

Portfolio document - Siobhan Fitzgerald-Gibson

Concept: Introduction to rigid body physics (Unity Technologies)

Portfolio notes

Unity follows a DITA-like content model, splitting topics into tasks, concepts, reference, and landing pages. This **concept page** introduces a real-world physics concept, and its simulation equivalent.

Rigid bodies are a fundamental part of real-world physics and physics simulation. The ideal outcome of this page is that the user knows what a rigid body is in real life, and understands how it relates to a “Rigidbody component” in Unity.



Introduction to rigid body physics

In real-world physics, a rigid body is any physical body that does not deform or change shape under physics forces. The distance between any two given points of a rigid body remains constant in time, regardless of external forces exerted on it.

To simulate physics-based behavior such as movement, gravity, collision, and joints, you need to configure items in your scene as rigid bodies. To configure GameObjects as rigid bodies in Unity's PhysX system, you can assign them the [Rigidbody](#) component. The Rigidbody component is represented in the API by the [Rigidbody](#) class.

Rigid body GameObjects with physics-based movement

In Unity, a [Rigidbody](#) component provides a physics-based way to control the movement and position of a GameObject. Instead of the [Transform](#) properties, you can use simulated physics forces and torque to move the GameObject, and let the physics engine calculate the results.

In most cases, if a GameObject has a Rigidbody, you should use the Rigidbody properties to move the GameObject, instead of the Transform properties. The Rigidbody properties apply forces and torque from the physics system, which change the GameObject's Transform; if you then change the Transform directly, Unity cannot correctly calculate the physics simulation, and you might see unwanted behavior. This is particularly true of rigid bodies with [Joints](#).

Unity handles physics-based movement globally, not locally. When a GameObject with a Rigidbody moves via physics-based movement, it moves independently of any parent or child GameObject. A child GameObject that has a Rigidbody still uses its parent GameObject to define its local position for initialization, but Unity calculates its physics-based movement in global space.

To control a Rigidbody via script, the primary methods are [AddForce](#) (to add forces to a GameObject) and [AddTorque](#) (to apply torque to a GameObject).

Rigid body GameObjects without physics-based movement

There are some cases where you might want the physics system to detect a GameObject, but not to control it. For example, you might want [Colliders](#) to detect a GameObject, but you intend to control that GameObject's movement and position via its [Transform](#).

Movement in Unity that is not physics-based is called kinematic motion. The Rigidbody component has the property **Is Kinematic** which, when enabled, defines the attached GameObject as non-physics-based, and removes it from the control of the physics engine. This allows you to move it kinematically via the Transform without Unity's physics simulation calculations overriding the changes.

A kinematic Rigidbody can apply physics-based force to physics-based Rigidbody GameObjects, but does not receive physics-based force. For example, a kinematic Rigidbody can collide with and "push" a Rigidbody that has physics-based movement, but a Rigidbody with physics-based movement cannot "push" a kinematic Rigidbody.

If you use a [Joint](#) to attach a kinematic Rigidbody to a non-kinematic Rigidbody, the Joint cannot exert forces to move the kinematic Rigidbody. It can only ever move the non-kinematic Rigidbody. However, you can still move the kinematic Rigidbody via the Transform, and the Joint can adjust the pose of the non-kinematic body to satisfy the joint limits.

Rigidbody optimization

When a Rigidbody moves at a slower speed than the **Sleep Threshold** (see the [Physics](#) Project Settings), Unity sets the Rigidbody to "sleep", which means that the physics system does not include it in physics calculations. When a sleeping Rigidbody receives a collision or force, Unity "wakes up" the Rigidbody and continues to include it in physics calculations.

By default, the sleeping and waking of a Rigidbody component happens automatically. However, you can also control this behavior yourself via script, via the methods [Rigidbody.Sleep](#) and [Rigidbody.WakeUp](#).

A Rigidbody might fail to wake up in response to movement and collisions from static colliders (that is, colliders without a Rigidbody) that are moving via the **Transform** position instead of the physics system. This is particularly likely if the physics system can no longer detect the static collider. If this happens, you can use [Rigidbody.WakeUp](#) to wake up the sleeping Rigidbody.

To learn more about Rigidbody component sleeping and how to optimize your Rigidbody components, refer to [Use Rigidbody sleeping to improve physics performance](#).

For more information on how the PhysX physics system handles sleeping, see the NVIDIA PhysX SDK Rigidbody Dynamics documentation on [Sleeping](#).