, Primitives or Off.
- `Resolution` — detail level of the approximate combined surface, from 12 to 40.
- `Opacity` — Wireframe opacity, from 0.003 to 0.5.

These settings affect the Scene View preview only.

Space — Placement & Following

- `Volume Root` — a separate Transform used as the volume origin. A moving Composite requires a child Root containing its parts.
- `Create / Group Root` — creates or organises a shared Root for the parts.
- `Enable Following` — enables following without clearing the assigned target when disabled.
- `Follow Target` — the Transform followed by the habitat.
- `Offset` — an offset in the target's local coordinates. It rotates with the target regardless of Follow Rotation.
- `Follow Rotation` — transfers the target's rotation to the habitat while following.

Without a separate Root, disabling Follow returns a primitive Volume to the system's position. A separate Root retains its last position.

Space — Boundary & Return

- `Allow Temporary Exit` — allows boids outside the habitat, with steering forces bringing them back. When disabled, each boid's centre is additionally constrained to remain inside.
- `Return Growth` — how quickly the return force increases with distance outside. Higher values bring distant boids back more strongly. This is not a delay.

- `Soft Margin` — the width of the inner boundary-response band. A wider band generally allows earlier turns. It also contributes to the outside return calculation.
- `Steering Strength` — the weight of boundary steering.
- `Max Boundary Force` — the base force and limit for regular boundary steering. With temporary exit enabled, outside return can increase to eight times this value.
- `Look Ahead` — position prediction time based on current velocity, in seconds. Higher values allow an earlier response.

Soft Margin does not add permitted space outside the Volume. Exit distance depends on the balance of forces, not one field.

Space — Spline Route

- `Path` — the SwarmSplinePath component containing the route.
- `Spline Corridor Radius` — the radius of the soft corridor around the route. Minimum: 0.01.
- `Spline Soft Margin` — the inner transition band for return steering, from 0 to Corridor Radius.
- `Spline Forward Strength` — the weight of steering along the route.
- `Spline Return Strength` — the weight of lateral return towards the route. Outside the corridor, return is further strengthened with distance.
- `Spline Look Ahead` — the distance ahead along the route. Minimum: 0.05.

Spline Look Ahead is a distance; Volume Look Ahead is a time.

Space — Scene Guides

- `Draw Bounds` — displays the habitat and enables the Composite preview.
- `Draw Margin` — displays the inner boundary band of a primitive Volume.

Hiding guides does not disable spatial constraints.

Environment

- `World` — the environment used by the system.
- `Obstacles` — obstacles registered in that World.
- `Influences` — registered Attract, Repel and Flow zones.

The category's tools create and connect zones. Shape and influence settings are located on their components and covered in the next chapter.

Population — Capacity

- `Capacity` — the maximum population per cycle, from 1 to 1000 boids. Changing it recreates the population.

Dead slots are not automatically reused.

Population — Spawn System

- `Source` — an optional SwarmSpawn. Without an active source, the system uses Volume or a valid Spline.
- `Initial Direction` — Random or the active source's Forward direction.
- `Initial Speed` — speed at spawn, in units per second. Subsequent movement is determined by the simulation.

Spawn Mode & Batches

- `Mode` — All At Once, Fixed Batches or Random Batches.
- `Boids per Batch` — the fixed batch size.
- `Interval (s)` — the fixed batch interval.
- `Min Boids per Batch` — the minimum random batch size.
- `Max Boids per Batch` — the maximum random batch size.
- `Min Interval (s)` — the minimum random interval.
- `Max Interval (s)` — the maximum random interval.

The minimum batch size is 1, and the minimum interval is 0.01 seconds. Maximum values must not be below their corresponding minimums.

The first batch does not wait for the interval. The final batch is limited to the remaining Capacity.

Population — Lifetime

- `Enable Lifetime` — allows boids to die from age.
- `Lifetime Min` — the lower lifetime limit, at least 0.01 seconds.
- `Lifetime Max` — the upper limit, no lower than Lifetime Min.

Age starts at actual spawn. Changing the range does not reset existing boid ages.

This is separate from the main VFX particles' technical lifetime.

Population — Secondary Emission

- `Emission Mode` — Disabled, On Die, Continuous or Both.
- `Rate / Alive Boid` — the Continuous emission rate per living boid. Used in Continuous and Both modes.

SC/SD appearance and the On Die particle count are configured on the Visual Effect. See **Secondary VFX** for the shared reference.

Population — Death Zones

- `Zones` — this system's Death Zone list.
- `Box / Sphere` — tools for creating a zone of the corresponding shape.

Death Zones do not need to be added to World.

Population — Monitoring

- `Alive Count Polls / s` — how often living population statistics and completion are checked, from 1 to 10 times per second.
- `Log All Boids Dead` — logs population completion to the Console.

Polling frequency does not change simulation speed.

Triggers

- `Swarm Triggers` — up to three unique zones assigned to the system.
- `Trigger Polls Per Sec` — how often trigger statistics are retrieved, from 1 to 10 times per second.
- `Create Box Trigger` — creates and assigns a trigger.

Count mode, threshold and events are configured on the zone component.

Visuals

- `Visual Effect` — the rendering component reference.
- `Select VFX Object` — selects its object for appearance editing.

This category does not provide a separate set of visual parameters. Those settings remain on the Visual Effect component.

World & Components — Parameter Reference

These components define the environment, spatial geometry, spawning and events.

Shape fields appear according to the selected type. Transform settings work differently across components: Spawn and zones account for scale, while Volume Primitive dimensions use world units.

Swarm World

- `Obstacles` — obstacles available to systems using this World.
- `Influences` — registered Attract, Repel and Flow zones.

A component's presence in the scene does not replace list registration. Multiple systems can share one World.

Swarm Spawn

- `Shape` — Point, Box or Sphere.
- `Box Size` — the full local dimensions of the spawn area.
- `Sphere Radius` — the local spawn radius.
- `Draw Gizmos` — displays the source shape and direction in Scene View.

The Transform supplies position, rotation and scale. Non-uniform Scale stretches a spherical spawn area.

Assign the source under Population → Source. It controls spawning, but does not keep the swarm nearby.

Swarm Spline Path

- `Loop` — closes the route. Requires at least three points; do not repeat the first point at the end.
- `Samples Per Span` — detail between control points, from 2 to 24. The complete GPU route is limited to 256 segments.
- `Control Points` — points in the object's local coordinates. An open route requires at least two.
- `Draw Path` — displays the route in Scene View.
- `Debug Corridor Radius` — the radius of the Path component's visual guide. It does not change SwarmSystem's working Spline Corridor Radius.

Editor buttons:

- `Add Point` — adds a point.
- `Remove Last` — removes the last point.
- `Reverse` — reverses point order and route direction.

Swarm Volume Primitive

Used as a part of a Composite Volume.

- `Shape` — Box, Sphere, Capsule or Cylinder.
- `Size` — the Box's full dimensions.
- `Radius` — the radius of a Sphere, Capsule or Cylinder.
- `Height` — the full height of a Capsule or Cylinder. A Capsule cannot be shorter than its diameter.

Position and rotation come from the Transform. Dimensions use world units and are not multiplied by Scale.

Assign the part to the system's **Primitive Parts** list.

Swarm Obstacle Zone

Obstacle

- `Mobility` — Static or Dynamic. Selects how obstacle data is prepared and updated. Use Dynamic for moving obstacles; Static does not lock the Transform in place.

Shape Source

- `Use Attached Collider` — uses a supported Collider's geometry instead of manual dimensions.
- `Source Collider` — a BoxCollider, SphereCollider or CapsuleCollider.

The Collider supplies a shape rather than Rigidbody collision solving. MeshCollider does not create exact collision against the visible mesh.

Manual Shape

- `Shape` — Box, Sphere, Capsule or Cylinder.
- `Center` — the shape's local centre offset.
- `Size` — the Box's full local dimensions.
- `Radius` — the selected rounded shape's radius.
- `Height` — the full height of a Capsule or Cylinder.
- `Axis` — the Capsule or Cylinder's local axis: X, Y or Z.

A Capsule's effective height is at least its diameter.

Manual dimensions account for Transform Scale. Sphere uses the largest axis scale. Capsule and Cylinder use the scale along their selected axis and the largest perpendicular scale.

Wall Settings

- `Open Axes` — open Box axes. Each selected axis removes both opposing sides.
- `Open Ends` — opens both Cylinder end caps.
- `Wall Margin` — extra clearance from active walls, added to Agent Radius.

Open Axes options:

- `None` — a closed Box.
- `X / Y / Z` — opens the corresponding pair of sides.
- `XY` — leaves the Z walls.
- `XZ` — leaves the Y walls.
- `YZ` — leaves the X walls.

The remaining walls work from both sides.

Debug

- `Draw Gizmo` — displays the obstacle shape. It does not disable its effect.

Register the obstacle in the assigned World's Obstacles list.

Swarm Influence Zone

Behaviour

- `Type` — Attract, Repel or Flow.

Shape

- `Shape` — Sphere, Box, Capsule or Cylinder.
- `Center` — the local centre offset.
- `Radius` — the selected shape's radius.
- `Size` — the Box's full dimensions.

- `Height` — the full height of a Capsule or Cylinder.
- `Axis` — the Capsule or Cylinder's local axis.

The shape accounts for Transform Scale. A Sphere with Radius = 0 can serve as a point Attract or Repel. Flow requires a non-zero area.

Attract and Repel

- `Inner Distance` — the distance from the shape's surface where falloff reaches full weight. For Attract, reaching this distance ends the current interest.
- `Outer Distance` — the outer influence limit. It must exceed Inner Distance.
- `Strength` — influence strength. Zero disables its force contribution.
- `Falloff` — Constant, Linear or Smooth.

Falloff controls weakening from Inner to Outer:

- `Constant` — a constant weight.
- `Linear` — linear weakening.
- `Smooth` — smoothly blended weakening.

Distances are measured from the shape's surface. Inside the shape, the distance is treated as zero.

Attract Only

- `Attract Release Distance` — the re-entry eligibility distance after reaching Inner Distance. It lies between Inner and Outer.
- `Attract Interest Chance` — the chance of taking an interest in an available zone, from 0 to 1.

Release does not set the outer attraction range. It determines how far a boid must move away after completing its interest before the zone can be selected again.

Interest Chance is not a fixed portion of the swarm. Rejection is remembered until the boid moves beyond Outer Distance.

Flow

- `Strength` — flow strength.
- `Edge Falloff Distance` — the width of the inner weakening band near the boundary.
- `Falloff` — the transition within that band.
- `Turbulence` — enables noise-driven changes to the flow.
- `Strength` under Turbulence — the amplitude of noise deviations.
- `Scale` — the noise's spatial frequency. Higher values produce finer spatial changes.
- `Speed` — how quickly the noise changes over time.

Flow acts inside its shape along the object's local **+Z** axis. A zero Edge Falloff gives an abrupt transition at the boundary.

Inner, Outer and Release do not define Flow's working area.

Debug

- `Draw Gizmo` — displays the shape and working ranges. It does not disable the influence.

Register the zone in the assigned World's Influences list.

Swarm Death Zone

- `Shape` — Box or Sphere.
- `Box Size` — the Box's full local dimensions.
- `Sphere Radius` — the Sphere's local radius.
- `Override Secondary VFX` — overrides whether On Die is permitted for deaths in this zone.
- `Secondary VFX` — allows or suppresses On Die when Override is enabled.
- `Draw Gizmo` — displays the death area.

Allowing Secondary VFX does not enable a globally disabled On Die mode. The system's overall Emission Mode still applies.

The shape follows its Transform. Sphere scale is determined by the largest axis.

Assign the zone to SwarmSystem's Death Zones list.

Swarm Trigger Zone

- `Size` — the Box's full local dimensions, including Transform scale.
- `Count Mode` — Unique Entered or Currently Inside.
- `Threshold Percent` — the threshold, from 1 to 100% of the GPU snapshot's living population.
- `Trigger Once` — fires once until rearmed.
- `Log Events` — logs event messages to the Console.
- `Draw Gizmo` — displays the area.
- `On Threshold Reached` — the threshold UnityEvent.

Unique Entered counts distinct living visitors since Rearm. **Currently Inside** counts living boids inside at the time of counting.

With Trigger Once disabled, Currently Inside rearms after falling below the threshold. Unique Entered requires an explicit Rearm for a fresh count.

Polling frequency is set in SwarmSystem. Assign the zone to that system's Triggers list.

Fish VFX — Parameter Reference

These settings are located on the **Visual Effect** component using **Vfx_Fish_Anim**.

The graph bends the body sideways along local X. A prepared mesh uses +Z forward, +Y up and a **Vertex Color.R** mask.

Primary VFX

- `BaseSize` — the base visual size of each boid. It does not change Agent Radius or spatial dimensions.
- `ScaleVariation` — the minimum and maximum individual scale multipliers. It does not affect speed variation.
- `RollCorrectionRadPerSec` — how quickly roll is gradually corrected towards world up, in radians per second. It does not control body bending.

Primary Shader — Colour and Surface

- `MeshTexture` — the base colour texture.
- `Tint` — a colour multiplier applied to the texture.
- `Normal` — the normal map.
- `NormalStrength` — normal map strength, not animation amplitude.
- `Metallic` — the surface's metallic appearance.
- `Smoothness` — surface smoothness and highlight response.
- `EmissionMap` — the emission texture.
- `EmissionColor` — the emission colour multiplier.
- `EmissionIntensity` — emission intensity.
- `AOMap` — the ambient occlusion texture.
- `AOStrength` — the AOMap contribution. Zero removes its influence.

Emission does not create a light source by itself. Its visible result depends on the materials and rendering configuration.

Fish Body Animation

- `HeadZ` — the local coordinate at the head end of the lengthwise range.
- `TailZ` — the local coordinate at the tail end.
- `AmpRatio` — lateral bending amplitude relative to the HeadZ–TailZ range length.
- `WaveCycles` — the number of spatial wave cycles along the body. Higher values fit more bends into the same length.
- `TailFocus` — changes the mask profile from its original value towards its square. Within the usual 0–1 range, this reduces intermediate values more strongly while preserving full influence in white areas.
- `PhaseOffset` — the animation phase offset. It is not a separate swimming speed setting.

The expected relationship is `HeadZ` > `TailZ`. Coordinates use the prepared mesh's local units rather than a normalised 0–1 range.

First fit HeadZ/TailZ to the model, then adjust AmpRatio and WaveCycles. TailFocus does not detect the tail automatically; its effect depends on the prepared mask.

Scripted

- `Mesh` — the main output mesh. SwarmSystem overrides it when Source Mesh is assigned.
- `BoidCount` — the system-controlled number of visual slots, not the living population count.
- `BoidBuffer` — the position and velocity buffer assigned by SwarmSystem.
- `SwarmLifecycleBuffer` — the lifecycle buffer assigned by SwarmSystem.
- `SecondaryContinuousRate` — the system-controlled Continuous rate.

Do not assign buffers manually or use BoidCount to change the population. Set Capacity in the controller.

Secondary VFX

`SecondaryDieCount` and all **SC_ / SD_** parameters are covered in **Secondary VFX**. Their roles are the same across the supplied graphs.

Bird / Wing VFX — Parameter Reference

This chapter covers **Vfx_Bird_Anim_Simple** and **Vfx_Bird_Anim_Expensive**.

Both expose the same user parameters but calculate deformation differently. Simple uses sequential rotations; Expensive builds a more smoothly curved wing centreline.

Check the result on your model when switching variants. Identical values do not necessarily produce identical shapes.

Primary VFX

- `BaseSize` — the base visual size.
- `ScaleVariation` — the range of individual scale multipliers.
- `RollCorrectionRadPerSec` — roll correction speed towards world up, in radians per second. This is not wingbeat frequency.

These settings do not adjust Agent Radius or zone geometry.

Primary Shader — Colour and Surface

- `MeshTexture` — the base colour texture.
- `Tint` — the colour multiplier.
- `Normal` — the normal map.
- `NormalStrength` — normal map strength.
- `Metallic` — the surface's metallic appearance.
- `Smoothness` — surface smoothness.
- `EmissionMap` — the emission texture.
- `EmissionColor` — the emission colour multiplier.
- `EmissionIntensity` — emission intensity.
- `AOMap` — the ambient occlusion texture.
- `AOStrength` — its contribution.

Frequency and Animation Influence

- `WingFreqHz` — the base wingbeat frequency in cycles per second.
- `WingFreqVariation` — individual frequency variation between boids.
- `WingMotionStrength` — the overall deformation influence, combined with Vertex Color.R. It is clamped to 0–1 during calculation.

WingMotionStrength = 0 removes wing deformation without stopping boid movement.

Main Flap

- `FlapUpDeg` — the amplitude of the upward phase, in degrees.
- `FlapDownDeg` — the amplitude of the downward phase, in degrees.
- `FlapRestDeg` — the base wing angle around which flapping occurs.

Different Up/Down values allow different amplitudes for the two phases.

Shoulder and Elbow

- `ShoulderPivot` — the wing root position in mesh-local coordinates.
- `ShoulderBlend` — the width over which deformation gradually begins from the shoulder along the lateral axis.
- `ElbowPivot` — the position of the additional bend and the start of the outer wing section.
- `ElbowBlend` — the transition width for the outer section's influence.

Coordinates use the prepared model's local units. The positive X coordinate is automatically mirrored for the other wing.

These fields do not use a normalised 0–1 range. Recheck Pivot and Blend when changing the model or its source dimensions.

Additional Motion

- `FoldDeg` — the additional outer-wing bend amplitude, in degrees.
- `FoldPhaseOffset` — the bend's phase offset relative to the main flap, in radians.
- `SweepDeg` — the amplitude of forward-and-back wing motion, in degrees.
- `SweepPhaseOffset` — the sweep's phase offset, in radians.
- `TwistDeg` — the outer wing's twisting amplitude, in degrees.

Start with the main flap and pivots, then gradually add Fold, Sweep and Twist. Large values for every influence at once can heavily distort unsuitable geometry.

Model Requirements

The nose points along +Z, up is +Y, and the wings extend along ±X.

The red vertex colour channel controls each vertex's participation in animation. The body and stationary areas are usually black; moving wing areas are light.

Procedural deformation does not automatically create landing, takeoff or leg animation.

Scripted

- `Mesh` — the main output mesh, overridden by the system when Source Mesh is assigned.
- `BoidCount` — the number of visual slots.
- `BoidBuffer` — positions and velocities.
- `SwarmLifecycleBuffer` — lifecycle states.
- `SecondaryContinuousRate` — the system-controlled Continuous rate.

SwarmSystem assigns the count and buffers. Do not edit them as appearance settings.

Secondary VFX

`SecondaryDieCount` and **SC_ / SD_** parameters are covered in **Secondary VFX**.

Non-Deforming Mesh VFX — Parameter Reference

Vfx_Mesh renders an arbitrary mesh without specialised fish or wing deformation. The shared simulation determines its position and orientation.

Use it for objects that need boid movement and rotation without changing shape.

Primary VFX

- `BaseSize` — the base visual size.
- `ScaleVariation` — the minimum and maximum individual scale multipliers.
- `RollCorrectionRadPerSec` — roll correction speed towards world up, in radians per second.

Size does not change Agent Radius, Volume or obstacles.

Primary Shader

- `MeshTexture` — the base colour texture.
- `Tint` — the colour multiplier.
- `Normal` — the normal map.
- `NormalStrength` — normal map strength.
- `Metallic` — the surface's metallic appearance.
- `Smoothness` — surface smoothness.
- `EmissionMap` — the emission texture.
- `EmissionColor` — the emission colour multiplier.
- `EmissionIntensity` — emission intensity.
- `AOMap` — the ambient occlusion texture.
- `AOStrength` — the AOMap contribution.

Scripted

- `Mesh` — the rendered geometry. The system's mesh is used when Source Mesh is assigned.



- `BoidCount` — the number of visual slots assigned by the controller.
- `BoidBuffer` — positions and velocities from the simulation.
- `SwarmLifecycleBuffer` — lifecycle states.
- `SecondaryContinuousRate` — the rate set by the system.

Buffers and slot count require no manual setup.

Use Forward/Up in Setup to correct orientation. Assigning a skeletal model does not run its Animator.

Secondary VFX

`SecondaryDieCount` and **SC_ / SD_** parameters are covered in **Secondary VFX**.

Quad VFX — Parameter Reference

Vfx_Quad renders textured quads instead of meshes. Their movement uses the same simulation as the other VFX variants.

The graph does not use Source Mesh and has no Fish Body Bending or Wing Flapping settings.

Primary VFX

- `Texture` — the main quad texture.
- `Metallic` — the Lit surface's metallic appearance.
- `Smoothness` — Lit surface smoothness.
- `BaseSize` — the base visual size.
- `ScaleVariation` — the range of individual scale multipliers.
- `RollCorrectionRadPerSec` — roll correction speed towards world up, in radians per second.

This graph does not expose the full mesh material set. Separate main-effect Tint, Normal, EmissionMap and AOMap fields are absent.

Quad size does not adjust Agent Radius. Large quads can overlap visually even when boid centres are separated.

Scripted

- `BoidCount` — the number of visual slots.
- `BoidBuffer` — the position and velocity buffer.
- `SwarmLifecycleBuffer` — the lifecycle buffer.
- `SecondaryContinuousRate` — the system-controlled Continuous rate.

SwarmSystem assigns these values. Use the controller's Capacity to change the population.

Secondary VFX

`SecondaryDieCount` and **SC_ / SD_** parameters are covered in **Secondary VFX**. Secondary branches are configured independently of the main quad's texture and size.

Thank You!

I hope this asset saves you time, helps you avoid some of the usual headaches of creating ambient flocks and swarms, and becomes genuinely useful in your project.

I've tried to make both the system and its documentation as practical and transparent as possible. If you have any questions, problems, or feedback, feel free to contact me.

Thank you for your support, and good luck with your project!


SKULMOVSKI
STUDIO