Character Movement

Class Description:
This class manages your character's movement, seamlessly combining animation curves with Unity's Rigidbody2D component. This dynamic setup empowers you to create various types of movement, whether snappy or smooth.

In the inspector, exposed properties allow you to fine-tune both 'ground' and 'aerial' movement, providing animation curves for acceleration and deceleration. Additionally, dedicated values enable control over the evaluation speed of these curves. These values play a crucial role in determining how quickly your character reaches the designated top speed, as defined by the 'moveSpeed' variable.

Considerations and Caveats:
The CharacterMovement class operates independently of external changes in your character's speed and movement direction. It calculates these parameters autonomously. In the event of a sudden velocity change or a shift in movement direction outside of this script, it's crucial to manually adjust both the CharacterMovement.CurrentSpeed and CharacterMovement.MoveDirection properties accordingly.

For instance, if an external force causes your character to abruptly change direction, you need to update the MoveDirection property to reflect the new movement direction. Similarly, adjusting the CurrentSpeed property ensures synchronization between the script and any external modifications to the character's velocity. This proactive approach guarantees accurate and responsive control over your character's movement behavior.

Public Properties:
- float TopSpeed { get; }: Character's movement speed.
- float CurrentSpeed { get; set; }: Current speed of your character.
- Vector3 MoveDirection { get; set; }: The direction in which your character is moving.

Public Methods:
1. void SetCharacterMoveSpeed(float value):
- value: New movement speed value.
- Description: *Lets you change the movement speed (top speed) for your character.*

2. Vector2 OnGroundHorizontalVelocity(Vector2 moveInput, bool runsIntoWall):
- moveInput: Directional input provided by the player or AI controller.
- runsIntoWall: Indicates whether the character is colliding with a wall.
- Returns: A vector with the calculated X-axis velocity.
- Description: *Calculates the horizontal movement of your character.*

3. Vector2 VerticalVelocity(bool againstWall):
- againstWall: True if your character is performing a jump against a wall.
- Returns: A vector with the calculated Y-axis velocity.
- Description: *Calling this method will make your character perform a jump.*

4. Vector2 OnAirHorizontalVelocity(Vector2 moveInput, bool runsIntoWall):
- moveInput: Directional input provided by the player or AI controller.
- runsIntoWall: Indicates whether the character is colliding with a wall.
- Returns: A vector with the calculated velocity for the X-axis only, based on the move speed the character had when grounded.
- Description: *Calculates your character's aerial movement.*