The Link to the Online and Constantly Updated Version of this Document HERE

Purpose

Linear Interpolation is nice and in Unity there are multiple classes with some version of .Lerp(). Namely Mathf, Vector2, Vector3, Vector4, and Color. All of these ask for a start and end sequence of values and a **lerp value**. However, given Unity's Guides all we can do is specify that lerp value and use (Time.deltaTime or Time.fixedDeltaTime) or not, to make the linear interpolation non frame dependent or frame dependant respectively.

But I wanted to travel from location A to location B in a certain number of seconds or frames. Doing this manually for every project is a massive hassle. That is why I created this API.

How It's Done

I do this by defining the distance between A and B as a **lerp distance** and the time it takes to travel lerp distance as **lerp time**. From these we can create a **lerp velocity** (lerp distance / lerp time). A velocity doesn't require the specific distance or the specific time, it only requires the proportion of both to be the same. So we use a **guide distance** and a **guide time** to then calculate a lerp velocity instead.

This allows us to define multiple standard guide distances below, or even set our own custom guide distance to calculate the lerp velocity. Note that the guide distance and guide time may very well be equal to lerp distance and lerp time; but now the choice of using that or an alternative is up to you.

What Lerp Kit Does

Lerp Kit creates 2 functions...

1. calcGuideDistance
2. calcLerpValue
    a. Version 1: by using only a lerp velocity
    b. Version 2: by using Guide Distance, Guide Time, Unit of Time, and Update Location to calculate lerp velocity and then run Version 1

for each type that has a Lerp method in Unity. Namely...

- Mathf
- Vector2
- Vector3
- Vector4
- Color (in RGB Color Space)
    - NOTE: as part of a separate API called colorKit I have lerp functions for Color in RGB, RYB, and CMYK (but it is not included in this API)

NOTE: that lerp velocity is always in **distance per frame** when calculated by the functions in lerpHelper and should be in distance per frame if passed manually.

How To Use Lerp Kit

There are 2 ways to use Lerp Kit; each with their own pros and cons...

1. using the "lerpKit" namespace
    a. use the "using lerpKit;" directive → use "lerpHelper.function(...)" to call the function
2. using extension methods
    a. create a "dummy instance" of (Mathf, Vector2, Vector3, Vector4, or Color) → use "dummyInstance.function(...)" to call the function
    b. create an "instance" of (Mathf, Vector2, Vector3, Vector4, or Color) → use "instance.function(..)" to call the function with "instance" replacing the first parameter [of the dummy instance method]

Enumerables

- **updateLocation** { fixedUpdate, Update };
    - it makes a difference where you place your "Type.lerp(...)" and you must indicate where you place it to properly calculate your lerp velocity
    - NOTE: this can be read in by reading the call stack but its too expensive to be worth it
- **unitOfTime** { frames, seconds };
    - this easily converts your guide time into frames if you pass it seconds
- **guideDistance** { distBetween_Other, distBetween_StartAndEnd, distBetween_CurrAndEnd, distBetween_StartAndCurr };
    - it makes sense to use one of these often. So you can simply chose an enumerable and the functions will calculate the distance you chose for you
    - NOTE: distBetween_Other should not be used. To pass a custom distance you should not need the guideDistance enum.

Lerp Kit Documentation

The namespace that contains the implementation of every class used is called **"lerpKit"**
The only class contained within that namespace is called **"lerpHelper"**
The Functions within "lerpHelper" are… (all of these are public static float)

- **For Guide Distance Calculation**
    - calcGuideDistance(float startValue, float currValue, float endValue, guideDistance GD)
    - calcGuideDistance(Vector2 startVect2, Vector2 currVector2, Vector2 endVector2, guideDistance GD)
    - calcGuideDistance(Vector3 startVect3, Vector3 currVector3, Vector3 endVector3, guideDistance GD)
    - calcGuideDistance(Vector4 startVect4, Vector4 currVector4, Vector4 endVector4, guideDistance GD)

    - calcGuideDistance(float[] startValues, float[] currValues, float[] endValues, guideDistance GD)
- **For Lerp Value Calculation**
    - **using Guide Velocity**
        - calcLerpValue(float startValue, float currValue, float endValue, float lerpVelocity_DperF)
        - calcLerpValue(Vector2 startVector2, Vector2 currVector2, Vector2 endVector2, float lerpVelocity_DperF)
        - calcLerpValue(Vector3 startVector3, Vector3 currVector3, Vector3 endVector3, float lerpVelocity_DperF)
        - calcLerpValue(Vector4 startVector4, Vector4 currVector4, Vector4 endVector4, float lerpVelocity_DperF)
        - calcLerpValue(float[] startValues, float[] currValues, float[] endValues, float lerpVelocity_DperF)
        - calcLerpValue(Color startColor, Color currColor, Color endColor, float lerpVelocity_DperF)
    - **using Guide Distance, Guide Time, Unit of Time, and Update Location**
        - calcLerpValue(float startValue, float currValue, float endValue, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
        - calcLerpValue(Vector2 startVector2, Vector2 currVector2, Vector2 endVector2, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
        - calcLerpValue(Vector3 startVector3, Vector3 currVector3, Vector3 endVector3, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
        - calcLerpValue(Vector4 startVector4, Vector4 currVector4, Vector4 endVector4, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
        - float calcLerpValue(float[] startValues, float[] currValues, float[] endValues, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
        - calcLerpValue(Color startColor, Color currColor, Color endColor, float guideDistance, float guideTime, unitOfTime UOT_GD, updateLocation UL)
- **EXTRA**
    - distance(Color color1, Color color2)
        - This was Created because Color has no definition of Distance. So I simply Imagined every possible color as a value in a 3D color cube and then I was able to use Vector3.Distance()