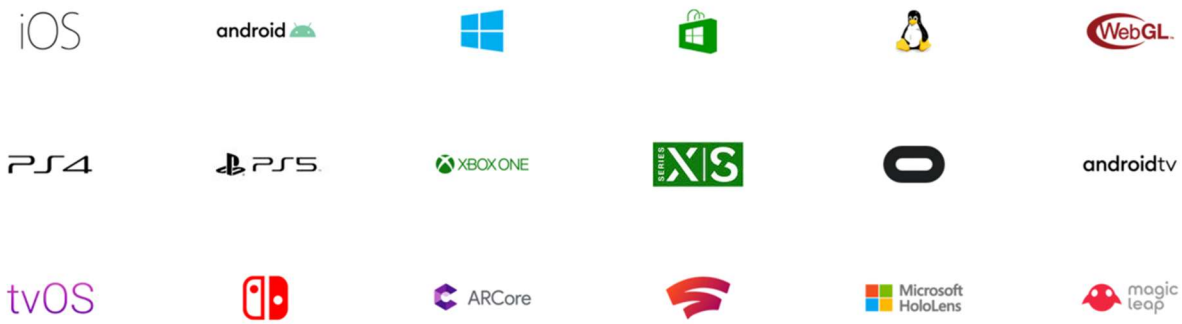
Simple Bicycle Physics

Documentation
Offered by AIKODEX

Simple Bicycle Physics package for Unity delivers a joint physics-controlled approach for bicycle simulation. This asset requires Animation Rigging v1.0.0+. Native 1.0.3. Inclusion of Animation Rigging package allows for an IK framework which allows a bipedal character avatar to ride any pedaled vehicle with appropriate posture and mechanics.

The Unity3D joint physics, controlled and corrected by quaternions, allows for a robust and majorly unbreakable physics system. The inputs are based on axes hence compatible with all platforms, including but not limited to, Windows (PC), Mac, Universal Windows platform (UWP), Linux Standalone, iOS, Android, ARKit, ARCore, Microsoft HoloLens, Windows Mixed Reality, Magic Leap (Lumin), Oculus, PlayStation VR, PS5, PS4, Xbox One, Xbox X|S, Nintendo Switch, Google Stadia, WebGL.



This package supports all render pipelines: Standard, High-Definition Render Pipeline (HDRP), Universal Render Pipeline (URP) and Scriptable Render Pipeline (SRP).

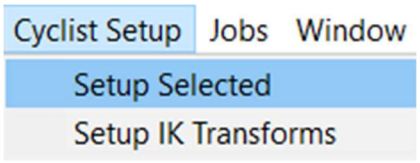
DETAILS

- **Animations:** Contains the Avatar controller for the cyclist and a few basic animations like idle, transition to riding and reversing. The included Unity's mechanim based animations are of humanoid type and apply to all humanoid characters.

- **Editor:** Contains a script which sets up the selected custom character in the hierarchy to ride the bike.

  To set up a custom character, select the GameObject with the bicycle controller script attached. Expand the hierarchy and scroll down to find the character. Select the character.

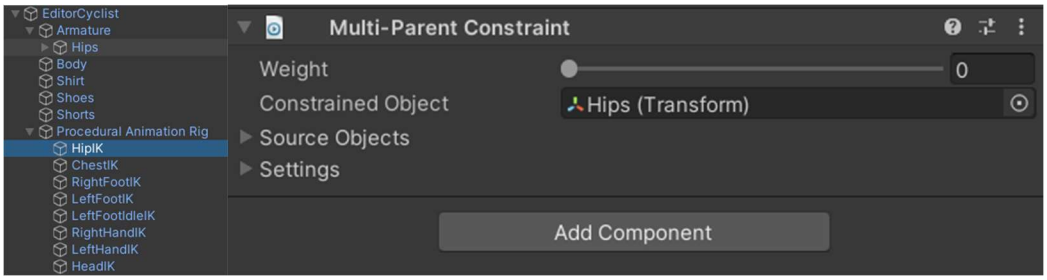
  With the character selected, click on Setup Selected.



  This will transfer the properties of the cyclist such as Transform, Animator Controller, Procedural IK Handler to your custom character in one click.

  Delete the original cyclist.

  Go through the Procedural Rig's gameObjects and fill in the inspector values with your custom character's armature parts.



Video Tutorial Guide: https://youtu.be/Ko_OtdNN6Os

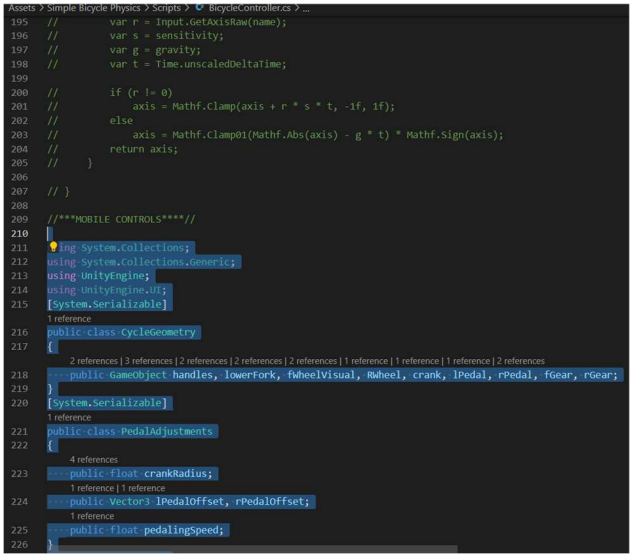
- **Example Scenes:** This folder contains different types of bikes and their configurations for you to test out.

- **Materials:** Contains materials (all standard shaders) compatible with all the three pipelines.

- **Mobile Controls:** Contains the relevant files to run on the iOS/Android Platform. To setup mobile controls, please follow the mobile setup instructions given below:
    - **Open Scene:** MobileBicycleStandardSetup.scene file

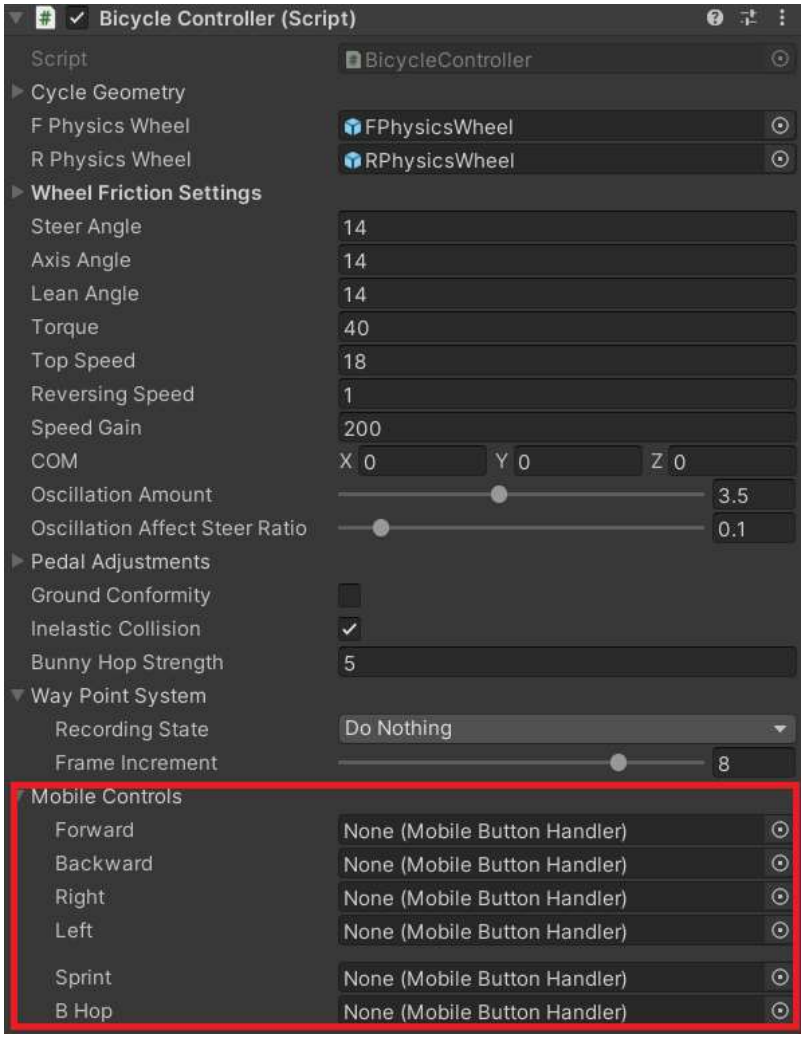
    - Click on the Bicycle Controller Script to open in the Visual Studio or any other IDE.

    - Comment out lines before //***MOBILE CONTROLS***// and comment the lines after //***MOBILE CONTROLS***//. To comment or uncomment highlight the text and press Ctrl + /



    - Set the screen resolution to 1920x1080 or the resolution of your device.

    - You can control the on-screen buttons with touch or mouse. Additional settings are present at the bottom of the bicycle controller script in the inspector:

- 

    - The Mobile Controls are automatically referenced when the developer enters play mode. Please note that we have used GameObject.Find(string s) just for the convenience of the developer in the Start() function.
    - Build the project for iOS/Android to test out the controls.

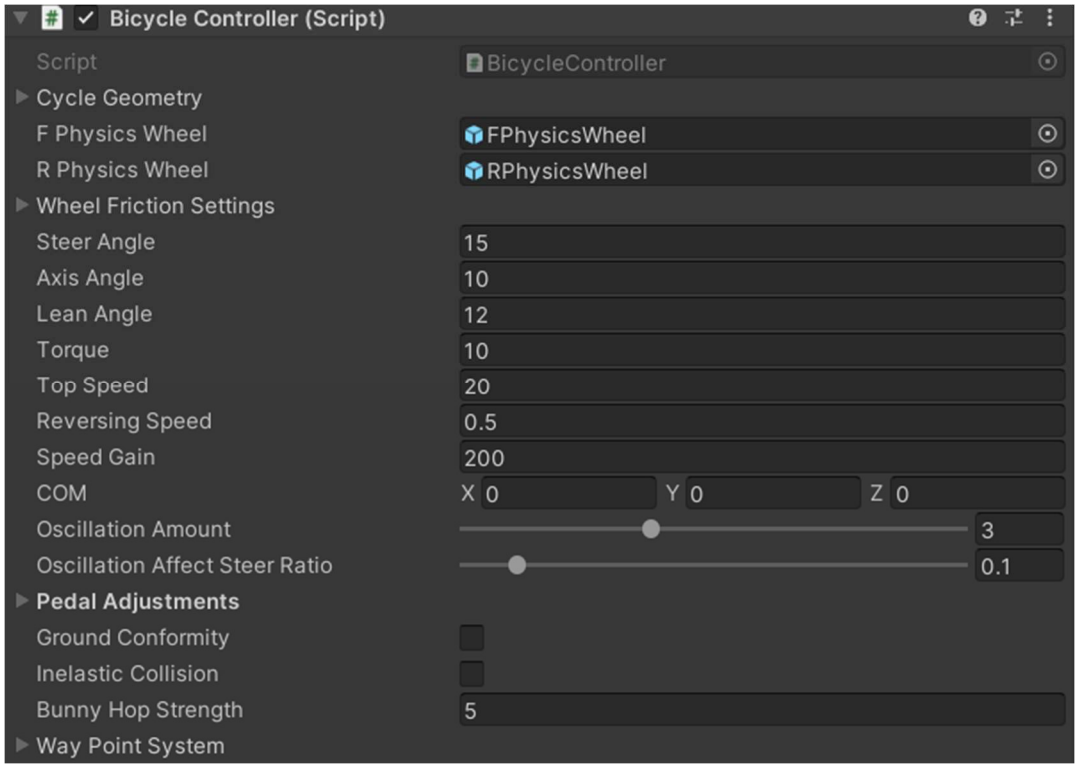
- **Models:** Contains models of different vehicle types that the cyclist can ride. These models are all separated on:

    1. Handles
    2. Lower Fork
    3. Front Wheel
    4. Rear Wheel
    5. Crank
    6. Front Gear
    7. Rear Gear
    8. Right / Left Pedals

- **Physic Materials:** Contains physics materials of tyres and terrains. Inspector settings available in Bicycle Controller.
- **Prefabs:** Prefabs of the completely setup cycles. Drag and drop the vehicle types.
- **Prototyping:** Contains test environment data, textures and models.
- **Scripts:** Elaborated in the section below
- **Textures:** Contains the textures [2K - 4K] used on the materials. Provides different jersey types and colors for you to choose from.

Scripts

- Bicycle Controller



(EXPANDED)

- **(Class) Cycle Geometry:** The cycle geometry class contains the gameObjects required to animate the bicycle and its control its movements. It accepts public members:

    1. Handles
    2. Lower Fork
    3. Front Wheel
    4. Rear Wheel
    5. Crank
    6. Front Gear
    7. Rear Gear

8. Right / Left Pedals

This class works intimately with the model's anchor/pivot positions and rotates it with the help of Quaternions. Please make sure the anchors and positions of any new bicycles that you want to integrate follow the sample bicycles.

- **F Physics Wheel:** This is a Unity physics joint configurable joint which has its positional motion locked on all axes and angular motion free on two axes: Rolling Forwarding + Turning. This physics body also has a Rigidbody component of a realistic mass attached to it. Uses sphere collider as it offers a great rotational curvature, better than that of a highly subdivided mesh.

- **R Physics Wheel:** This is a Unity physics joint configurable joint which has its positional motion locked on all axes and angular motion free on one axis: Rolling Forward. This physics body also has a Rigidbody component of a realistic mass attached to it. Uses sphere collider.

- **(Class) Wheel Friction Settings:** This class sets the physics materials on to the tires of the bicycle. F Friction pertains to the front tire friction and R Friction to the rear. They are of the Vector2 type. X field edits the static friction information and Y edits the dynamic friction. Please keep the values over 0.5. For more information, please read the commented scripts.

- **Steer Angle:** Controls the extent to which the wheel will turn from the mean position.

- **Axis Angle:** The amount of slant the turned wheel has. This value is purely visual. This value does not affect the physics.

- **Lean Angle:** This value corresponds to the visual body weight shift of the cyclist. Higher values contribute to a higher weight shift. This value does not affect the physics.

- **Top Speed:** Maximum Velocity magnitude the bicycle is capable of reaching.

- **Reversing Speed:** Maximum Velocity the biker can reverse with. This does not change the reversing animation speed.

- **Torque:** The strength with which the rider pedals. Higher values correspond to a higher acceleration amount.

- **Speed Gain:** Acceleration booster amount. Not physically accurate. Artificially inflates speed.

- **COM:** Center of Mass of the vehicle.

- **Oscillation Amount:** The degree of cycle waddling side to side upon pedaling. Higher values correspond to higher waddling. This property also affects character IK.

- **Oscillation Affect Steer Ratio:** Following the natural movement of a cyclist, an oscillation of the cycle from side to side also affects the steering to a certain extent. This value refers to the counter steer upon cycle oscillation. Higher values correspond to a higher percentage of the oscillation being transferred to the steering handles.

- **(Class) Pedal Adjustments:** Holds the properties that control the movement of pedals. The value in this class affects character IK.

    - **Crank Radius:** Radius from the center of pedaling. Higher values correspond to longer pedal radii.

    - **L/R Pedaling Offset:** Vector3 controlling pedal's local position.

    - **Pedaling Speed:** This value corresponds to how fast the cyclist is pedaling.

- **Ground Conformity:** For non-gyroscopic wheel systems like the tricycle, enabling ground conformity ensures that the tricycle is not always upright and follows the curvature of the terrain.

- **Inelastic Collision:** Attempts to Reduce/eliminate bouncing of the bicycle after a fall impact

- **Bunny Hop Strength:** The upward force the rider can bunny hop with.

- **(Class) Waypoint System:** This system is designed to create bots/multiple offline players riding the bicycle along with the player on the map. It is essentially a replay/ghosting system with real-time physics.

  How to Use:

  Step 1: Select **Record** from the recording state drop-down.

  Step 2: Enter the Play Mode and ride your bicycle around for a few seconds.

  Step 3: Select the bicycle controller GameObject in the hierarchy while in the Play Mode and go to Window > Save Bicycle Replay.

  Step 4: Exit out of the Play Mode and go to Window > Load Bicycle Replay

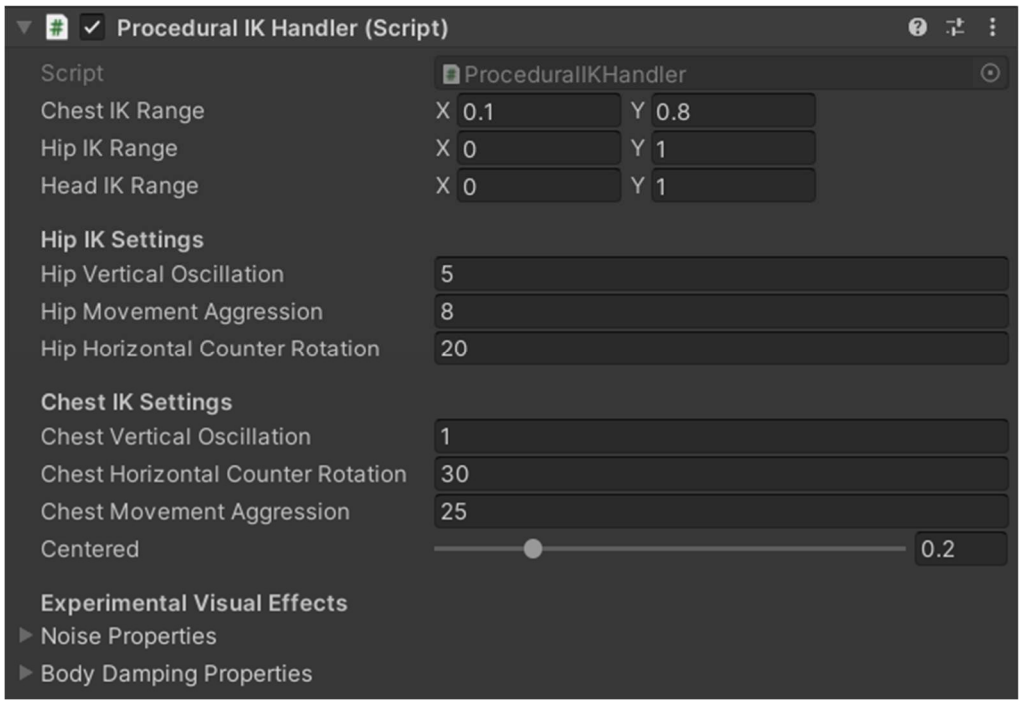
  Step 5: Change the recording state to **Playback** and enter play mode to test it out.

  Please Note: The replay is frame-rate dependent so please record with V-Sync on and play it back with the same.

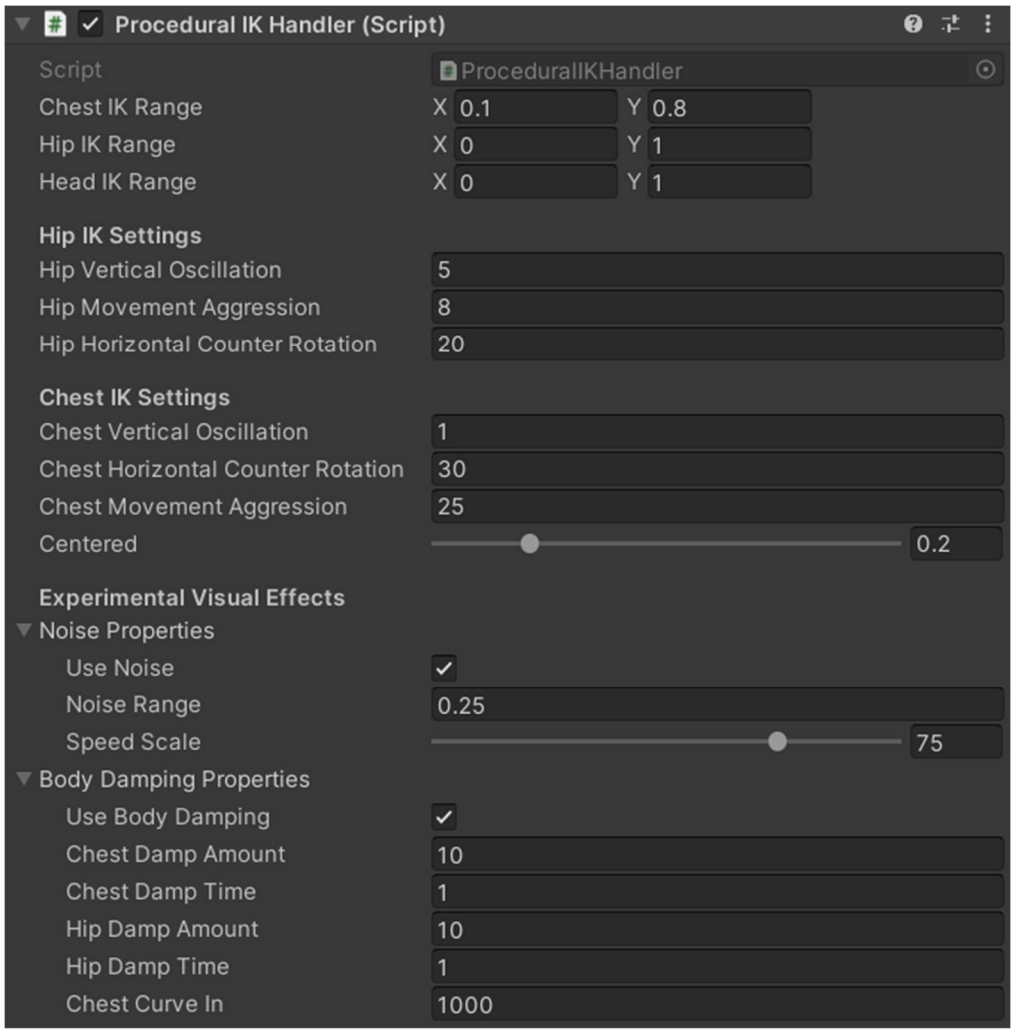
- **Procedural IK Handler**

  This script heavily depends on the animation rigging package and employs various IK

constraints to simulate the motion of the rider.



(EXPANDED)



- **Chest IK Range:** A Vector2 value which takes in the lower (X) and upper limit (Y) of the motion specified by the ChestIKTarget gameObject and modifies the weight of the movement. Weight refers to the percentage of movement between two points – very similar to Lerp.

- **Hip IK Range**: A Vector2 value which takes in the lower (X) and upper limit (Y) of the motion specified by the HipIKTarget gameObject in the prefab's hierarchy.

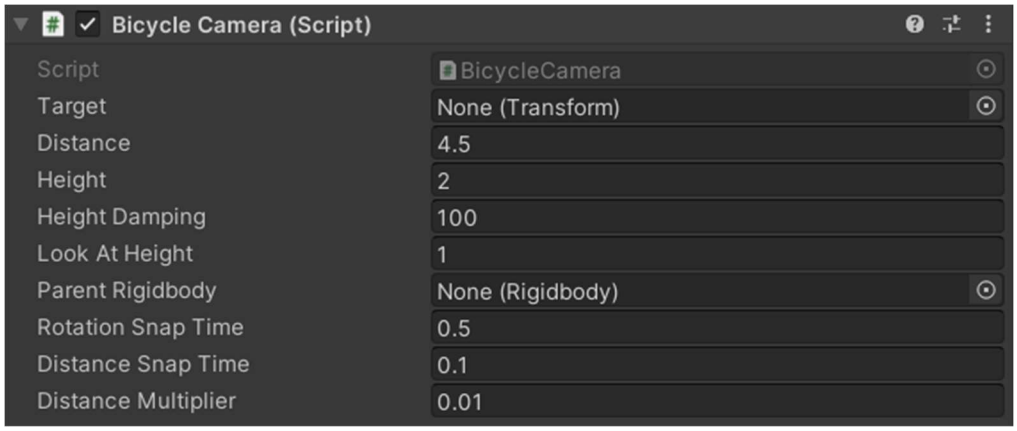
- **Head IK Range**: A Vector2 value which takes in the lower (X) and upper limit (Y) of the motion specified by the HeadIKTarget gameObject in the prefab's hierarchy.

- **Hip IK Settings** – Hip Vertical Oscillation: Refers to the up and down vacillation of the hip in response to pedaling.

- **Hip IK Settings** – Hip Movement Aggression: Refers to the rotational vacillation of the hip in response to pedaling.

- **Hip IK Settings** – Hip Horizontal Counter Rotation: Refers to the side-to-side vacillation of the hip in response to pedaling.
- **Chest IK Settings** – Chest Vertical Oscillation: Refers to the up and down vacillation of the Chest in response to pedaling.
- **Chest IK Settings** – Chest Movement Aggression: Refers to the rotational vacillation of the Chest in response to pedaling.
- **Chest IK Settings** – Chest Horizontal Counter Rotation: Refers to the side-to-side vacillation of the Chest in response to pedaling.
- **Centered:** Master control of Hip IK and Chest IK settings. Damps/exaggerates the natural movement. Keeps the cyclist in the center of the cycle.
- **Noise Properties (Class):** Provides visual balancing, quivering, shaking motion to the cyclist to enhance realism.
- **Body Damping Properties (Class):** Controls the bodily motion in response to the angle of the ground below. Higher values of chest and hip damping allow the body to act like shock absorbers.

- **Bicycle Camera**



- **Target**: GameObject to follow around.
- **Distance**: Horizontal distance from the camera
- **Height**: Vertical Distance from the camera
- **Height Damping**: Time taken for the camera to reach that height
- **Lookatheight:** Lookat offset for Height (0, Y, 0)
- **Parent Rigidbody**: Distance multiplier Calculations

- **Rotation/Distance Snap**: Time taken to reach specified rotation and distance values
- **Distance Multiplier**: Move away effect of the camera as the object speeds up

- **Perfect Mouse Look**

When the mouse moves the perfect mouse look script is activated and the user can look around with the mouse around the target object. When the movement is stopped, the rotation snaps back to the original position smoothly

Procedural Animations

With only three animation states governing the entire movement of the cyclist in our asset, there are a few procedural techniques we have utilized. These are noise, and prepositioned IKs that are controlled via weight. The pseudo procedural animations are a direct outcome of the animation rigging package. All the animations are based off Unity's mechanim humanoid animation clips. These clips are directly editable in the editor itself and are highly customizable.

If you have any questions, please feel free to contact us at info@aikodex.com.

Happy Cycling!
AiKodex