UnitFormation

Table of Contents

Introduction

This package is a simple helper to calculate positions of the units. To start using it in your own code, first you must add "using TRavljen.UnitFormation" at the top of the C# file. From there you can start using the "FormationPositioner" class that will calculate the position based on the formation your provide. You can use of the existing ones that can be found in "Unit Formation/Scripts/Formations/" path.

Script Reference

FormationPositioner.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

FormationPositioner

```
public class FormationPositioner
```

Class responsible for providing unit positions in formation on a target position facing the respective angle. Use method 'GetAlignedPositions' when you are manually calculating facing angle of the formation. Use method 'GetPositions' when you wish for angle calculation to be done manually. It will be calculated based on magnitude of the position change and the angle from units center position to the new target position.

Methods

GetAlignedFormation

```
public static UnitFormationData GetAlignedFormation(
     int unitCount,
     IFormation formation,
     Vector3 targetPosition,
     Vector3 targetDirection)
```

Returns aligned unit formation positions with facing targetDirection passed.

unitCount: Amout of units in formation.
formation: Formation that units will position in.
targetPosition: Position of the formation.
targetDirection: Facing direction of the formation.

Returns: Returns aligned positions of the units in formation.

GetAlignedPositions

```
public static List<Vector3> GetAlignedPositions(
     int unitCount,
     IFormation formation,
     Vector3 targetPosition,
     float targetAngle)
```

Returns aligned units formation positions that are facing the passed angle.

unitCount: Amount of units in formation.
formation: Formation that units will position in.
targetPosition: Position of the formation.
targetAngle: Facing angle for the formation.

Returns: Returns aligned positions of the units in formation.

GetPositions

```
public static UnitFormationData GetPositions(
     List<Vector3> currentPositions,
     IFormation formation,
     Vector3 targetPosition,
     float rotationThreshold = 4.0f)
```

Finds new positions for the passed positions and the formation. If distance from current positions center is less than rotation threshold, units formation will not be rotated around the target. New rotation angle is calculated from center position of all current positions and the target positions.

currentPositions: Current unit positions.
formation: Formation used on units
targetPosition: Position to where the units will be moved.
rotationThreshold: Threshold used to specify when the unit formation should be rotated around target position (pivot).

Returns: Returns list of the new unit positions and their new facing angle

UnitFormationData

```
[System.Serializable]
public struct UnitFormationData
```

Data structure that represents the units new formation positions and angles.

Constructors

UnitFormationData

```
public  UnitFormationData(
     List<Vector3> unitPositions,
     float finalRotation)
```

Variables

UnitPositions

```
public List<Vector3> UnitPositions
```

Specifies the new positions that units can move to new formation.

FacingAngle

```
public float FacingAngle
```

Specifies the units facing angle (loot at direction) for the new position.

FacingEuler

```
public readonly Vector3 FacingEuler
```

Returns euler vector for facing direction. To convert into Quaternion use Quaternion.Euler(Vector3).

FormationUnit.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

AFormationUnit

```
public abstract class AFormationUnit
```

FacingRotationEnabled

```
[Tooltip("Specifies if rotating towards the facing angle is enabled.
" +
"Set this to 'false' if you wish to manually handle
synced " +
"rotation of units in rotation.")]
public bool FacingRotationEnabled
```

Properties

IsWithinStoppingDistance

```
public abstract bool IsWithinStoppingDistance
      { get; }
```

destination

```
protected abstract Vector3 destination
      { set; }
```

SetTargetDestination

```
public void SetTargetDestination(
      Vector3 newTargetDestination,
      float newFacingAngle)
```

FormationUnit

```
[RequireComponent(typeof(NavMeshAgent))]
public class FormationUnit
```

Unit component for moving to target position with Unity’s NavMesh system. It also faces the angle of the formation after it reaches its destination (if enabled).

Variables

agent

```
[SerializeField]
NavMeshAgent agent
```

IsWithinStoppingDistance

```
public override bool IsWithinStoppingDistance
```

Properties

destination

```
protected override Vector3 destination
      { set; }
```

IFormationUnit.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

IFormationUnit

```
public interface IFormationUnit
```

Interface used for communicating formation positions to units and their facing angles. There is also default implementation available which uses Unity’s NavMesh system, FormationUnit.

Properties

IsWithinStoppingDistance

```
public bool IsWithinStoppingDistance
     { get; }
```

Methods

SetTargetDestination

```
public void SetTargetDestination(
     Vector3 newTargetDestination,
     float newFacingAngle)
```

UnitFormation.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

UnitFormation

```
public class UnitFormation
```

Component responsible for managing units formations targets. This is done through IFormationUnit interface which allows any type of movement control to the destination target of each unit within a formation.

GroundPositioner

```
public IGroundPositioner GroundPositioner
```

HasUnits

```
public bool HasUnits
```

Returns true if there is more than 1 unit present.

CurrentFormation

```
public IFormation CurrentFormation
```

Returns current formation definition.

Units

```
public List<Transform> Units
```

List of units used for placing in formation.

Properties

FormationPositions

```
public UnitFormationData FormationPositions
     { get; }
```

Specifies current calculated positions for the unit formation.

NoiseEnabled

```
public bool NoiseEnabled
     { get; set; }
```

Specifies if a random noise is appleid ontop of formation positions.

Methods

SetUnitFormation

```
public void SetUnitFormation(
     IFormation formation)
```

Set new formation definition

ApplyCurrentUnitFormation

```
public void ApplyCurrentUnitFormation(
     UnitFormationData formationData)
```

Applies formation data to current units. Make sure the unit count matches!

formationData: New formation data

ApplyCurrentUnitFormation

```
public void ApplyCurrentUnitFormation(
     Vector3 position,
     Vector3 direction)
```

Calculates and applies new formation positions based on new position and direction.

position: New position
direction: New direction

ApplyCurrentUnitFormation

```
public void ApplyCurrentUnitFormation(
     Vector3 position)
```

Calculates and applies new formation positions based on new position. This method calculates direction from units group center to the new position of the formation.

position: New position

CalculatePositions

```
public UnitFormationData CalculatePositions(
     Vector3 position,
     Vector3 direction)
```

Calculate new unit formation group positions.

position: Formation position
direction: Formation direction

Returns: Returns calculated formation data for this unit formation group.

MovePositionOnGround

```
public Vector3 MovePositionOnGround(
     Vector3 position)
```

Finds the nearest valid ground and returns it. If there was no valid positions, then value returned is unchanged. It uses GroundPositioner to calculate the position.

position: Returns new position if tere is a valid one within maxGroundDistance

UnitFormationHelper.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

UnitFormationHelper

```
public static class UnitFormationHelper
```

Methods

TryMovePositionOnGround

```
public static bool TryMovePositionOnGround(
     Vector3 position,
     float maxDistance,
     out Vector3 groundPosition,
     LayerMask groundMask)
```

TryMovePositionOnGround

```
public static bool TryMovePositionOnGround(
     Vector3 position,
     float maxDistance,
     out Vector3 groundPosition,
     int areaMask = NavMesh.AllAreas)
```

Finds the nearest valid NavMesh area for the position.

position: Current position
maxDistance: Max distance for check
groundPosition: Ground position
areaMask: NavMesh area

Returns: Returns true if ground position was found

ApplyFormationCentering

```
public static void ApplyFormationCentering(
     List<Vector3> positions,
     float rowCount,
     float rowSpacing)
```

Applies offset to the Z axes on positions in order to move positions from pivot in front of formation, to pivot in center of the formation.

positions: Current positions, method will update the reference values.
rowCount: Row count produced with formation.
rowSpacing: Spacing between each row.

GetNoise

```
public static Vector3 GetNoise(
     float factor)
```

Generates random "noise" for the position. In reality takes random range in the offset, does not use actual Math noise methods.

factor: Factor for which the position can be offset.

Returns: Returns local offset for axes X and Z.

Formations

CircleFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

CircleFormation

```
[System.Serializable]
public struct CircleFormation
```

Formation that positions units within a circle with fill. This formation must be adjusted manually in parameters for each unit count specifically as it is completely controlled by the outerRadius and unitSpacing. These two parameters are in complete control how units are positioned.

Constructors

CircleFormation

```
public  CircleFormation(
     float outerRadius,
     float unitSpacing)
```

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

ComputedCircleFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

ComputedCircleFormation

```
[System.Serializable]
public struct ComputedCircleFormation
```

Formation that positions units within a circle with fill. Works similar to CircleFormation, but this formation attempts to get the best possible option of a filled circle for specified unit count. For this in most cases it will need to do iterations to find the best fit, there for you control maximum iterations with maxIterations.

Constructors

ComputedCircleFormation

```
public  ComputedCircleFormation(
     float unitSpacing)
```

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

GetPositionIteration

```
public bool GetPositionIteration(
     float radius,
     int unitCount,
     out List<Vector3> result)
```

ConeFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

ConeFormation

```
[System.Serializable]
public struct ConeFormation
```

Formation that positions units in a cone shape with specified spacing.

Constructors

ConeFormation

```
public  ConeFormation(
     float unitSpacing,
     bool pivotInCenter = true)
```

Instantiates cone formation.

unitSpacing: Specifies spacing between units.
pivotInCenter: Specifies if the pivot of the formation is in the middle of units. By default it is in first row of the formation. If this is set to true, rotation of formation will be in the center.

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

IFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

IFormation

```
public interface IFormation
```

Defines the contact that all formations must implement. Formation should be generated or provided on the fly by calling GetPositions(int).

Methods

GetPositions

```
List<Vector3> GetPositions(
     int unitCount)
```

LineFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

LineFormation

```
[System.Serializable]
public struct LineFormation
```

Formation that positions units in a straight line with specified spacing.

Constructors

LineFormation

```
public  LineFormation(
     float unitSpacing)
```

Instantiates line formation.

unitSpacing: Specifies spacing between units.

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

RectangleBorderFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

RectangleBorderFormation

```
[System.Serializable]
public struct RectangleBorderFormation
```

Formation positions along the edges of a rectangle. When there are less than 4 units to position, straight line is formed. If units cannot be split equally amongs the edges, leftovers are placed in the front line.

Constructors

RectangleBorderFormation

```
public  RectangleBorderFormation(
     float unitSpacing,
     float aspectRatio = 2.0f,
     bool pivotInCenter = true)
```

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

RectangleFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

RectangleFormation

```
[System.Serializable]
public struct RectangleFormation
```

Formation that positions units in a rectangle with specified spacing and maximal column count.

Constructors

RectangleFormation

```
public  RectangleFormation(
     int columnCount,
     float unitSpacing,
     bool centerUnits = true,
     bool pivotInCenter = false)
```

Instantiates rectangle formation.

columnCount: Maximal number of columns per row (there are less rows if number of units is smaller than this number).
unitSpacing: Specifies spacing between units.
centerUnits: Specifies if units should be centered if they do not fill the full space of the row.
pivotInCenter: Specifies if the pivot of the formation is in the middle of units. By default it is in first row of the formation. If this is set to true, rotation of formation will be in the center.

Variables

ColumnCount

```
public readonly int ColumnCount
```

Returns the column count which represents the max unit number in a single row.

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

RingFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

RingFormation

```
[System.Serializable]
public struct RingFormation
```

Formation that positions units in a ring with specified angle and spacing between units.

Constructors

RingFormation

```
public  RingFormation(
     float unitSpacing,
     float circleAngle = 360f)
```

Instantiates circle formation.

unitSpacing: Specifies spacing between units in cricle
circleAngle: Specifies angle for units to be placed, 360 degree means that the units will go entire path around the circle and 180 degree angle means that only half of the circle will be formed.

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

TriangleBorderFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

TriangleBorderFormation

```
[System.Serializable]
public struct TriangleBorderFormation
```

Formation that positions units along the borders of a triangle. Since there are 3 sides to a triangle and units cannot always fit equally, leftovers are filled in the back row, while left and right side are always equal in unit count.

Constructors

TriangleBorderFormation

```
public  TriangleBorderFormation(
     float unitSpacing,
     bool pivotInCenter = true)
```

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

TriangleFormation.cs

Namespaces

TRavljen.UnitFormation.Formations

```
namespace TRavljen.UnitFormation.Formations
```

Classes

TriangleFormation

```
[System.Serializable]
public struct TriangleFormation
```

Formation that positions units in a triangle with specified spacing.

Constructors

TriangleFormation

```
public  TriangleFormation(
     float unitSpacing,
     bool centerUnits = true,
     bool pivotInCenter = false)
```

Instantiates triangle formation.

unitSpacing: Specifies spacing between units.
centerUnits: Specifies if units should be centered if they do not fill the full space of the row.
pivotInCenter: Specifies if the pivot of the formation is in the middle of units. By default it is in first row of the formation. If this is set to true, rotation of formation will be in the center.

Methods

GetPositions

```
public List<Vector3> GetPositions(
     int unitCount)
```

Input

AInputControl.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

AInputControl

```
public abstract class AInputControl
```

Convenience abstraction for MonoBehaviour component implmenting IInputControl. Two default implementations exist for old and new input systems, see ActionInputControl and KeyInputControl.

Properties

OnPlacementActionPress

```
public UnityEvent OnPlacementActionPress
     { get; set; }
```

OnPlacementActionRelease

```
public UnityEvent OnPlacementActionRelease
     { get; set; }
```

OnPlacementActionCancel

```
public UnityEvent OnPlacementActionCancel
     { get; set; }
```

MousePosition

```
public abstract Vector3 MousePosition
     { get; }
```

Methods

new UnityEvent

```
= new UnityEvent()
```

new UnityEvent

```
= new UnityEvent()
```

new UnityEvent

```
= new UnityEvent()
```

ActionInputControl.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

ActionInputControl

```
public class ActionInputControl
```

Input control for formation placement using Unity’ new Input System.

MousePosition

```
public override Vector3 MousePosition
```

SetupDefaultActionsIfNull

```
public void SetupDefaultActionsIfNull()
```

Variables

UnityEngine.InputSystem

```
using UnityEngine.InputSystem
```

UnityEngine

```
using UnityEngine
```

IInputControl.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

IInputControl

```
public interface IInputControl
```

Input communication interface used for unit placement.

Properties

OnPlacementActionPress

```
public UnityEvent OnPlacementActionPress
     { get; set; }
```

Invoke this on placment action press (start).

OnPlacementActionRelease

```
public UnityEvent OnPlacementActionRelease
     { get; set; }
```

Invoke this on placement action release (end).

OnPlacementActionCancel

```
public UnityEvent OnPlacementActionCancel
     { get; set; }
```

Invoke this when placement action cancel is pressed.

MousePosition

```
public Vector3 MousePosition
     { get; }
```

Return current mouse position.

Variables

UnityEngine.Events

```
using UnityEngine.Events
```

KeyInputControl.cs

Namespaces

TRavljen.UnitFormation

```
namespace TRavljen.UnitFormation
```

Classes

KeyInputControl

```
public class KeyInputControl
```

Input control for formation placement using Unity’s old Input System.

MousePosition

```
public override Vector3 MousePosition
```

Placement

APlacementVisuals.cs

Namespaces

TRavljen.UnitFormation.Placement

```
namespace TRavljen.UnitFormation.Placement
```

Classes

APlacementVisuals

```
public abstract class APlacementVisuals
```

Abstract component for showing visuals when placing a formation with FormationPlacement. This can be extended to support any type of visuals, you can check out LinePlacementVisual and FormationIndicatorVisual implementations for examples.

Methods

StartPlacement

```
public virtual void StartPlacement(
      Vector3 start)
```

Invoked at the start of placement with the initial position.

start: World position of placement start

ContinuePlacement

```
public virtual void ContinuePlacement(
     Vector3 newPosition)
```

Invoked when placement is active and new end position was calculated.

newPosition: New position of active placement

StopPlacement

```
public virtual void StopPlacement()
```

Invoked when placement has stopped.

OnFormationReady

```
public virtual void OnFormationReady(
     UnitFormationData formation)
```

Invoked when formation is calculated for visuals during placement. To enable this FormationPlacement.alwaysCalculatePositions must be enabled. This can be disabled to improve performance when it is not required to calculate formation during placement.

formation: New unit formation data

EditorFormationPlacement.cs

Namespaces

TRavljen.UnitFormation.Editor

```
namespace TRavljen.UnitFormation.Editor
```

Enumerations

FormationType

```
[System.Serializable]
public enum FormationType{
     Ring,
     Circle,
     ComputedCircle,
     Cone,
     Line,
     Rectangle,
     RectangleBorder,
     Triangle,
     TriangleBorder}
```

GroundDetectionType

```
[System.Serializable]
public enum GroundDetectionType{
     Disabled,
     NavMesh,
     Raycast}
```

Classes

FormationPlacementTools

```
public class FormationPlacementTools
```

EditorFormationPlacement

```
[ExecuteInEditMode]
public class EditorFormationPlacement
```

groundDetectionType

```
[Tooltip("When placing units on ground, it is convenient to use
this feature " +
"to position them on it, instead of doing it by
hand. " +
"\n\nYou can use Unity’s ’NavMesh’ or using a
’Raycast’ with a specified ’LayerMask’")]
[SerializeField]
GroundDetectionType groundDetectionType
```

groundMask

```
[Tooltip("Specifies layer for detecting ground for unit
position")]
[SerializeField]
LayerMask groundMask
```

groundMaxDistance

```
[Tooltip("Maximal range allowed for ground detection
features.")]
[SerializeField, Range(10, 1000)]
float groundMaxDistance
```

target

```
[Tooltip("Specifies the parent GameObject for automatic
retrieval of unit GameObjects.")]
[SerializeField]
Transform target
```

positionColor

```
[Tooltip("Specifies the color in which Gizmo position spheres
are rendered.")]
[SerializeField]
Color positionColor
```

directionColor

```
[Tooltip("Specifies the color in which Gizmo direction shapes
are rendered.")]
[SerializeField]
Color directionColor
```

sphereRadius

```
[Tooltip("Specifies the sphere radius used on Gizmos for
formation positions.")]
[SerializeField, Range(0.1f, 15f)]
float sphereRadius
```

GroundPositioner

```
public IGroundPositioner GroundPositioner
```

FormationPositions

```
public UnitFormationData FormationPositions
```

ringFormation

```
[SerializeField]
internal RingFormation ringFormation
```

circleFormation

```
[SerializeField]
internal CircleFormation circleFormation
```

computedCircleFormation

```
[SerializeField]
internal ComputedCircleFormation computedCircleFormation
```

coneFormation

```
[SerializeField]
internal ConeFormation coneFormation
```

rectangleBorderFormation

```
[SerializeField]
internal RectangleBorderFormation rectangleBorderFormation
```

rectangleFormation

```
[SerializeField]
internal RectangleFormation rectangleFormation
```

lineFormation

```
[SerializeField]
internal LineFormation lineFormation
```

triangleFormation

```
[SerializeField]
internal TriangleFormation triangleFormation
```

triangleBorderFormation

```
[SerializeField]
internal TriangleBorderFormation triangleBorderFormation
```

AddUnit

```
public void AddUnit(
     Transform unit)
```

ClearUnits

```
public void ClearUnits()
```

ApplyCurrentUnitFormation

```
public void ApplyCurrentUnitFormation(
     bool drawGizmos)
```

Variables

TRavljen.UnitFormation.Placement

```
using TRavljen.UnitFormation.Placement
```

UnityEditor

```
using UnityEditor
```

FormationIndicatorVisual.cs

Namespaces

TRavljen.UnitFormation.Placement

```
namespace TRavljen.UnitFormation.Placement
```

Classes

FormationIndicatorVisual

```
public class FormationIndicatorVisual
```

Component for showing formation positions during active placement. This can be used with FormationPlacement.placementVisuals. Specify the game object used for unit position in formation and component will manage the visuals for their formation. FormationPlacement.alwaysCalculatePositions must be enabled in order to get information about formation positions while placement is active.

Methods

StartPlacement

```
public override void StartPlacement(
     Vector3 start)
```

StopPlacement

```
public override void StopPlacement()
```

OnFormationReady

```
public override void OnFormationReady(
     UnitFormationData formation)
```

FormationPlacement.cs

Namespaces

TRavljen.UnitFormation.Placement

```
namespace TRavljen.UnitFormation.Placement
```

Classes

FormationPlacement

```
public class FormationPlacement
```

Component for placing units in formation with a mouse drag. To disable this when placement is not desired, disable the component like you would any other. (placement.enabled = false)

input

```
internal IInputControl input
```

UnitFormation

```
public UnitFormation UnitFormation
```

IsPlacementActive

```
public bool IsPlacementActive
```

AddDefaultInput

```
public void AddDefaultInput()
```

StartPlacement

```
public void StartPlacement()
```

Starts the unit placement process. It will not start if UnitFormation was not set, if unit formation has no units or if raycast did not hit a valid ground.

FinishPlacement

```
public void FinishPlacement()
```

Completes the process of placement, if placement is active.

CancelPlacement

```
public void CancelPlacement()
```

Cancels current placement if one is active.

SetUnitFormation

```
public void SetUnitFormation(
    UnitFormation unitFormation)
```

Update unit formation used for placement.

unitFormation: New unit formation

SetInput

```
public void SetInput(
    IInputControl input)
```

Set new input reference.

input: Input reference

SetFormation

```
public void SetFormation(
    IFormation formation)
```

Set current formation used calculating units positions.

formation: New formation

ApplyCurrentUnitFormation

```
public void ApplyCurrentUnitFormation()
```

Apply formation positions based on active placement or last active placement.

IGroundPositioner.cs

Namespaces

TRavljen.UnitFormation.Placement

```
namespace TRavljen.UnitFormation.Placement
```

Classes

IGroundPositioner

```
public interface IGroundPositioner
```

Interface for searching nearest valid position for the pathfinding system.

Methods

PositionOnGround

```
public Vector3 PositionOnGround(
     Vector3 position,
     float maxDistance)
```

Searches for nearest ground position valid for pathfinding.

position: Desired position

Returns: Returns valid pathfinding position or the same value if no valid position was found inside the max distance radius.

LinePlacementVisual.cs

Namespaces

TRavljen.UnitFormation.Placement

```
namespace TRavljen.UnitFormation.Placement
```

Classes

LinePlacementVisual

```
[RequireComponent(typeof(LineRenderer))]
public class LinePlacementVisual
```

Component for showing formation center and direction during active placement. This can be used with FormationPlacement.placementVisuals. It uses LineRenderer for drawing the direction. If another technique is desired, use APlacementVisuals to implement your own.

StartPlacement

```
public override void StartPlacement(
     Vector3 start)
```

ContinuePlacement

```
public override void ContinuePlacement(
     Vector3 end)
```

StopPlacement

```
public override void StopPlacement()
```