PathEasy

PathEasy is a lightweight tool that makes creating motion paths in Unity pretty straightforward.
You can draw and adjust paths right in the Scene View, then have objects follow those paths or place prefabs along them—no code needed. It's built on top of DOTween, so everything moves smoothly.

The best part? **You can preview everything in the Editor without hitting Play Mode.** Just test how an object moves along a path instantly. This saves a ton of time when you're tweaking motion or layout.

PathEasy does two main things:

Moving objects along a path – Create and preview motion paths visually, then save them as ScriptableObjects so you can reuse them later.

Placing objects along a path – Automatically place objects using fixed counts or spacing. Supports multiple prefabs and you can edit everything in the Scene View.

Table of Contents

Overview

What is PathEasy?

PathEasy lets you create and edit paths visually:

1. **Visual Path Editor**: Draw and edit paths right in the Scene View - both PathEasyPlay and PathEasyPlacer have this built in
2. **Preset Templates**: Need a circle or star? There are presets for common shapes
3. **ScriptableObject Storage**: Save paths as assets and reuse them on different objects
4. **Path Play System**: Animate objects along paths with Editor Preview support
5. **Path Placer Tool**: Automatically place objects along a path - supports multiple prefabs, fixed count or spacing modes, plus animated spawning
6. **Editor Preview**: Test your paths without entering Play Mode (saves a lot of time)
7. **Two Curve Modes**: Catmull-Rom for smooth curves automatically, or Bezier if you want to control things manually

Why Use It?

- **No Coding Needed**: Everything is visual, so designers can use it too
- **See Changes Instantly**: Edit paths in the Scene View and watch them update in real-time
- **Reuse Paths**: Save a path once and use it on as many objects as you need
- **Fast at Runtime**: Paths are pre-calculated automatically so they run smoothly in-game
- **Works with TweenEasy**: If you're already using TweenEasy, they work well together

Quick Start

Installation

1. Import the PathEasy package into your project
2. Make sure you have DOTween installed (it's required)

Basic Setup

For Path Animations:

1. Add the `PathEasyPlay` component to a GameObject
2. Click **Create New Path Data File** to create a new path (or use an existing PathData)
3. Enable **Path Editing** to edit paths in the Scene View
4. Pick a preset template or just draw your own path
5. Configure playback settings (duration, loops, easing, etc.)
6. Hit Play or use Editor Preview to see it follow the path

For Path Object Placement:

1. Add the `PathEasyPlacer` component to a GameObject
2. Click **Create New Path Data File** to create a new path (or use an existing PathData)
3. Enable **Path Editing** to edit paths in the Scene View (optional)
4. Assign the prefab(s) you want to place (supports multiple prefabs)
5. Choose **Count** (place X objects) or **Spacing** (place objects every X units)
6. Set the value and it'll automatically place objects along the path
7. (Optional) Enable `placeObjectsOnStart` for animated spawning in Play Mode

System Requirements

- Unity 2020.3 or newer
- DOTween (the free version works fine, Pro works too if you have it)

Documentation

For more details, check out:

- **Path Editing Guide** - Everything about path editing features, curve modes, presets, and visual editing
- **Path Play Guide** - Complete guide to PathEasyPlay: animation, playback control,

Editor Preview, API, and examples

- **Path Placer Guide** - How to use the placer tool, placement modes, auto-update, API, and examples

FAQ

General Questions

Q: Do I need to write any code?

A: Nope! Everything can be done in the Inspector. You don't need to write any code.

Q: How do I create and edit paths?

A: Both `PathEasyPlay` and `PathEasyPlacer` have built-in path editing:

1. Add the component to a GameObject
2. Click **Create New Path Data File** to create a new path
3. Enable **Path Editing** in the Inspector
4. Edit paths directly in the Scene View using Unity's position handles
5. Use preset templates for quick shape generation

Q: Can I preview path animations in Edit Mode?

A: Yes! **PathEasyPlay has full Editor Preview support.**

- Just click the **Play Preview** button in the Inspector
- You don't need to enter Play Mode to test things
- All the settings work: duration, loops, easing, look-at, everything

Q: How do I use paths at runtime?

A: Here's how it works:

1. Create your path using PathEasyPlay or PathEasyPlacer in Edit Mode

2. Enable Path Editing and create/edit your path
3. Assign the PathData ScriptableObject to PathEasyPlay
4. Either call `Play()` from code or enable `playOnStart` to auto-play

Q: What's the difference between Absolute and Relative coordinate systems?

A:

- **Absolute**: The path uses fixed world coordinates. The object moves to exact positions regardless of where it starts.
- **Relative**: The path is offset from the object's starting position. Good for reusable patterns that should work anywhere.

Q: Why aren't my path changes saving?

A: Path changes are automatically saved to the PathData ScriptableObject when you edit them. Make sure Path Editing is enabled and you're actually editing the path - the changes should persist automatically.

Q: How do I improve path animation performance?

A: A few things that help:

1. The system automatically generates optimized path data
2. Paths are pre-calculated by default for better performance

Q: Can I share paths between multiple objects?

A: Yes! That's the whole point of using ScriptableObjects. Multiple PathEasyPlay or PathEasyPlacer components can all use the same PathData, and each object will use the path independently.

Q: What's the minimum/maximum number of path points?

A:

- Minimum: 2 points (you need at least two to make a path)
- Maximum: No hard limit, but I'd recommend staying under 100 for performance reasons

Q: What's the difference between Catmull-Rom and Bezier curve modes?

A:

- **Catmull-Rom Mode**: Automatically makes smooth curves between points. You can adjust the smooth intensity (0-10) to control how smooth it is. Good for quick setup when you just want smooth paths without much tweaking.
- **Bezier Mode**: You control the tangents manually. Adjust the In Tangent and Out Tangent handles to control exactly how curves enter and leave each point. Better if you need precise control over the curve shape.

Q: What are the different Tangent Modes in Bezier mode?

A: When you're using Bezier mode, there are three tangent control modes:

1. **Mirrored** (Default): Tangents stay in a straight line with equal lengths. Makes perfectly symmetric curves.
2. **Aligned**: Tangents stay in a straight line (opposite directions) but can have different lengths. Keeps things smooth but lets you make it asymmetric.
3. **Free**: Both tangents are completely independent. Adjust each one separately for maximum control.

Q: How do I make an object look at the path direction?

A: Enable `lookAtPath = true` in PathEasyPlay. Adjust `lookAtStrength` (1-50) to control how fast it rotates toward the path direction.

Q: Can an object look at a target while following a path?

A: Yes! Just set a `lookAtTarget` Transform. The object will face that target while moving along the path. Turn off `lookAtPath` if you only want to look at the target instead of the path direction.

Q: How do I trigger custom logic when a path animation starts or finishes?

A: Use the UnityEvents: `onPathStart` and `onPathComplete`. Just drag your functions into those slots in the Inspector and they'll fire at the right time.

Cross-System Integration

Q: Can PathEasy work with TweenEasy?

A: Yes! They're designed to work together. If you're using both, they'll work well with each other.

Support

If you run into issues or have suggestions, check out the support website.

Thanks for trying **PathEasy**! Hopefully it makes creating paths in Unity a bit easier.