Pick-and-place and IK Controller

Overview:

The `IKController` class provides an inverse kinematics solution for robotic arms in Unity. It allows the robotic arm to adjust its joints to reach or point to a specific target position. This is achieved through the gradient descent algorithm.

Properties:

- **_targetGameObject**: An array of GameObjects representing the targets the robotic arm should move towards.
- **actuator**: The GameObject representing the end effector of the robotic arm.
- **ikTarget**: The GameObject representing the inverse kinematics target.
- **endPoints**: An array of Transforms that indicate the end points of the robotic arm.
- **_targetTransform**: An array of Transforms associated with the `_targetGameObject`.
- **Joints**: An array representing the joints of the robotic arm. The joints determine the rotation axis.
- **Angles**: An array of floats representing the current angles of the robotic arm joints.
- **InverseKinematicController**: A GameObject that serves as the controller for inverse kinematics.
- **suctionCupTransform**: Transform component of the actuator.
- **delayedIKTarget**: An instance of `DelayedIKTargetUpdate` attached to the current GameObject.
- **collisionPenalty**: A penalty score added when joints come too close together to avoid self-collision.
- **target**: The target Vector3 position that the robotic arm should move towards.

Constants:

- **SamplingDistance**: The distance for sampling in the gradient descent algorithm.
- **LearningRate**: The rate at which the angles are adjusted during the gradient descent.
- **DistanceThreshold**: The threshold under which the arm is considered to have reached the target in terms of distance.
- **AngleThreshold**: The threshold under which the arm is considered to have reached the target in terms of angle.

Methods:

`Start()`

Initializes the `_targetTransform` array by fetching the `Transform` component of each target GameObject. It also initializes the `suctionCupTransform` and sets the initial target for the robotic arm based on the `Dropped` flag.

`StartMovementWithDelay()`

Starts the inverse kinematics movement after a delay. Not used in the demo implementation, but can be used as an alternative to th distance-based default.

`InverseKinematicCoroutine()`

A coroutine that updates the target position to the position of the `ikTarget` and adjusts the angles of the joints to move towards this new target.

`Update()`

Called once per frame. Adjusts the target position based on various conditions and executes the inverse kinematics algorithm.

`ForwardKinematics(float[] angles)`

Given an array of joint angles, it computes the position of the end effector using forward kinematics.

`DistanceFromTarget(Vector3 target, float[] angles)`

Calculates the distance between the target position and the end effector's position.

`AngleWithTarget(Vector3 target, float[] angles)`

Calculates the angle between the target position and the end effector's position.

`ErrorFunction(GameObject target, float[] angles)`

Calculates the error between the end effector's position and rotation and the target's position and rotation.

`PartialGradient(Vector3 target, float[] angles, int i)`

Computes the partial gradient of the error with respect to a specific joint angle.

`InverseKinematics(Vector3 target, float[] angles)`

The core method for the inverse kinematics solution. Adjusts the joint angles so that the end effector moves closer to the target position.

Remarks:

The script adjusts the robotic arm's angles using the gradient descent algorithm to minimize the error function. The error function encapsulates the distance between the end effector and the target and potential penalties due to near-collisions between joints.

Ensure to have the required targets and joint GameObjects correctly set in the Unity Editor for this script to work as intended.

Revision #3
Created 4 September 2023 12:54:32 by Matilda Fogato
Updated 18 October 2023 09:25:49 by Matilda Fogato