

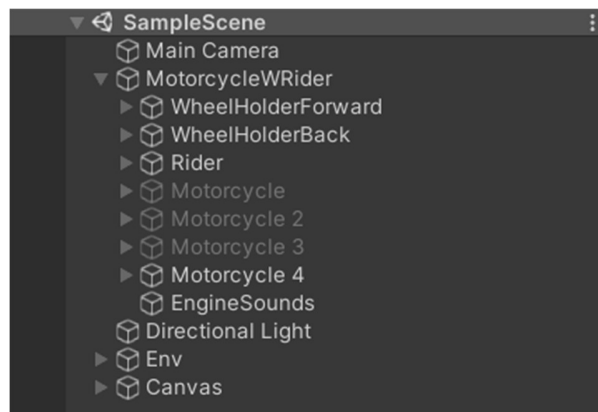
Simple Motorcycle Physics

Documentation

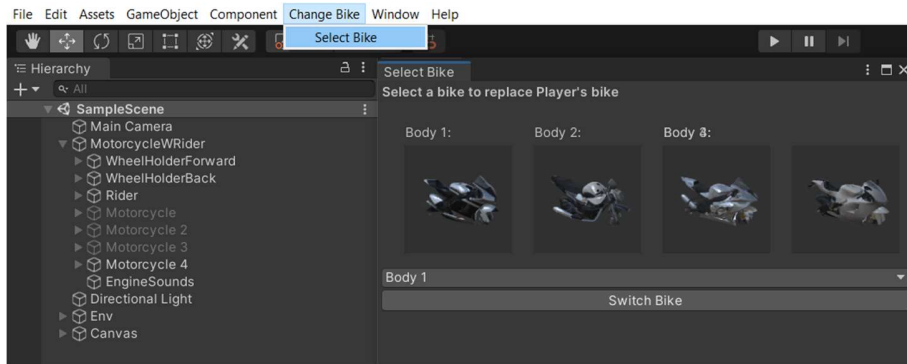
Simple Motorcycle Physics is based off of gyroscopic forces. The gyroscopic forces created by a moving wheel give it stability and help keep it upright. This package also makes use of WheelColliders offered by Unity for the construction of the motorcycle. Together with the properties of WheelColliders and gyroscopic forces, a plausible motorcycle simulation can be created. For better understanding and ease of usage and reproduction, there is only one C# script for controlling the vehicle. Alongside vehicular physics, there are a few moving parts in the bike such as the handles and the Rear Mudguard also accommodated in the script. Further, a fully rigged rider has been made available and an animation-based script for controlling the rider's movement has been included. An environment has been provided to test the motorbike's suspension and make the user comfortable with the controls.

Change Rider's Motorcycle:

On motorcycle 1 by default:



To change the motorcycle,



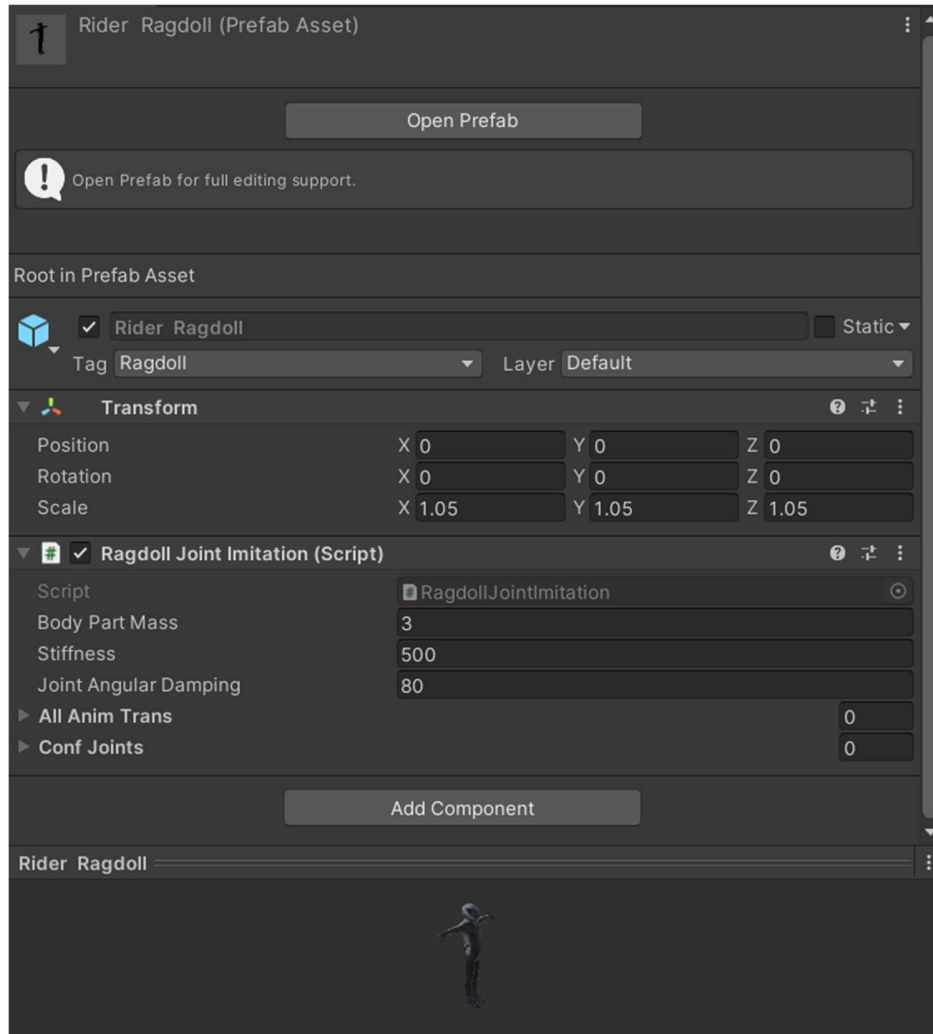
Select any Body for the Bike and click on Switch Bike.

(New) Active Ragdoll Components :

The active ragdoll in this asset is based off of blend tree animation found here <https://www.youtube.com/watch?v=LNidsMesxSE>

The ragdoll gets triggered by either pressing F or crashing into objects at high speeds. Ragdoll Resets by pressing R.

The ragdoll prefab can be edited in its prefab component window :



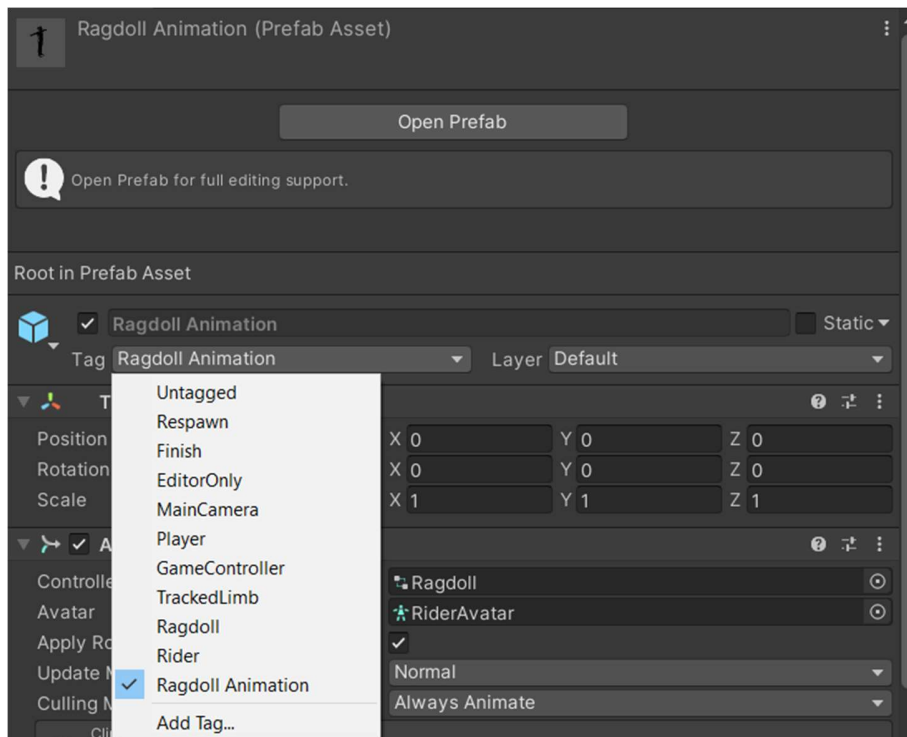
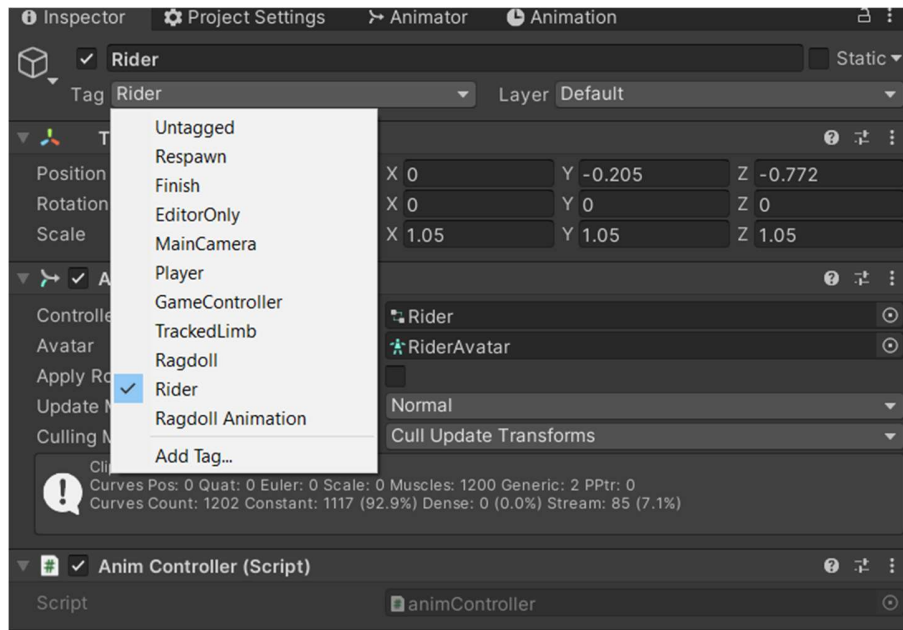
Upon entering the ragdoll mode, two gameobjects get instantiated and one gets destroyed. These are Rider Ragdoll, Ragdoll animations and Rider respectively. The camera switches focus from bike to ragdoll instantly.

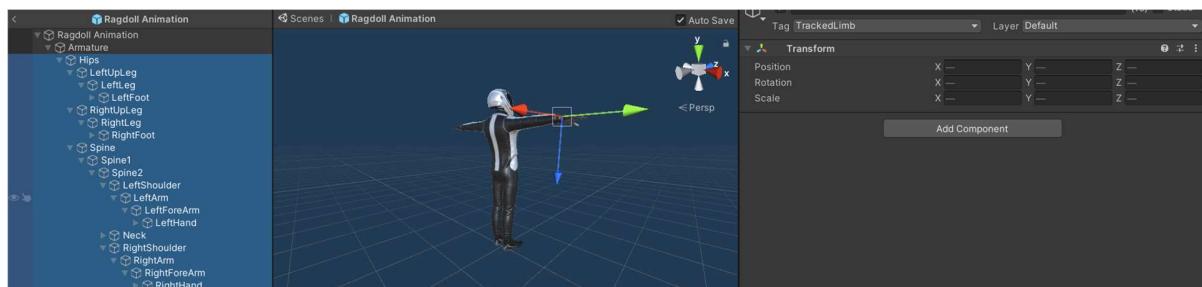
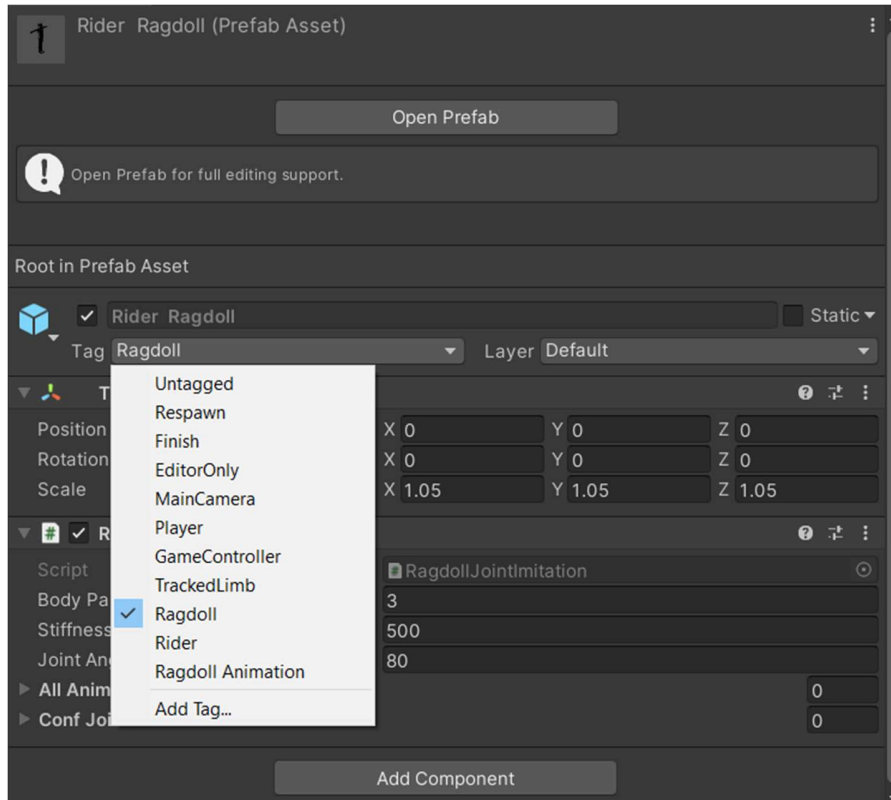
During the fall phase, the ragdoll tries copying the animations while physically following limp ragdoll physics. This movement can be considered a procedural animation or an active ragdoll.

The changes of the ragdoll are based on velocity and distance from the ground, but the user can add any other parameters to this equation as they please. Also, these animations are fully mechanim based and are editable.

The ragdoll is made of configurable joints that can be easily be altered. The script responsible for managing the ragdoll animations is RagdollAnimator.cs.

Please ensure your tags are properly set as below :





Summary

The ragdoll is first pose matched with the current limbs of the motorcycle rider. Then configurable joints are added. Target velocity is set to the limbs of the animated limb.

If you are experiencing any troubles with the ragdoll like limb separation or penetration of parts of the ragdoll into the ground please refer to the troubleshooting section

Video Tutorial Link

Package Components:

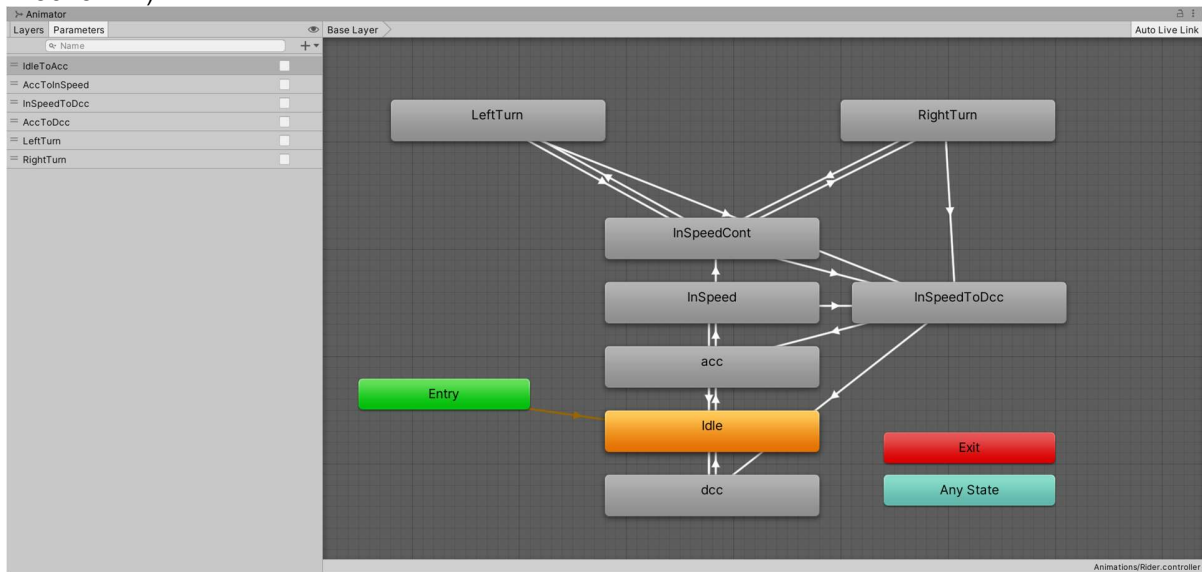
This package includes the following components:

Assets > Animations

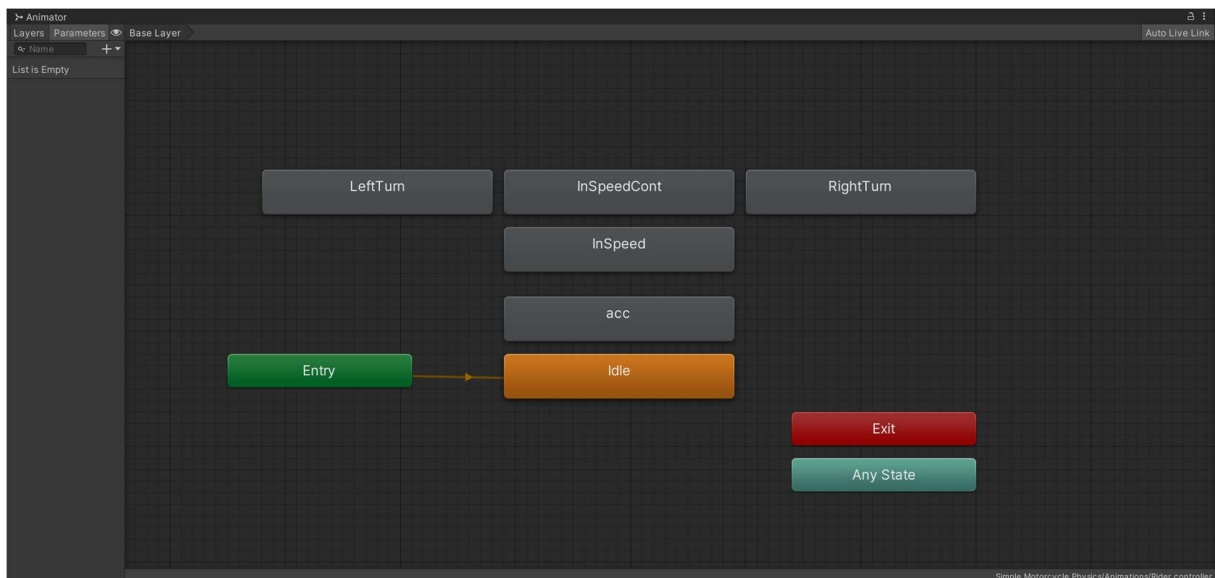


Animations

This folder contains all the rider's animations and are linked to the C# script animController. All the animations are implemented via the Animator window (formerly known as Unity mechanim).



(Old)



(New)

Easier using CrossFade

Materials

All the materials provided use standard unity shaders.

Models

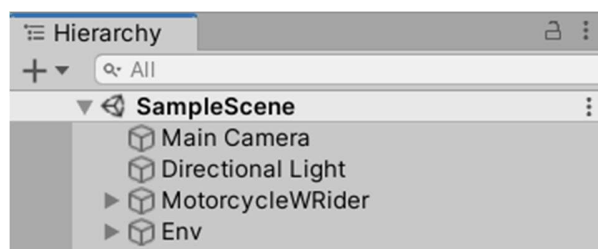
All the models provided true to scale and have their pivot set as the Center of Mass (COM), modelled in blender 2.92 exported as FBX. All models use external materials.

Prefabs

All the prefabs are set to location (0,0,0), rotation (0,0,0) and scale (1,1,1).

Scenes

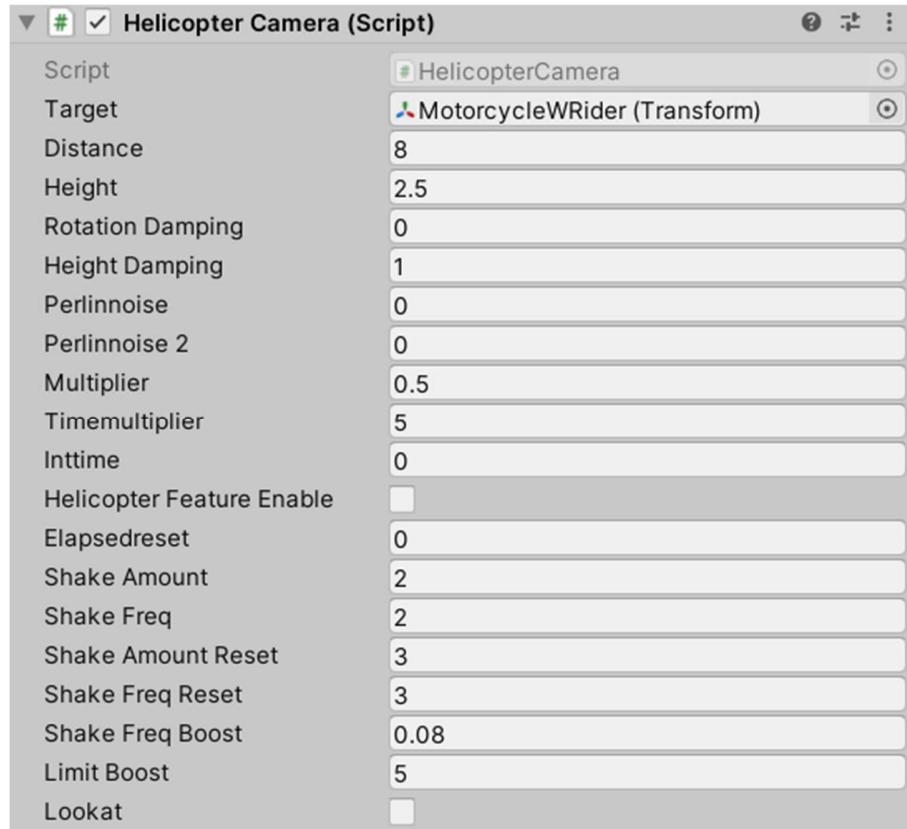
Sample .scene unity file is provided with contains the following GameObjects with a neat hierarchy



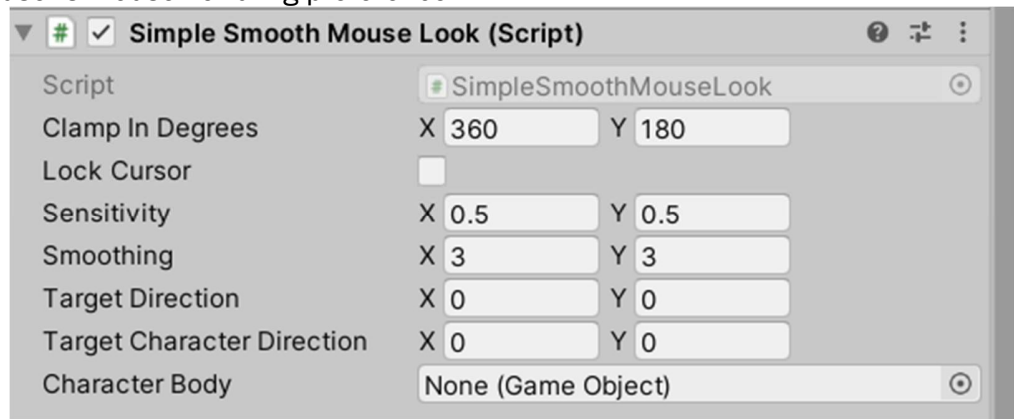
Scripts

Contains all the necessary scripts to run this motorcycle physics package :


1. **animController (C#)** : This script controls the behavior of the rider through exposed parameters inside the Animator window. (formerly known as Unity mechanim).
2. **HelicopterCamera (C#)** : This script is a custom camera script which not only follows the target around smoothly but also provides the user which a shake effect using Perlin noise and additional look at object feature to focus only on the object. To control the distance in-game please use keys "z" and "x" to zoom in and out, and mouse wheel to control the height of the camera.



3. **Simple Smooth Mouse Look (C#):** Takes an optional argument of a GameObject to rotate by default it is the main camera GameObject. Set clamp in degrees to restrict camera movement. Sensitivity and smoothing as per the user's mouse handling preference.



4. **MotorbikeController (C#) :** This script is to be attached to the motorcycle GameObject. Takes basic input public members such as the front and rear wheel colliders and wheel meshes. Handle system is used to control the handles of the bike which is paired with the steer angle of the front wheel. Rear Mud Guard is the rear mud guard GameObject present in the motorcycle body hierarchy. It is bound by the lookat quaternion function which points



toward the center of the rear wheel. Its offset is modifiable, but it is recommended to have it on 0,0,0.

Prevent fall algorithm works by tilt shifting the center of mass of the motorcycle if a critical angle is reached. It is defined in the function UprightForce in the c# script.

Can artificial stoppie or endo is balancing the motorcycle while front brakes are applied. Still an experimental feature. For low speed stunts. Works by shifting the center of mass of the motorcycle forward.

Stoppie or endo amount can be changed. This value determines how high the motorcycle will lift up during the procedure.

Max Steer angle : Maximum steering angle of the vehicle. Increasing this may cause the rider to tip over more often.

Max Motor Torque : Motor torque applied to both of the wheels of the motorcycle.

Max forward/Back brake : Separated brakes. Can be key coded in the script manually. (Requires beginner coding knowledge.)

Wheel Radius : Set locally which does not affect the wheel collider values. Please make sure this value coincides with the wheel collider radius to get an accurate simulation.

Steer Sensitivity : How quickly the rider is able to turn the vehicle. Increasing this may cause the rider to tip over more often.

Control Angle : The angle that the rider has control over. Gyroscopic functions such as (ω , ω^2) apply here.

Low/ High Speed : The steer angle is modified by these values for more accurate simulation at a variety of speeds.

New Additions :

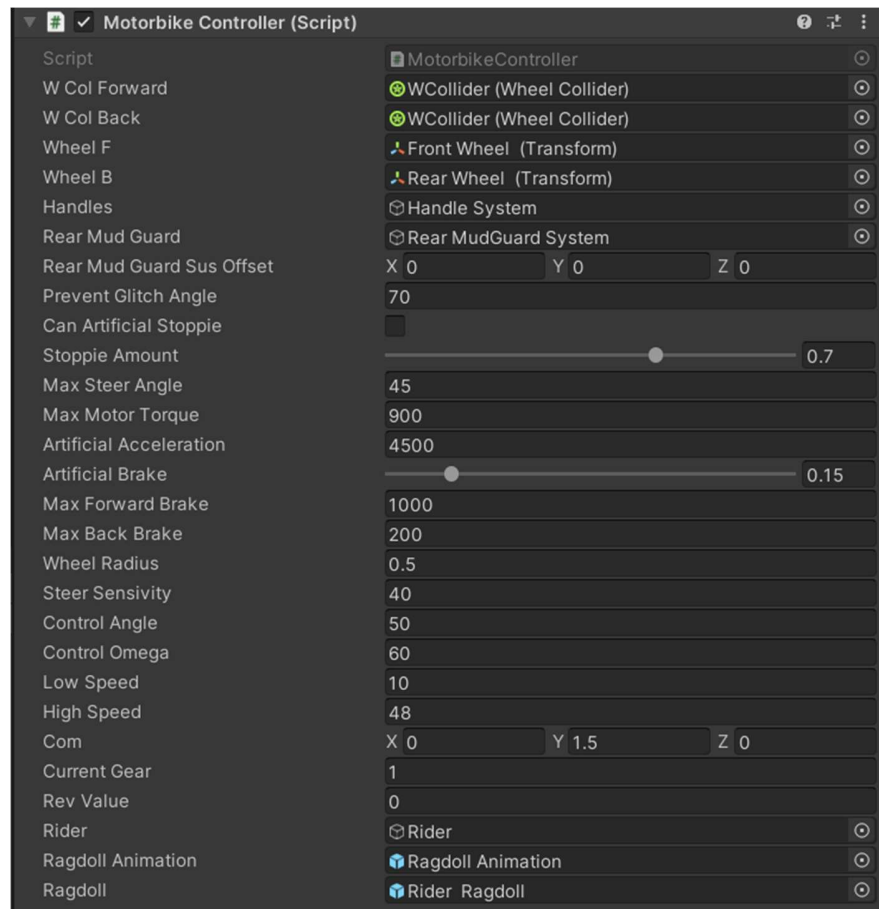
Artificial Acceleration and Brake : Uses Rigidbody add Force to speed up the bike otherwise not responding to increase in motortorque.

Current Gear and Rev Value : Denotes the current gear the motorcycle is on and the rev value is the engine speed. Automatic changing of gears.

Rider : Motorcycle Rider. Useful for enabling and disabling and replacing by ragdoll.

Ragdoll Animation. The animations get copied through this gameobject prefab.

Ragdoll : The ragdoll Prefab object container.

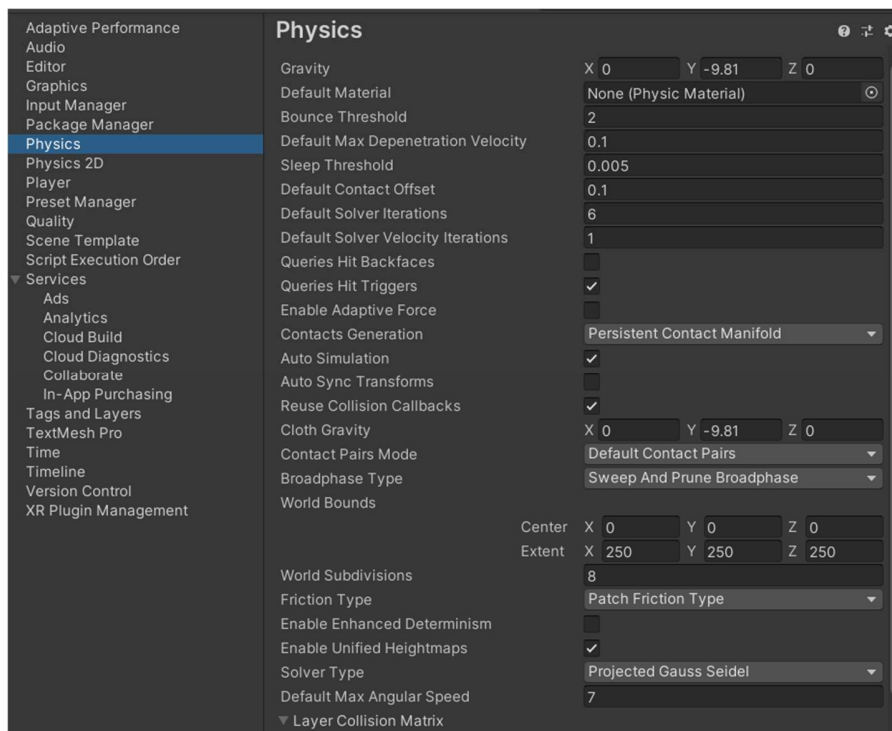
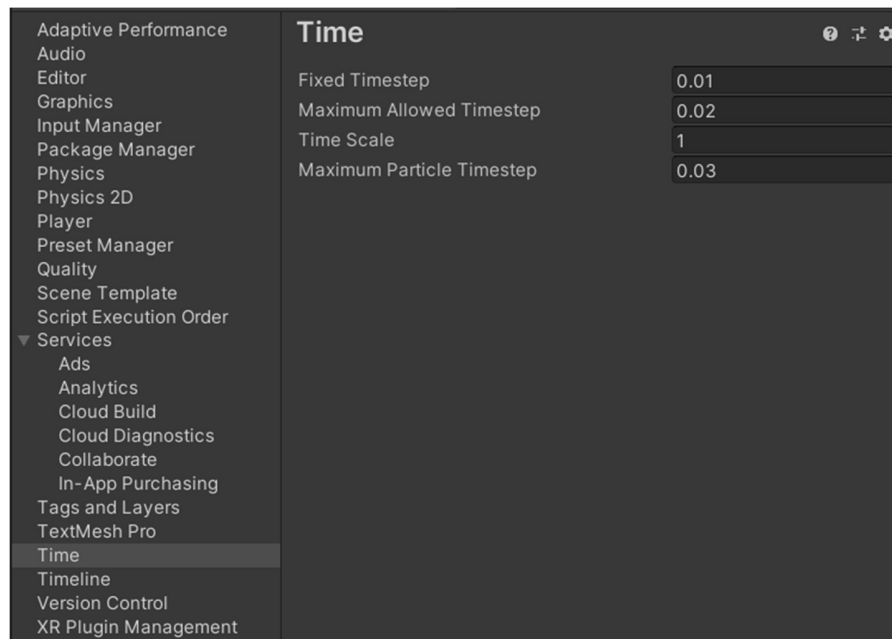


Troubleshooting

- If the steering is vibrating, try configuring the low and high speeds given in the MotorbikeController script.
- Try increasing the physics Fixed Timestep in the Project Settings in the Time Tab.
- If the radius of turning of the vehicle is too large, try setting the steer sensitivity to a higher value.
- If the rear suspension is facing the other way or is in an incorrect position, try changing the offset values.
- If Unity crashes after the motorcycle hitting a cone, disable the convex checkbox in mesh collider component attached to the cones.
- If the wheel is vibrating while resting, try changing the extreme slip value in the wheel collider settings.

Ragdoll

- If the limbs are get separated or penetrate into the ground or deform, please use these settings below.



Try lowering the Fixed Timestep to a lower value (<0.02), Default Max penetration to a smaller value (<0.1), Default contact offset to a greater value (>0.1).

Try changing the `RagdollJointLimitation.cs` Script value solver iteration to a lower value. Found here :

```

77     }
78     allAnimTrans = transList.ToArray();
79
80     foreach (Transform trans in ragTransArr)
81     {
82         ConfigurableJoint cj = trans.GetComponent<ConfigurableJoint>();
83         if (cj != null)
84         {
85             //default contact to 0.1, max depenetration to 0.1 Fixed TimeScale to 0.01
86             jointList.Add(cj);
87             cj.projectionMode = JointProjectionMode.PositionAndRotation;
88             cj.projectionDistance = 5f;
89             cj.projectionAngle = 5f;
90             cj.enablePreprocessing = false;
91             trans.GetComponent<Rigidbody>().solverIterations = 4;
92             trans.GetComponent<Rigidbody>().mass = bodyPartMass;
93             trans.GetComponent<Rigidbody>().velocity = Motorcycle.GetComponent<Rigidbody>().velocity * 1f;
94         }
95     }
96     confJoints = jointList.ToArray();
97 }
98
99 1 reference
100 private void AddJointFollowScript()
101 {
102     foreach (ConfigurableJoint cj in confJoints)
103     {
104         cj.gameObject.AddComponent<RagdollJointConfig>();
105         cj.connectedBody.collisionDetectionMode = CollisionDetectionMode.Continuous;
106         for (int t = 0; t < allAnimTrans.Length; t++)
107         {
108             if (allAnimTrans[t].name == cj.gameObject.name)

```