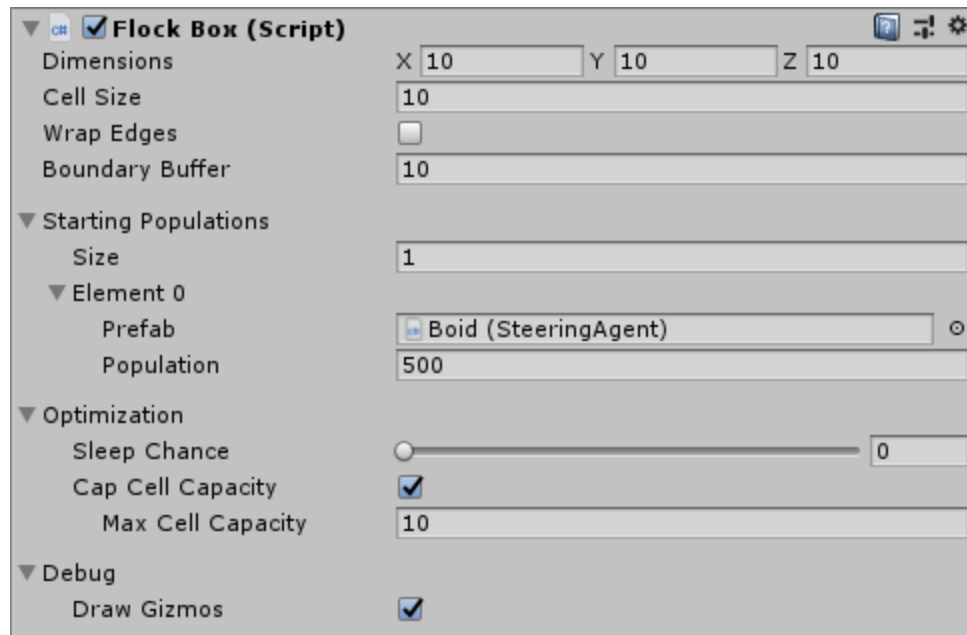# FLOCK BOX MANUAL

Version 2.1

QUICK START

Refer to any of the Demo scenes for an example of how to set up a FlockBox. If you're starting from scratch there are 3 primary components you'll need:

1. SteeringAgent prefab
   a. Add a SteeringAgent component to an empty GameObject and save it as a prefab.
   b. Add some sort of visual representation. You can easily add a primitive shape with *Right click > 3D Object > Cube*
2. BehaviorSettings asset
   a. Use *Create > Asset > BehaviorSettings* to create a new BehaviorSettings asset in the project. Add any desired behaviors (Alignment + Cohesion + Separation is the classic Boid formula).
   b. Drag and drop the settings onto the SteeringAgent prefab's activeSettings slot.
3. FlockBox component
   a. Add a FlockBox component to an empty GameObject in the scene.
   b. Create a new slot in Starting Populations and add the SteeringAgent prefab.

Press Play and watch them go! For more information on Steering Behaviors, see the References section

FLOCK BOXES

Dimensions	Number of cells in each dimension. The world space extents of the FlockBox will be dimensions * cellSize. Note: you can set one dimension to 0 to create a 2D FlockBox.
Cell Size	FlockBox uses spatial hashing for optimization, meaning that the box is divided into cells that each keep track of the agents they contain for faster lookup. Cells that are too large or small may cause performance issues. The ideal cell size is usually going to be around the max perception distance of the most common agents.
Wrap Edges	Defines whether or not agents can travel through the boundary and wrap to the other side of the box. If false, agents will always be constrained to positions inside the box.
Boundary Buffer	Defines a "buffer zone" which agents will attempt to steer out of when wrapEdges is set to false. (see Containment Steering Behavior)
Starting Populations	Define populations of Agents that should be instantiated in Start(). Their position and velocity will be random. Note: you can also drag individual Agents into the Scene as children of

|	
	the FlockBox and they will automatically be detected (see Avoidance Demo).
Sleep Chance	A percentage value that determines how often Steering Agents skip updating their velocity. A value of .5 means that roughly half the Steering Agents will not update on any given frame. This can be great for performance if the reduced fidelity isn't a problem.
Cap Cell Capacity	Limits the number of Agents that will be recorded in each cell. When Agents are evenly distributed, they can make use of spatial hashing to limit the number of neighbors they need to calculate with. If Steering Agents become bunched together, spatial hashing will no longer provide a benefit. In case of overcrowding, this cap will limit the total number of neighbors that each individual is aware of to prevent the framerate from getting dragged down.
Display Gizmos	Draw debug information in the Scene View. The full extent of the FlockBox and boundary buffer in world space are drawn as wireframe boxes. At runtime cells that have at least one occupant will be shaded in.

AGENTS

An Agent is any entity that is involved in the Flock Box simulation. A basic Agent will not move.
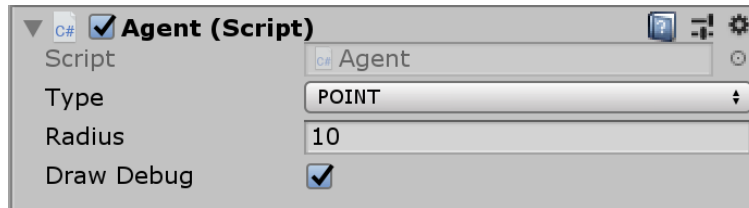
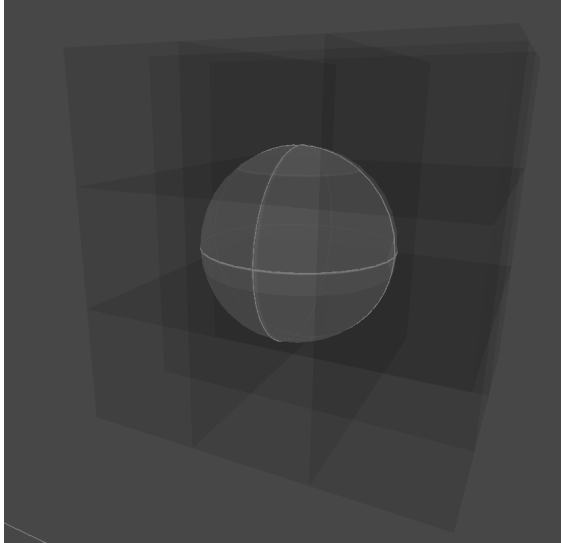| Type | POINT behaves similarly to SPHERE, but can update much faster when moved. The trade-off is that they offer lower fidelity. In general, SPHERE should generally only be used for large, immobile Agents, while POINT should be used for small, mobile Agents (like Boids).

POINT
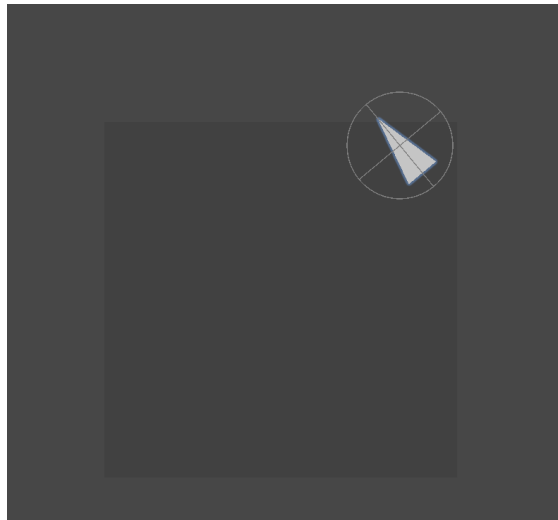
Agent will only occupy a single cell based on their center point, even if their radius extends into neighboring cells. |
|---|---|

	SPHERE This Agent will occupy all cells within a bounding box containing a sphere.
Draw Debug	Draw a Gizmo representing the Agent's Shape

Steering Agents

Steering Agent is derived from Agent, but adds the ability to update its position and velocity based on its surroundings.
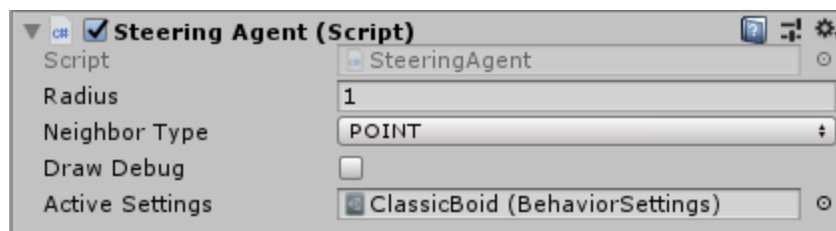
|	
Active Settings	Reference to a Behavior Settings asset.

Terrain Agents

A Terrain-friendly subclass of SteeringAgent. Will use raycasting to stick to the surface of level geometry (see Terrain_Demo).

Physics Agents

A Physics-friendly subclass of SteeringAgent. Will move by setting the velocity of a Rigidbody instead of setting the position of a transform to realistically collide with environmental geometry (see PhysicsAgent_Demo).

BEHAVIOR SETTINGS

Create a new BehaviorSettings asset with Create > Asset > BehaviorSettings.
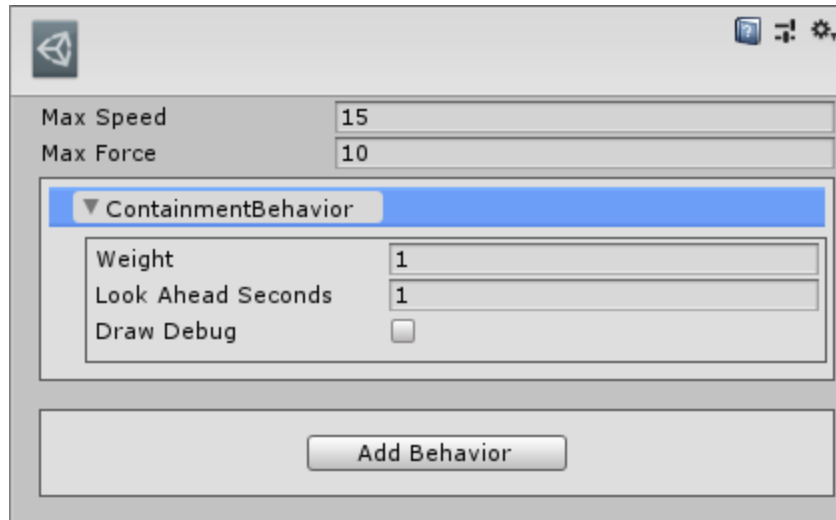
|	
Max Speed	The top speed that a steering agent can reach.
Max Force	The maximum force that can be applied to the steering agent. Tune this value to control how fast the agent accelerates.

Steering Behaviors
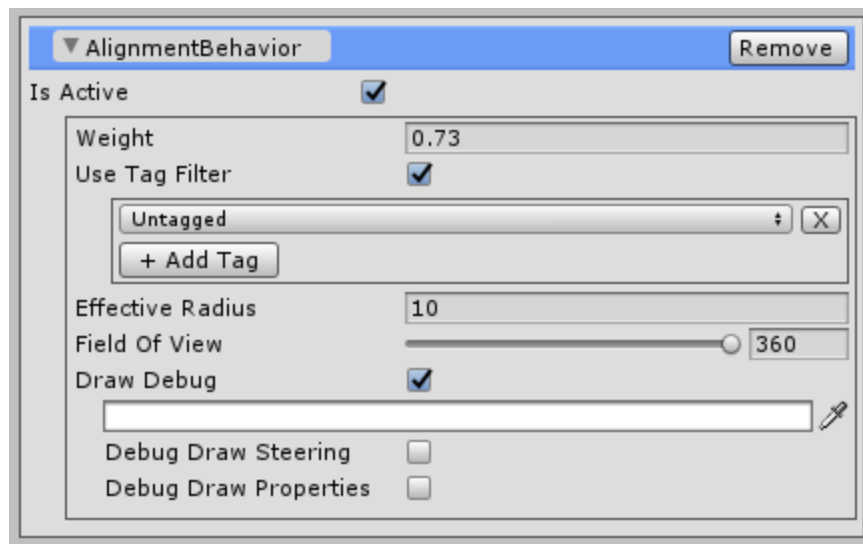
Weight	Adjust this value to change the relative influence of each steering behavior.

Tag Filter	Uses Unity's built-in Tag system. A tag filter should be used if a behavior's steering calculation should only take certain agents into consideration and ignore all others. For example: if agents with an Avoidance behavior should only avoid obstacles and not each other, tag the obstacle as "Obstacle" and have the Avoidance behavior filter for that tag. (See Avoidance example). This may not be applicable for all Steering Behaviors (eg. Wander, Containment).
Draw Debug	Enables debug visualization of the steering behavior.
Debug Color	Color to use when drawing debug information for this behavior.
Debug Draw Steering	Draws a line representing the behavior's steering. Will only appear in Play mode.
Debug Draw Properties	Draws gizmos representing the behavior's various options. Text labels will accompany the gizmos when not in Play mode (to avoid visual clutter).

PACKAGED BEHAVIORS

Alignment

Steer towards the average forward direction of surrounding agents.
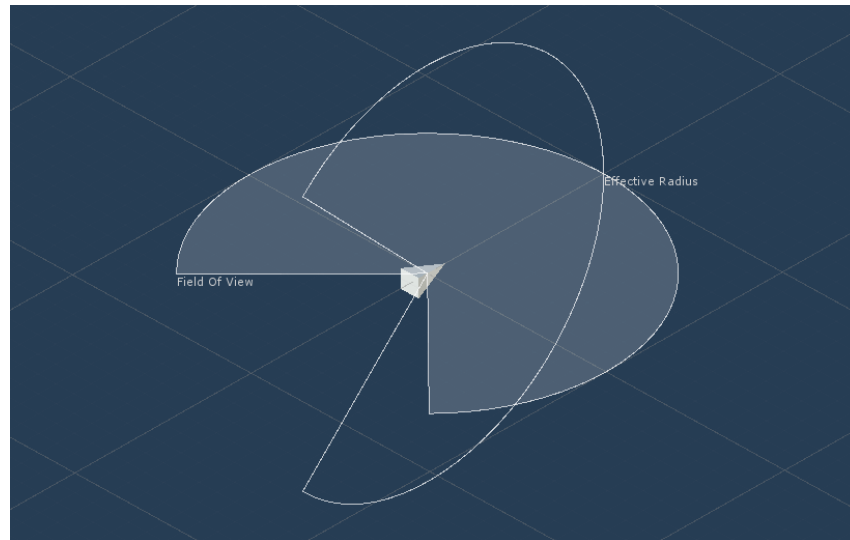
Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.

Separation

Steer away from nearby agents. Weighted by distance so that the closer other agents get, the greater the separation force becomes.

Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.

Cohesion

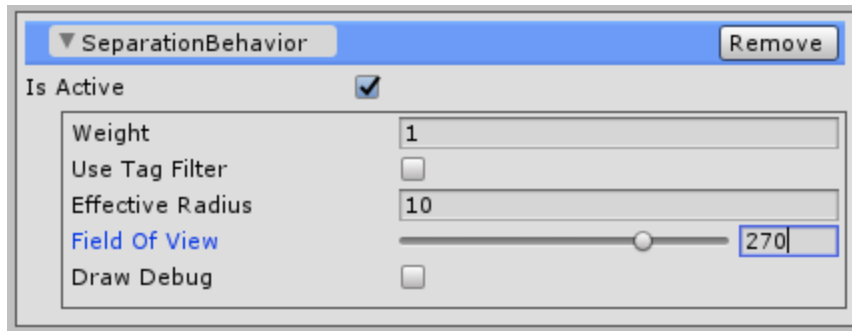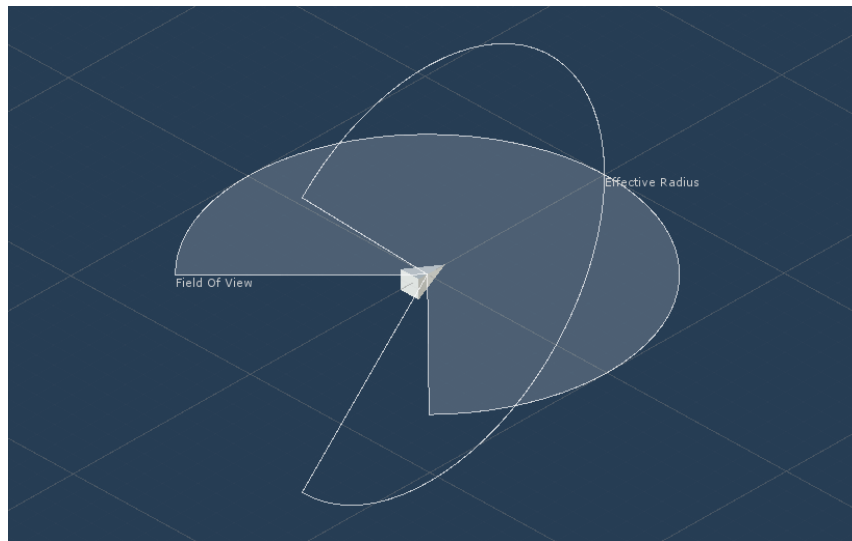Steer towards the midpoint of surrounding agents.

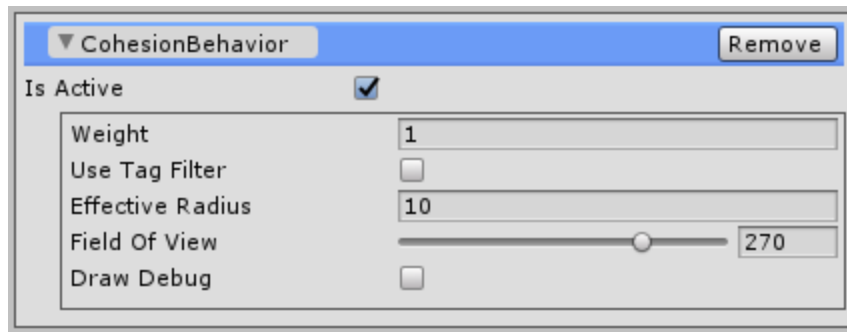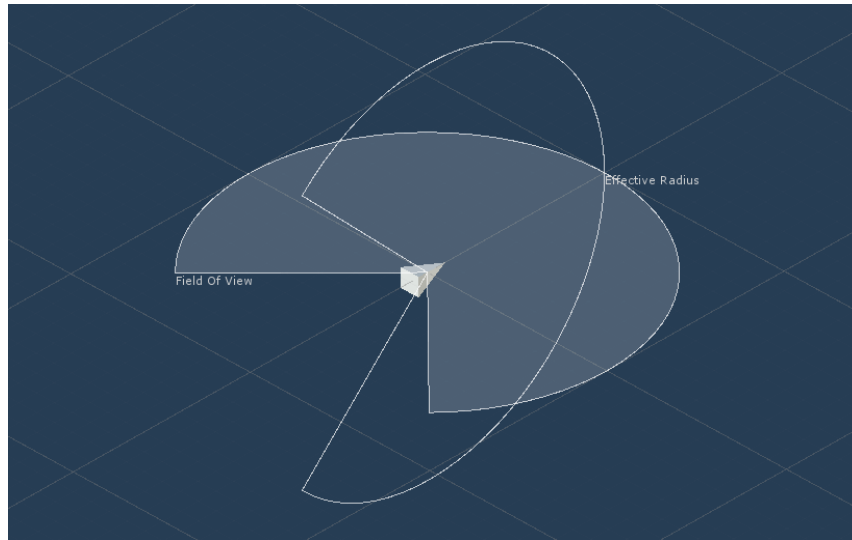| *Effective Radius*	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.

Containment

Containment is added automatically to each BehaviorSettings, and is only used when inside a FlockBox with wrapEdges disabled. This behavior will steer the agent out of a buffer zone defined by FlockBox.boundaryBuffer.
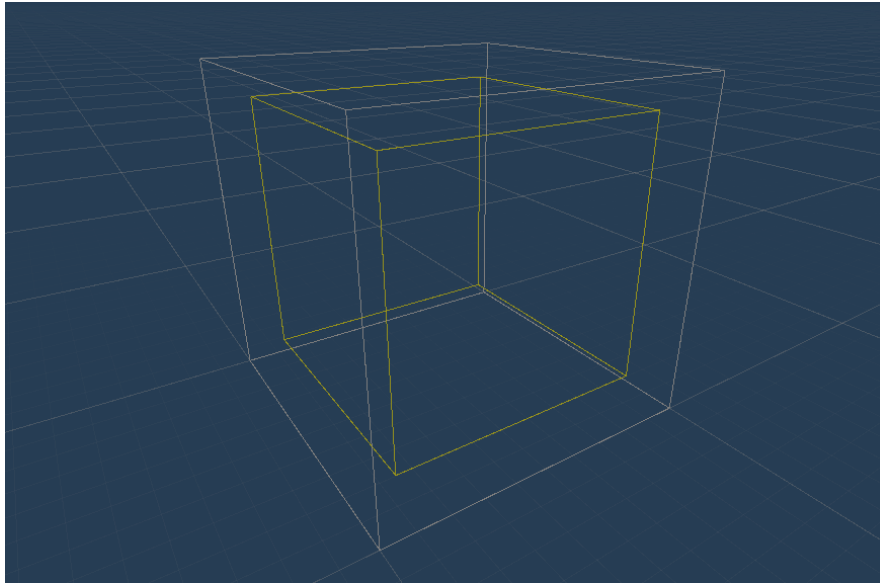
Look Ahead Seconds	How many seconds before entering the boundary buffer area that this behavior should begin to steer away from the edge of the Flock Box.

Avoidance

Steer to avoid an upcoming obstacle. Will stop applying force when the agent's path will not intersect, or is tangential to the surface of the obstacle. Obstacles must be Agents.

Look Ahead Seconds	How many seconds before collision that this behavior will begin to steer away from the expected impact.
Clearance	Extra space to maintain between the SteeringAgent and Obstacles.

ColliderAvoidance

Steer to avoid upcoming Colliders. Less performant than Avoidance, but more useful for complex environments.

Look Ahead Seconds	How many seconds before collision that this behavior will begin to steer away from the expected impact.
Clearance	Extra space to maintain between the SteeringAgent and Obstacles.

Flee

Steer away from the midpoint of other agents.

Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.
Global Tag Search	Optimization option. Search for neighbors globally by Tag instead of searching all neighborhood cells individually. Agents will still need to be within *Effective Radius* to influence steering. Most useful for situations where there will either be a small number of targets, or they will need to be perceptible from far away.

Seek

Steer towards a stationary target.

Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.
Global Tag Search	Optimization option. Search for neighbors globally by Tag instead of searching all neighborhood cells individually. Agents will still need to be within **Effective Radius** to influence steering. Most useful for situations where there will either be a small number of targets, or they will need to be perceptible from far away.

Pursuit

Steer towards an anticipated interception point with a moving target.
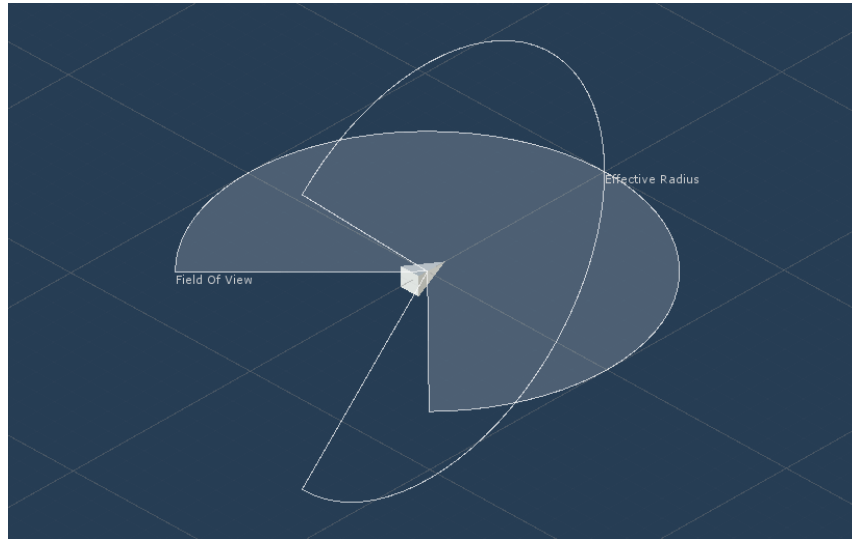

Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.
Global Tag Search	Optimization option. Search for neighbors globally by Tag instead of searching all neighborhood cells individually. Agents will still need to be within ***Effective Radius*** to influence steering. Most useful for situations where there will either be a small number of targets, or they will need to be perceptible from far away.

Arrival

Steer towards a stationary target, but apply braking force when approaching to come to a stop.
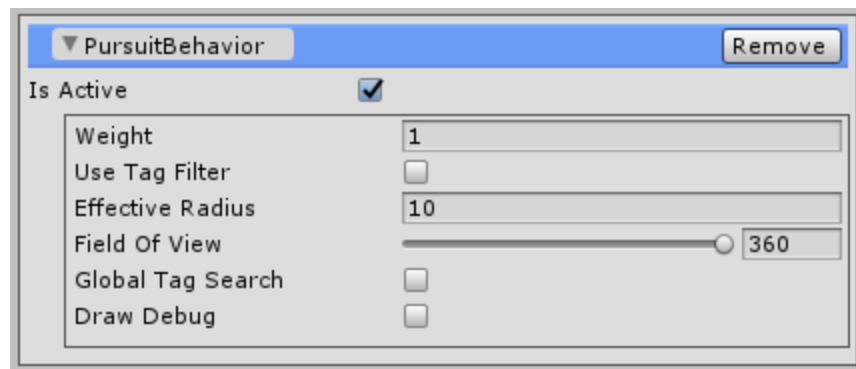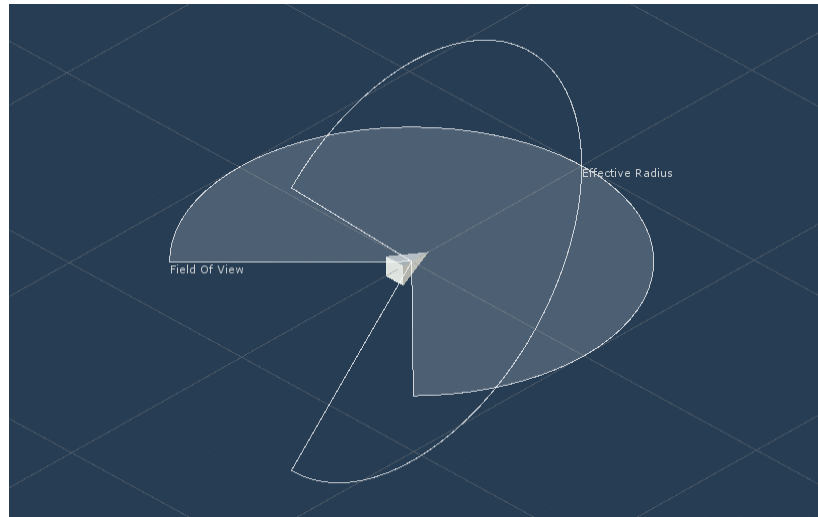

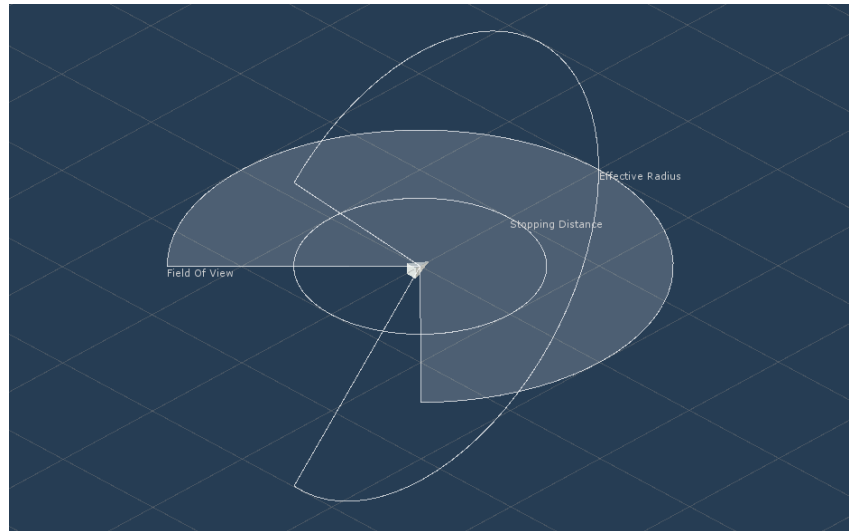| Effective Radius	Furthest distance at which other Agents will affect this behavior's steering calculation.
Field of View	Range in degrees within which other Agents will affect this behavior's steering calculation.
Global Tag Search	Optimization option. Search for neighbors globally by Tag instead of searching all neighborhood cells individually. Agents will still need to be within **Effective Radius** to influence steering. Most useful for situations where there will either be a small number of targets, or they will need to be perceptible from far away.
Stopping Distance	Distance at which braking force should be applied.

Wander

Apply steering force in a random direction. Calculated with Perlin noise.
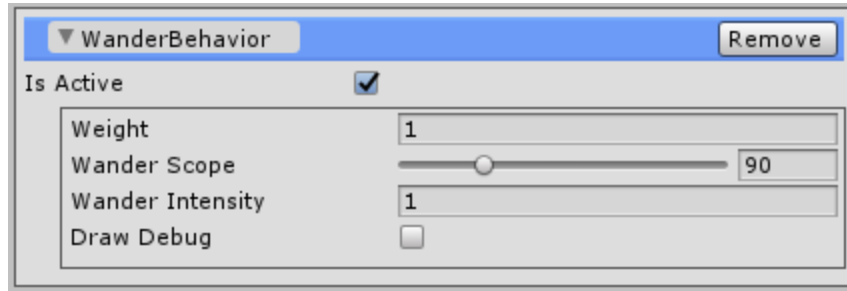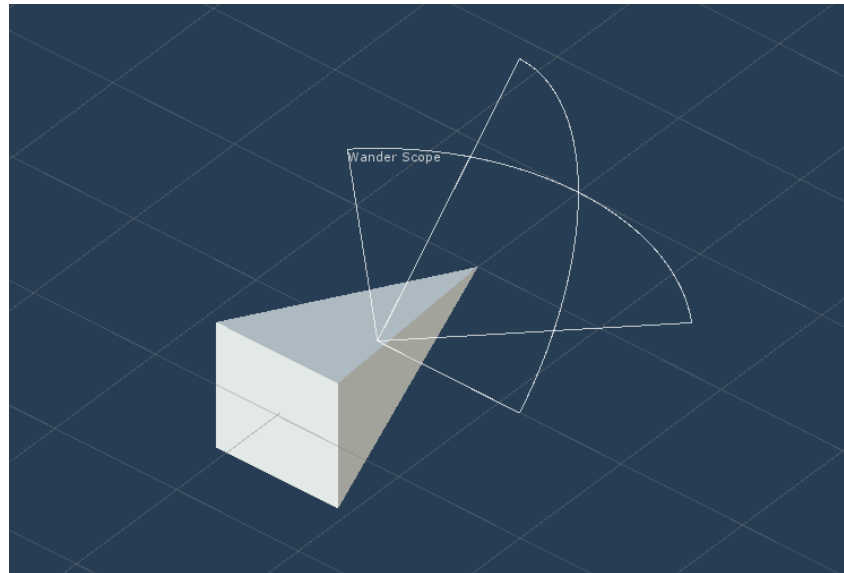
Wander Scope	Determines the wander force's maximum deviation in degrees from the SteeringAgent's current direction.
Wander Intensity	Determines the rate of change for the wander direction.

Leader Follow

Similar to Arrival. Steer toward a point that is behind a target Leader. If multiple potential leaders are found, the closest one will be chosen. Followers will prioritize steering out of the space ahead of the Leader to prevent them from getting in the Leader's way.
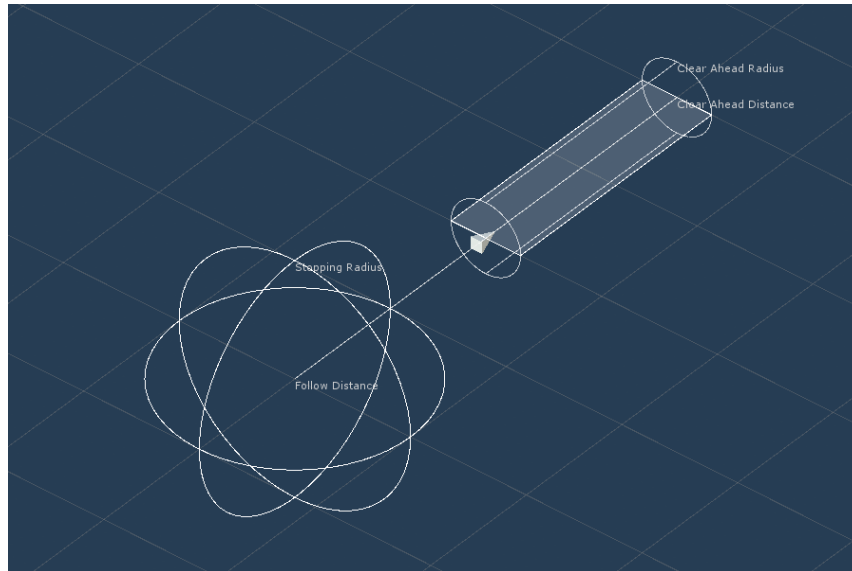
Follow Distance	Distance behind the Leader that Followers should target. Prevents Followers from crowding the Leader.
Stopping Radius	Distance at which braking force should be applied.
Clear Ahead Distance	Length of the cylindrical space ahead of the Leader that Followers will avoid.
Clear Ahead Radius	Radius of the cylindrical space ahead of the Leader that Followers will avoid.

Queue

Apply braking force when blocked by another Agent. Produces the effect of a crowd slowing down when trying to pass through a congested area.

Queue Distance	Determines the center of the search area for Agents blocking this one.
Queue Radius	Determines the radius to search for Agents blocking this one.

DOTS MODE

This feature is intended to be accessible to users who have no knowledge of DOTS (Data Oriented Tech Stack). However, if you're unfamiliar with Unity DOTS, I would recommend looking at Unity's documentation for an introduction:

> [DOTS Information Page](#)
>
> [Unity ECS Manual](#)

The benefit of using DOTS mode is a massive performance increase. It could potentially increase the number of active agents by an order of magnitude. Unfortunately it comes with some drawbacks.

Limitations
- Not all features of the single-threaded version of Flock Box have been ported over. This includes Agent variations like ExternalAgent, Debug visualization gizmos, and several Steering Behaviors.
- Not compatible with MonoBehaviors. Any additional custom behavior needed for the Entities would have to be re-implemented with DOTS.
- No way to swap BehaviorSettings on an Entity at runtime.

How to Use DOTS
Flock Boxes now have a "Use DOTS" toggle. If enabled, it will attempt to convert the Agent prefabs in Starting Populations into Entities and spawn them as such instead of GameObjects.
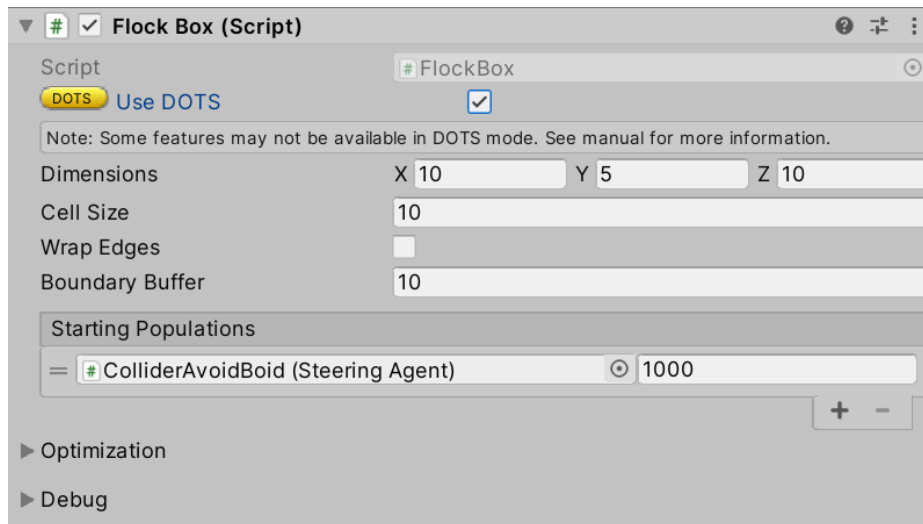
Unity automatically handles the conversion of many built-in components (like MeshRenderer), but if your Agent prefab has custom components that you would like to carry over into DOTS mode, you'll need to re-implement its functionality with [Systems](#) and have the original MonoBehavior implement *IConvertGameObjectToEntity* from [Unity's Conversion Workflow](#). See SteeringAgent.cs for an example implementation of the *IConvertGameObjectToEntity* interface.

You won't see Entities listed in the Hierarchy window. To see their data you'll need to open the Entities window under *Window>DOTS>Entities.*

Just like the single-threaded system, you'll be able to modify their behavior in real time by changing values on their respective BehaviorSettings. You'll notice a yellow DOTS badge on some of the behaviors.
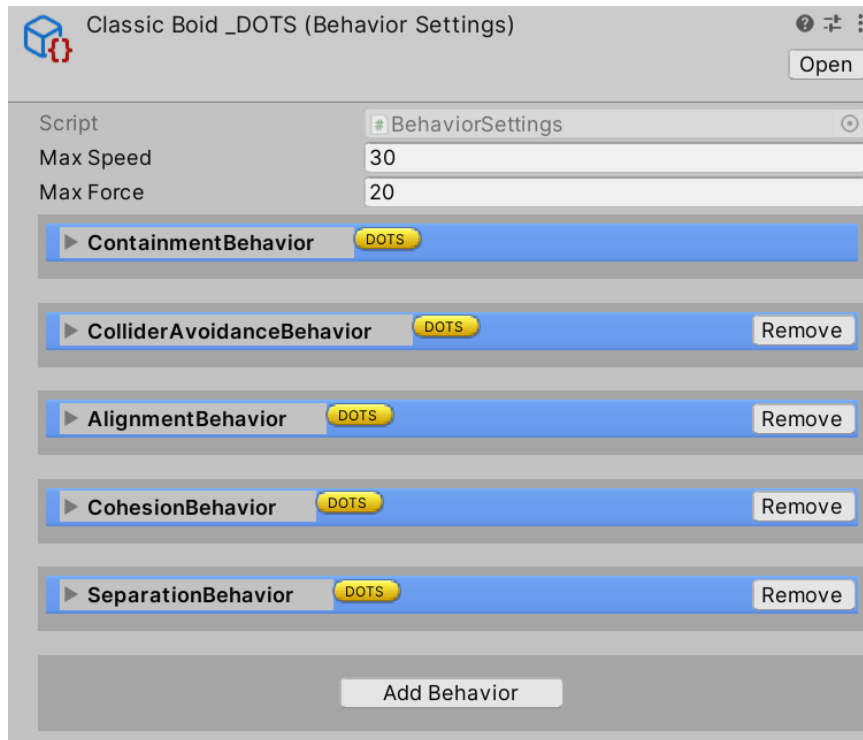
These have been implemented as Systems for DOTS. Any behavior without this badge will have no effect on Entities when "Use DOTS" is enabled. Not all behaviors have DOTS implementations as of version 2.0. More will be implemented over time.

If you want to use ColliderAvoidance with DOTS, you'll also need to set up your environment to work with DOTS Physics. See the Unity Physics Manual and the DOTS_ColliderAvoidanceDemo scene for examples. Note how each part of the environment in the demo scene has *ConvertToEntity* and *PhysicsShape* components.

To create a custom behavior, write a class that inherits from SteeringBehavior (or any subclass like RadialSteeringBehavior or ForecastSteeringBehavior) and override GetSteeringBehaviorVector().

public override void GetSteeringBehaviorVector(out Vector3 steer, SteeringAgent mine, SurroundingsContainer surroundings)

{

    //calculate steering and set Vector3 steer.

}

Getting Information from Surroundings

Every behavior needs to make sure that enough information is being collected about the Agent's surroundings to calculate the proper steering vector. To accomplish this in a custom behavior, override AddPerception and use the following functions on the SurroundingsContainer that gets passed in.

public override void AddPerception(SteeringAgent agent, SurroundingsContainer surroundings)

{

    // surroundings.SetMinPerceptionRadius(float radius);

}

SetMinLookAheadSeconds(float seconds)	Add all Agents this many seconds in front of mine in a straight line.
SetMinPerceptionRadius(float radius)	Add all Agents within this radius around mine.
AddGlobalSearchTag(string tag)	Add all Agents with this Tag (anywhere in the Flock Box)
AddPerceptionSphere(float radius, Vector3 position)	Add all Agents within a sphere at a point in space.

Any Agents that were discovered will be added to SurroundingsContainer.allAgents, and be available for steering calculations when the SurroundingsContainer object gets passed into GetSteeringBehaviorVector(Vector3, SurroundingsContainer).

Agent Instance Data

It's possible for agents to have custom instance data without extending the Agent class through Agent.SetAgentProperty<T>() and Agent.GetAgentProperty<T>().

This is used in SeekBehavior to keep track of a particular seek target and prevent constant target switching, which leads to more natural behavior.

Adding Custom Behaviors to a Behavior Settings

All non-abstract classes derived from SteeringBehavior will appear in the "Add Behavior" list automatically using Reflection.

Editing Behavior Values in the Inspector

Any properties that you want to be able to edit in the inspector should either be public or marked with the [System.Serializable] attribute.

Further editor customization is not supported because PropertyDrawers cannot use polymorphism when the inspected objects are stored in a collection as their base class (such as how behaviors are stored in a SteeringBehavior array in BehaviorSettings). If you were to write a custom PropertyDrawer for a behavior implementation, the PropertyDrawer for SteeringBehavior would be shown instead.

Property Gizmos

To visualize a custom behavior's properties using Gizmos, override DrawPerceptionGizmos inside of UNITY_EDITOR compile tags.

```
#if UNITY_EDITOR

 public override void DrawPropertyGizmos(SteeringAgent agent, bool drawLabels)

{

        // your Gizmos or Handles here

}

#endif
```

FAQ

My Steering Agents seem "twitchy".

The first step should be adjusting the relative weights of the behaviors to try to achieve a smoother result. Twitchiness can be a sign that one behavior has too much influence. If that's not getting you anywhere, try adding a *TwitchDampening* component to your agent's root model object. See the ColliderAvoidBoid prefab for an example setup.

I have an Agent that needs to be able to move outside of the Flock Box.

Replace the Agent component with ExternalAgent. Be sure to either drop a FlockBox into the AutoJoinFlockBox field or add it to a FlockBox at runtime by calling *ExternalAgent.Spawn(FlockBox fb);* See the Player prefab and PlayerControl_Demo scene for reference.

Can an Agent be player-controlled?

Yes. See the Player prefab and PlayerControl_Demo scene for reference.

How do I add or remove Agents at runtime?

You can instantiate your Agents anywhere, just be sure to call *Agent.Spawn(FlockBox fb);* to assign them to a FlockBox. See the StressTest_UI_Demo scene and PopulationControlUIExample.cs for reference.

CHANGE LOG

Version 2.1 5/18/2021
- Added steering debug lines support to DOTS behaviors
- Used scripting define FLOCKBOX_DOTS to support backwards compatibility to Unity 2018.4

Version 2.0 5/1/2021
- Added DOTS compatibility.
- Removed support for Line and Cylinder Shapes.
- Moved all classes into the CloudFine.Flockbox namespace. Cleaned up function and field names for clarity.
- Added FAQ section to the Manual
- Upgrade Project to Unity 2020.1.1f for the latest DOTS packages

Version 1.8 3/7/2021

- Added ExternalAgent. An Agent type that does not need to be parented to a FlockBox.
- Added PlayerControl_Demo
- Added SeekPostionBehavior. A Seek behavior that can be directed to an arbitrary point instead of an Agent.
- Fix bug with BehaviorSettings custom inspector in Unity 2020.1+

Version 1.7 4/20/2020
- Added QueueBehavior.cs, a behavior that applies brake forces to simulate the slowing of congested traffic.
- Added QueueFunnel_Demo scene as an example of Queue behavior.
- Added "Clearance" property to ColliderAvoidanceBehavior and AvoidanceBehavior.
- Added "GlobalTagSearch" property to some behaviors for optimization. Will allow behaviors to look up neighbors by tag instead of searching all neighborhood cells individually. Most useful for situations where there will be either a small number of targets, or they will need to be perceptible from far away.
- Added Debug visualizations for each behavior's properties.
- Added editor tooltips for behavior properties.
- Added TwitchDampening, a component that can be used to dampen twitchy SteeringAgent behavior.
- Added population control to UI_Demo scene. Click anywhere to spawn new agents.

Version 1.6 3/17/2020
- Added ColliderAvoidanceBehavior.cs, a behavior similar to Avoidance that uses Raycasting to steer away from colliders ahead.
- Added PhysicsAgent.cs, a SteeringAgent that supports rigidbody collisions.
- Replaced some Lists with HashSets to solve duplicate entries in neighbor lists.
- Suppressed unnecessary warnings.
- LeaderFollowBehavior has a "clear zone" ahead of the leader that followers will avoid.

Version 1.5 1/29/2020
- Added TerrainAgent.cs, a Terrain-friendly subclass of SteeringAgent.
- Added Terrain_Demo scene as an example of TerrainAgents.

Version 1.4 1/5/2020
- Added helper function to retrieve specific steering behaviors from a BehaviorSettings
- Added example scene for creating in-game UI for changing BehaviorSettings values

Version 1.3 12/12/2019
- Optimized memory allocation.
- Added Optimization option to cap the number of Agents recorded in each cell.

Version 1.2 11/9/2019
- Added LeaderFollow behavior
- Fixed bug preventing Pursuit behavior from functioning.

Version 1.1 11/2/2019
- Added CYLINDER shape
- Expanded support for LINE shape, including new fields exposed in Inspector
- Changed "NeighborType" to "Shape"

CONTACT

Send and comments or questions to:

marc@cloudfinegames.com

REFERENCES

[1] Reynolds, Craig. (1999). Steering Behaviors For Autonomous Characters.
https://www.red3d.com/cwr/steer/gdc99/

[2] Shiffman, Daniel. (2012). Nature of Code.
https://natureofcode.com/book/chapter-6-autonomous-agents/

[3] Bevilacqua, Fernando. (2013). Understanding Steering Behaviors.
https://gamedevelopment.tutsplus.com/series/understanding-steering-behaviors--gamedev-12732