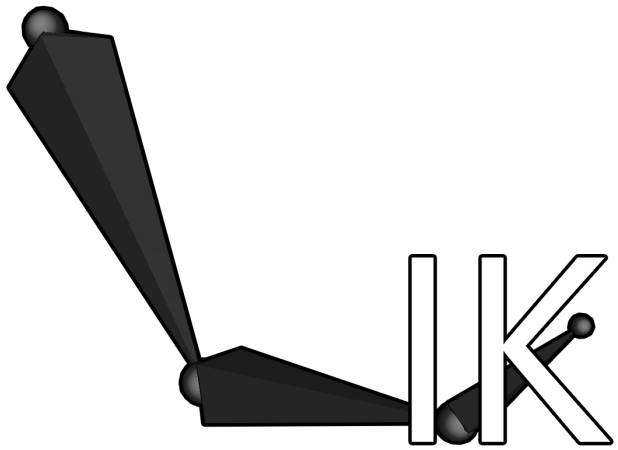


EzyIK Documentation

Henry Oliver

Table Of Contents

1.0 Preface

The goal of this plugin is to ease the implementation of Inverse Kinematics to work inside a Unity project without pre-existing knowledge of Inverse Kinematics.

1.1 Acknowledgments

I would like to thank my friends Vaughan Webb, Ryan Wilkson, Matt Moore, David Morris, Elijah Shadbolt, Harry Orsborne, Xin Yin Lee, Lucielle Liu and Darren Yu for and lecturer Zac Watson for giving me feedback and pushing me further to make the plugin more accessible and feature complete.

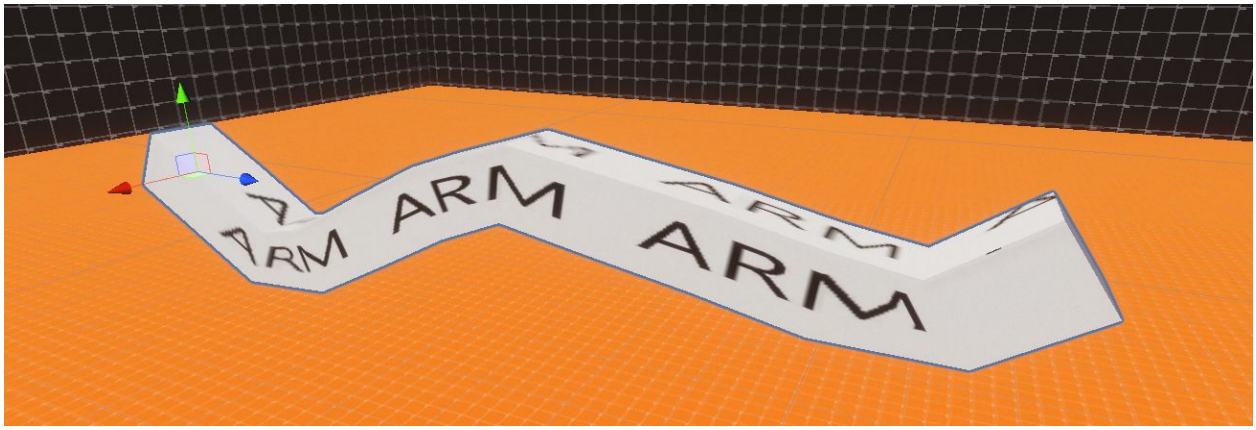
1.2 Introduction

Inverse Kinematics is used in games as an alternative to or used in conjunction with normal animation to help a bone reach a target destination in a realistic manner. This can be difficult to implement without appropriate knowledge so I made this plugin to ease the implementation of Inverse Kinematics.

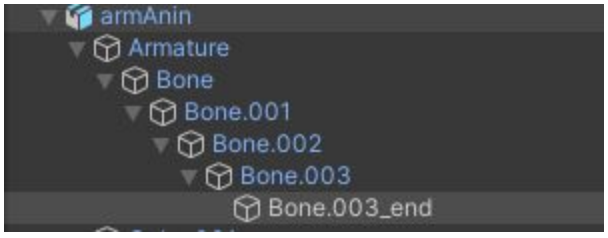
2.0 Quick Start

You can start utilizing the EzyIK system with five easy steps.

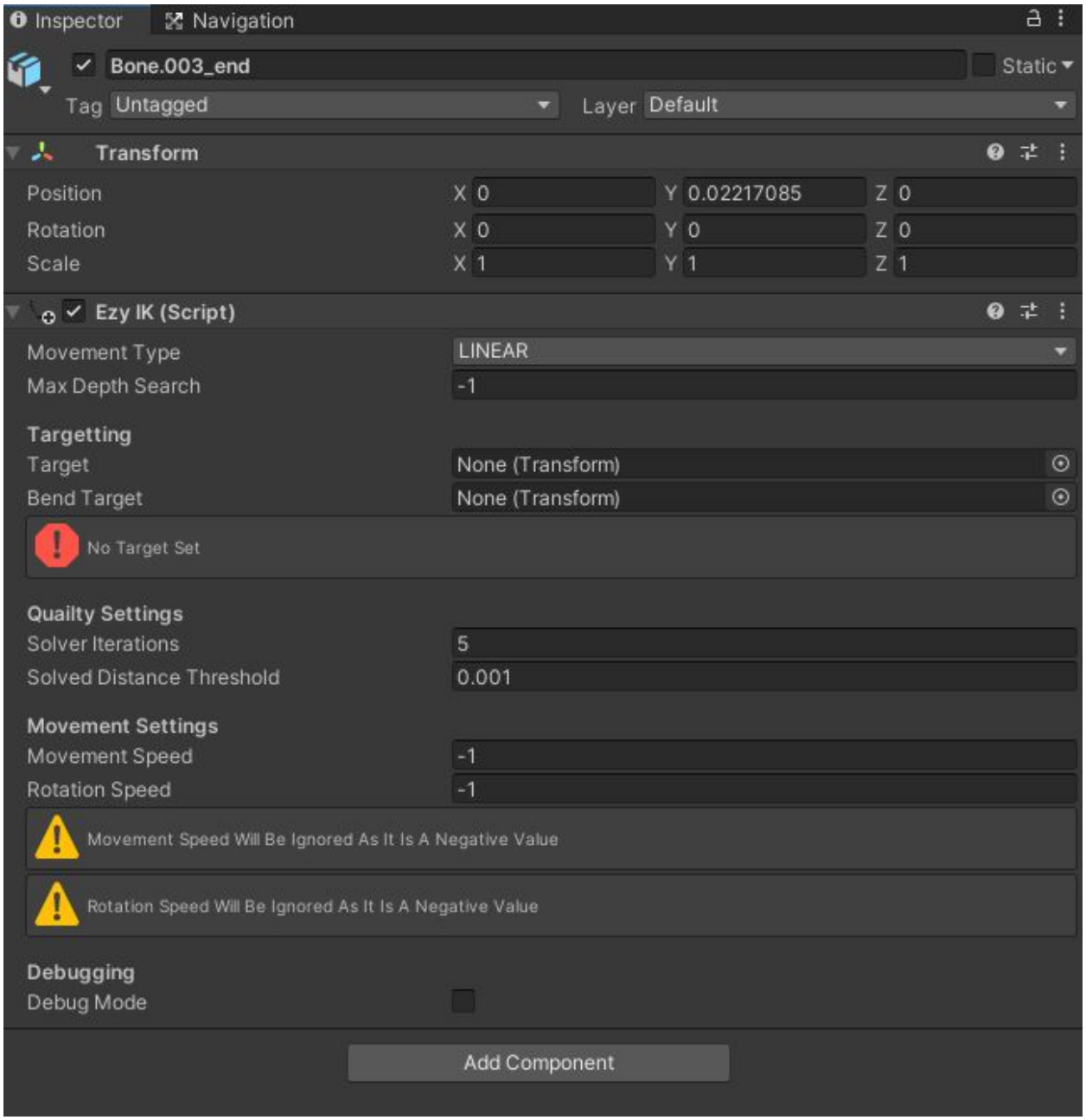
1. Drag a rigged object into the scene



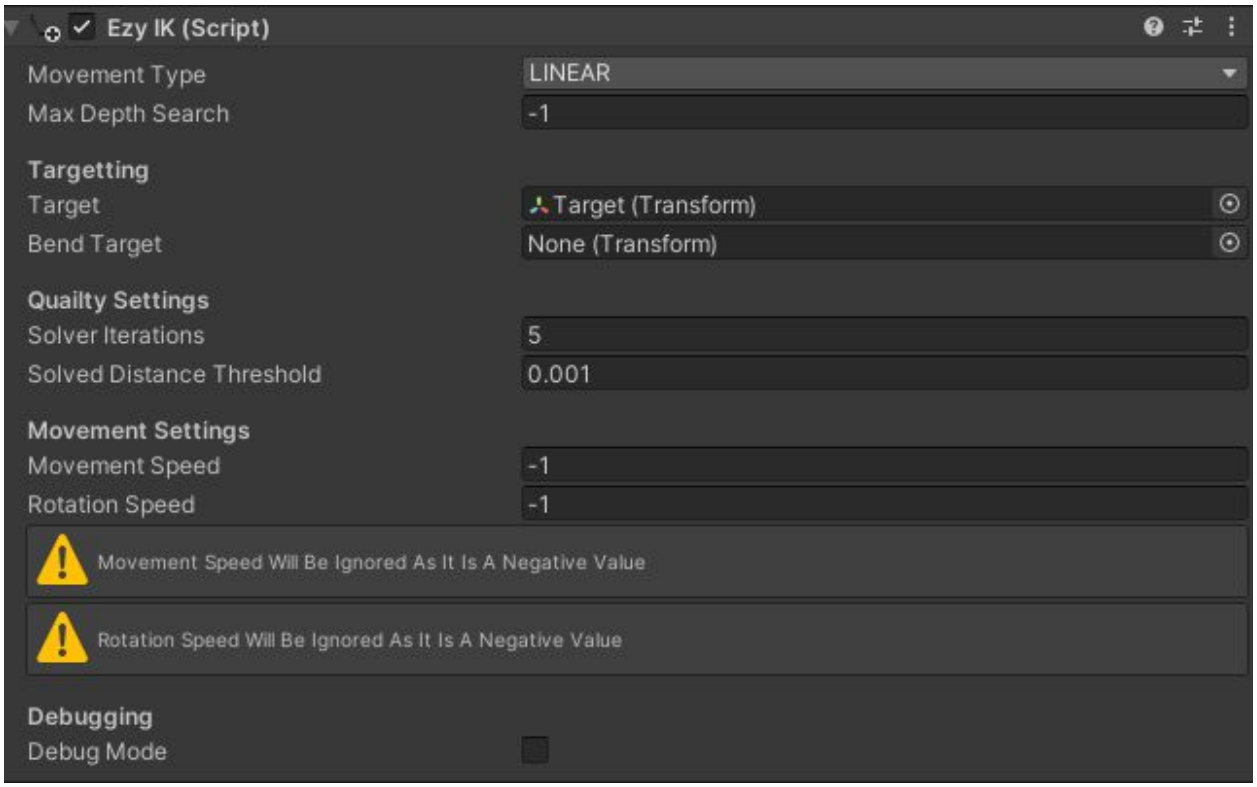
2. Navigate to the last bone in the rig that you want to affect



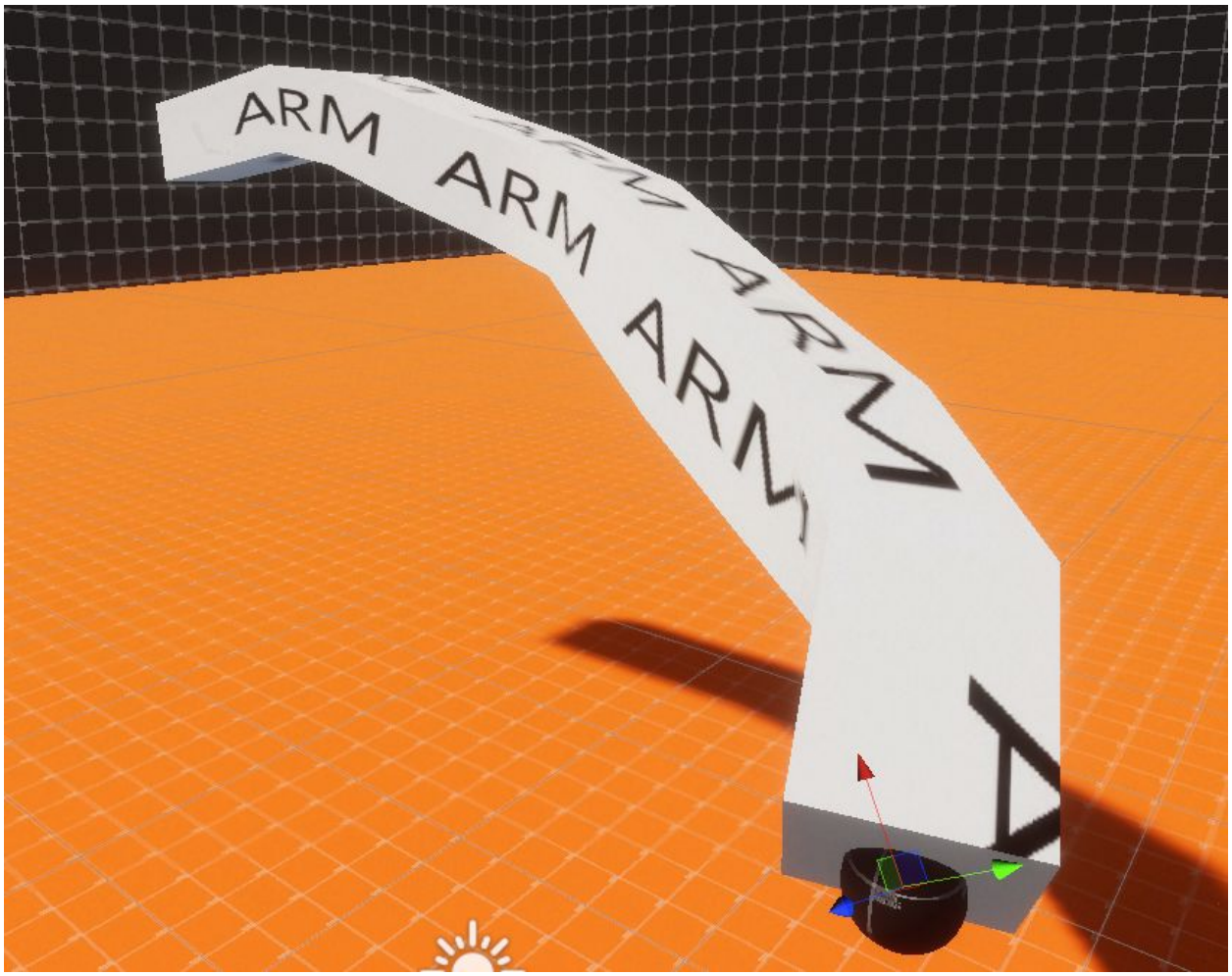
3. Add the EzyIK Component onto the bone



4. Create and Drag a target into the target slot



5. Press Play



2.1 EzyIK Parameters

Note that you cannot change parameters during runtime.

Max Depth Search

The Max Depth Search defines how many parent bones we will affect, if the value is set to -1 then we will affect all bones that we are parented to, otherwise, we will be limited to the value specified in the max depth search field.

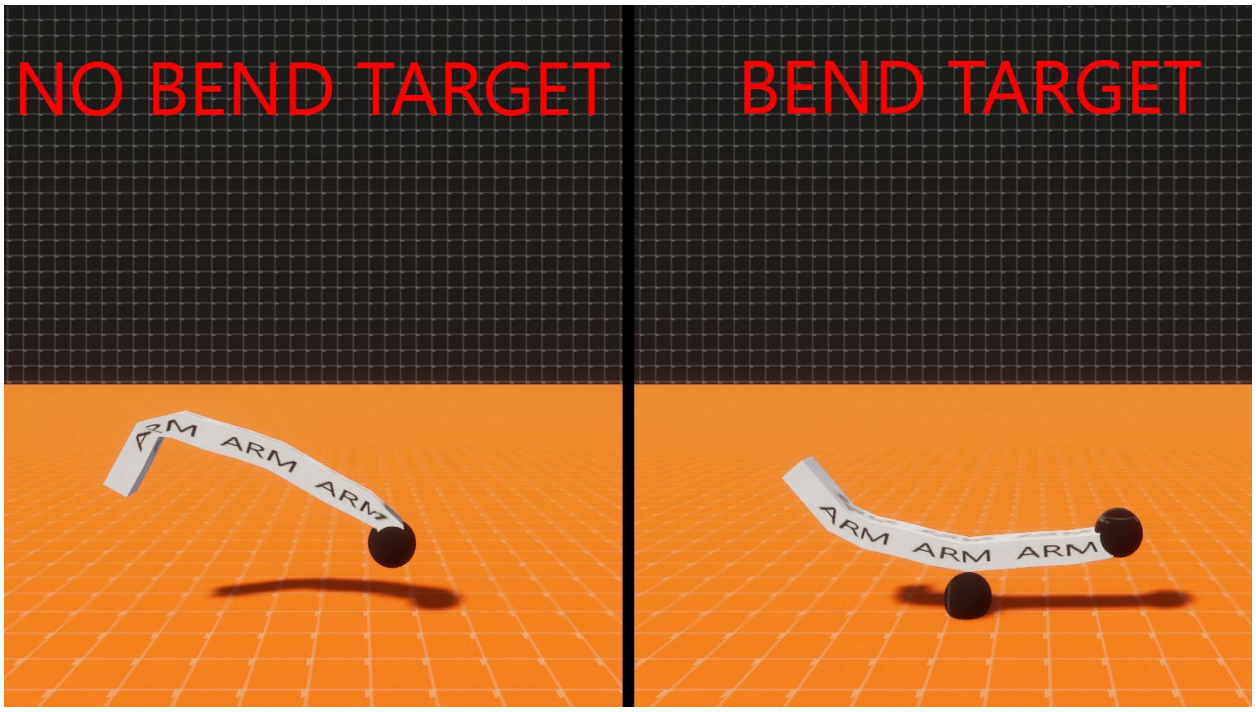
Targeting Settings

Target

Parameter Type: Transform
The target parameter is used by the plugin to determine what transform to attempt to reach. A good example of a target would be the player.

Bend Target

Parameter Type: Transform
The bend target parameter is used to help inform the bones which direction they should bend.



Quality Settings

Solver Iterations

Parameter Type: Integer
Ezy IK uses the FABRIK algorithm to implement it's Inverse Kinematics, because of this we need to solve the Forwards and Backwards part of the algorithm which is done with iterations, the more iterations the more accurate the outcome.

Solved Distance Threshold

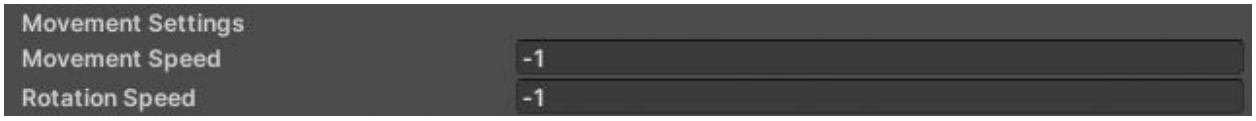
Parameter Type: Float
The Solved Distance Threshold parameter is used to determine if we can stop attempting to solve the Forward and Backward section of the FABRIK algorithm before we hit our limit.

Movement Type

Parameter Type: Dropdown Menu

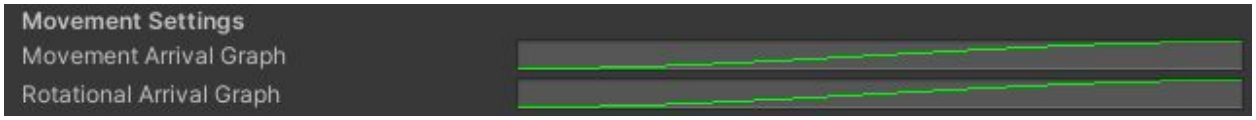
Linear

When the movement type is set to linear bones will move in a linear fashion according to the movement speed and rotation speed parameters.



Custom

When the movement type is set to custom, bones will move in relation to the graph movement arrival speed and rotation arrival graph.



Movement Settings

Parameter Type: Float/Animation Curve

Linear Settings

Movement Speed

The movement speed parameter restricts how fast bones can move if set to -1 bones will move with unrestricted speed.

Rotation Speed

The rotation speed parameter restricts how fast bones can rotate if set to -1 bones will rotate with the same speed that is set in the movement parameter.

Custom Settings

Movement Arrival Graph

The movement arrival graph restricts how fast bones can move based on the leaf node's distance from the target. The X-axis represents the distance of the leaf node from the target, the Y-axis represents speed. If you need to graph speeds or distances greater than 1.0 then you can use the middle mouse button to drag the graph.

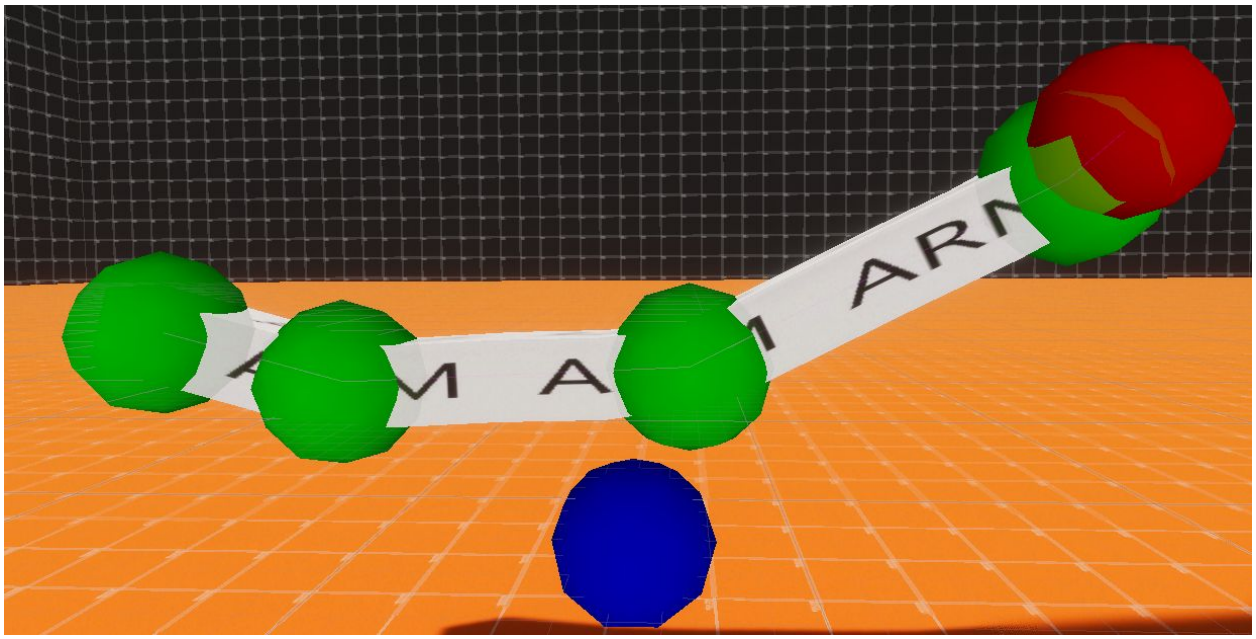
Rotational Arrival Graph

The rotational arrival graph restricts how fast bones can rotate based on the leaf node's distance from the target. The X-axis represents the distance of the leaf node from the target, the Y-axis represents speed. If you need to graph speeds or distances greater than 1.0 then you can use the middle mouse button to drag the graph.

Debugging

Debug Mode

To enable debugging mode tick the Debug Mode parameter. Spheres will visualize the bone structure. Red representing the target, Blue represents the bend target and green represents the bone nodes.



2.2 Building A Custom EzyIK Controller

Below are functions and classes that you can interface with using your own custom controller.

BoneNode

public class BoneNode(ref GameObject _node);

This class is used to represent a node in our bone structure.

Parameters

Parameter	Description
_node	The game object that you want to become a bone node.

Properties

Property	Description
node	Returns our node's game object
nodeTransform	Returns our node's transform
startRot	Returns our node's starting rotation
startDirTarget	Returns a Vector3 of our node's starting rotation to our target, if we are the leaf node it will be relative to the target transform otherwise it will be relative to the next node.
initalDistanceToChild	Returns a float of our starting distance to our child node

BoneStructure

public class BoneStructure(ref GameObject startBone, ref int maxDepth, ref Transform _target, ref Transform _bendTarget, ref float _arriveThreshold, ref int _maxSolveIterations, ref float _moveSpeed, ref float _rotSpeed, ref MoveType _moveType, ref AnimationCurve _moveCurve, ref AnimationCurve _rotCurve);

This class is used to represent the bone structure that the plugin builds and uses during runtime.

Parameters

Parameter	Description
startBone	The game object that has the EzyIK component attached.
maxDepth	The maximum depth to search for bones.
_target	The target's transform.
_bendTarget	The bend target's transform.
_arriveThreshold	The arrival threshold for the Forward and Backwards loop.
_maxSolveIterations	The maximum iterations that the Forward and Backwards loop can run.
_moveSpeed	How fast bones move linearly.
_rotSpeed	How fast bones rotate linearly.
_moveType	Assigns the type of movement are we using.
_moveCurve	How fast bones move along a graph.
_rotCurve	How fast bones rotate along a graph.

Properties

Property	Description
boneNodes	Returns a List of bone nodes, used to store

	all the bones in our bone structure.
MoveType	Enumerator Containing the following: ● LINEAR ● CUSTOM
moveType	Returns a MoveType depending on what current movement move we are on.
moveCurve	Returns an AnimationCurve, is used for moving nodes at a speed relative to a graph while in CUSTOM move mode.
rotCurve	Returns an AnimationCurve, is used for rotating nodes at a speed relative to a graph while in CUSTOM move mode.
startTargetRot	Returns a Quaternion of the starting rotation of the root node relative to the target.
rootNode	Returns a Transform of the root node.
target	Returns the Transform of the target.
bendTarget	Returns the Transform of the bend target.
totalChainDistance	Returns a float of the total length of all the bones.
totalChainDistanceSquared	Returns a float of the total length of all the bones squared.
arriveThreshold	Returns a float of the arrival threshold.
arriveThresholdSquared	Returns a float of the arrival threshold squared.
nodeMoveSpeed	Returns a float of the node movement speed for linear movement mode.
rotMoveSpeed	Returns a float of the node rotation speed for linear movement mode.
maxDepth	Returns an integer of our max depth search.
maxSolveIterations	Returns an integer of our max iterations.

IKStep()

public static void IKStep(ref BoneStructure boneStructure);
IKStep() should be called on LateUpdate(). This is used to update the bone structure.

Parameters

Parameter	Description
boneStructure	The bone structure that we want to update.

3.0 Known Bugs

If you move the target closer and further between the reaching threshold the bone appears to snap into place, which doesn't look too realistic. You can try to lower this effect by adjusting movement speeds.

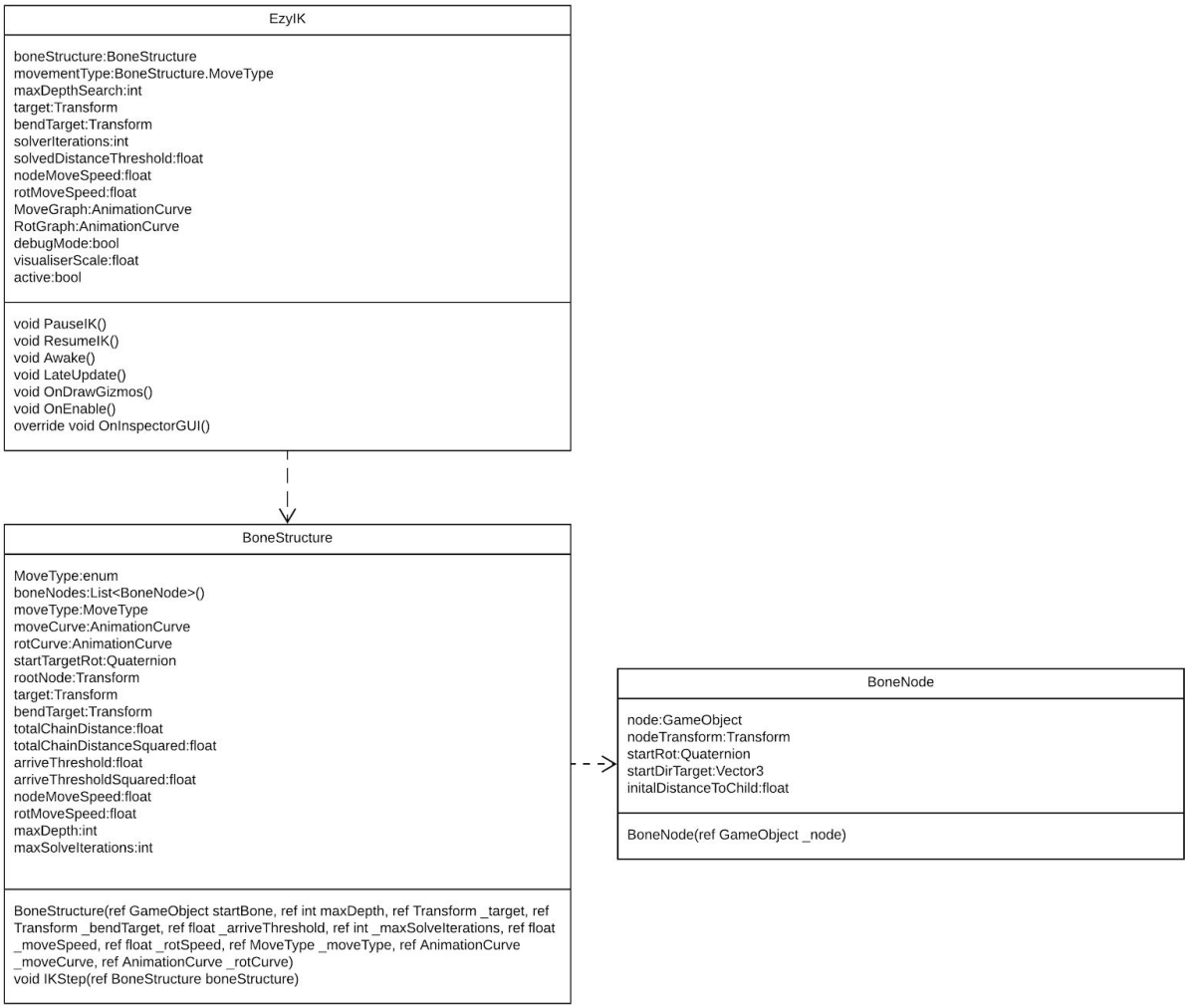
3.1 Future Improvements

Future iterations of this plugin will be focused on fixing bugs.

4.0 How I made the plugin

When creating this plugin I first searched for pre-existing algorithms, I chose the FABRIK algorithm as it was the best I found that balanced performance and quality to my preference. I then created a basic rig of a rectangle to be my arm, I then added a small animation to test blending and exported it as a fbx file. I found some coding examples after choosing my algorithm and mostly followed a YouTube tutorial I found on implementing Inverse Kinematics in Roblox. After implementing what I had learnt in the video I found that there was no mention of rotations and of course the end result made the bones move to the correct positions but lacked any rotations. After a lot of googling I managed to expand my existing work without any reworking to implement rotation. I then was very happy with the result so I created the hands sample scene. I then decided to enable the animation to see if I could blend it only to be meet with it instantly working fine with my plugin which was a pleasant surprise. After this had been done I focused on making a custom editor script as the current component was messy. Once that was done I decided to finish building the plugin as I saw it as complete.

Below are included a UML diagram of the Plugin.


EzyIK
boneStructure:BoneStructure
movementType:BoneStructure.MoveType
maxDepthSearch:int
target:Transform
bendTarget:Transform
solverIterations:int
solvedDistanceThreshold:float
nodeMoveSpeed:float
rotMoveSpeed:float
MoveGraph:AnimationCurve
RotGraph:AnimationCurve
debugMode:bool
visualiserScale:float
active:bool
void PauseIK()
void ResumeIK()
void Awake()
void LateUpdate()
void OnDrawGizmos()
void OnEnable()
override void OnInspectorGUI()
BoneStructure
MoveType:enum
boneNodes:List<BoneNode>()
moveType:MoveType
moveCurve:AnimationCurve
rotCurve:AnimationCurve
startTargetRot:Quaternion
rootNode:Transform
target:Transform
bendTarget:Transform
totalChainDistance:float
totalChainDistanceSquared:float
arriveThreshold:float
arriveThresholdSquared:float
nodeMoveSpeed:float
rotMoveSpeed:float
maxDepth:int
maxSolveIterations:int
BoneStructure(ref GameObject startBone, ref int maxDepth, ref Transform _target, ref Transform _bendTarget, ref float _arriveThreshold, ref int _maxSolveIterations, ref float _moveSpeed, ref float _rotSpeed, ref MoveType _moveType, ref AnimationCurve _moveCurve, ref AnimationCurve _rotCurve)
void IKStep(ref BoneStructure boneStructure)
BoneNode
node:GameObject
nodeTransform:Transform
startRot:Quaternion
startDirTarget:Vector3
initalDistanceToChild:float
BoneNode(ref GameObject _node)