Vehicle Development Basics

Basics of vehicle development for VRChat worlds using UdonSharp

- Physics "Hacks"
  - Improve Slide Stability With Modified Rigidbody Interia Tensor

Physics "Hacks"

Useful workarounds for improving the handling of vehicles

Improve Slide Stability With Modified Rigidbody Interia Tensor

FOLLOWING PAGE CONTAINS MATERIAL THAT IS WORK IN PROGRESS!

Modifying inertia tensor is a useful "hack" for improving the slide stability of vehicles that use Unity's native WheelColliders. If you are developing your own wheel collider solution, this workaround shouldn't be necessary.

What is rigidbody inertia tensor?

> Inertia tensor is a rotational analog of mass: the larger the inertia component about a particular axis is, the more torque that is required to achieve the same angular acceleration about that axis.
>
> Rigidbody.inertiaTensor.html - Unity Documentation

In simple terms: higher inertia tensor increases the resistance to changes in angular velocity, which in this case helps us mitigate the issues with the default wheel collider behaviour which can be unpredictable when entering and exiting a sideways sliding state, which can cause severe issues with the vehicle’s handling and cause the vehicle to spin out.

By increasing the inertia tensor, the vehicle essentially “feels” heavier on the physics’ end due to increased resistance to angular velocity changes while still preserving the original mass.

> Check out the official documentation for rigidbody's interia tensor:Rigidbody.inertiaTensor - Unity Documentation

How are we going to apply the modified inertia tensor?

In order to properly implement this solution, we're going to need following elements:

- Expected maximum sideways velocity of the vehicle (Float)
- Inertia tensor multiplier (Vector3)
- Original inertia tensor of the rigidbody (Vector3, cache at Start)
- Modified inertia tensor (Original inertia tensor * Inertia tensor multiplier, cache at Start)

Example Code

```
/// <summary>
/// Maximum expected lateral velocity magnitude of the vehicle
/// </summary>
[SerializeField]
private float lateralVelocityRangeMagnitude;

/// <summary>
/// Multiplier for the vehicle's rigidbody's inertia tensor at maximum slide velocity
magnitude
/// </summary>
[SerializeField]
private Vector3 inertiaTensorMultiplier = new Vector3(1.5f, 4f, 1.5f);

/// <summary>
/// Relative velocity of the vehicle's rigidbody
/// </summary>
private Vector3 velocity;

/// <summary>
/// Rigidbody of the vehicle
```

```
/// </summary>
private Rigidbody rb;

private void Start()
{
  // Store the rigidbody of the vehicle
  rb = GetComponent<Rigidbody>();

// Store the original intertia tensor
originalInertiaTensor = rb.inertiaTensor;

  // Calculate the modified intertia tensor based on the original and the multiplier
    slideInertiaTensor = Vector3.Scale(originalInertiaTensor, inertiaTensorMultiplier);
}

private void FixedUpdate()
{
  // Cache the relative velocity of the rigidbody
velocity = transform.InverseTransformVector(RB.velocity);

UpdateBodyIntertiaTensor();
}

/// <summary>
/// Updates the vehicle's rigidbody's inertia tensor based on the sideways velocity to
increase slide stability
/// </summary>
private void UpdateBodyIntertiaTensor()
{
  // Lerp between the default and modified inertia tensor and apply the result on the
rigidbody
    rb.inertiaTensor = Vector3.Lerp(originalInertiaTensor, slideInertiaTensor,
Mathf.Abs(velocity.x / lateralVelocityRangeMagnitude));
}
```