Physics Simulations in Unreal

Unreal Engine 4 has a plethora of tools available to developers for creating realistic mechanics that mimic that of the real world. Where it goes above and beyond however, is the deep customization of these tools that allow for creating something truly unique. The system uses the PhysX 3.3 engine as a framework for all collision and physical interactions between objects and is actively being expanded upon. [1]

Just about every actor type provided in Unreal has a toggle-able option in the details panel to enable physics. Enabling automatically applies gravity, assigns a mass based on what Unreal determines that specific object should have, and provides you with the default settings for all other necessary computation. All variables also come with the option to be further tweaked by the developer. This panel gives you access to such tools as linear and angular damping, allowing you to add a theoretical frictional force applied to slow the displacement and rotation of the object. Another variable that is frequently used are constraints which allow the restriction of which axis or plane the object can move along. Expanding the menu allows you to change the center of mass offset, the mass scale and gives access to the wake/sleep functionality. Assigning a physics object to be “asleep” allows you to have it act as a non-physics object until a certain force is applied to the object. This comes with two unique events that allow you to customize what happens upon waking up and being set to sleep.

UE4 makes applying basic force incredibly simple for the developer. Most computation is done through pre-set functions, the most used of which is aptly named “Add Force”. This function will use the amount of force taken in as well as the mass specified in the details tab of the object the force is being applied on to calculate the acceleration being applied. Physics thrusters are another easy-to-implement component that can be added to an actor to achieve basic force application.

Physics constraints allow the connection of two meshes with the ability to restrain angular and linear motion, leading to the ability to create highly customizable joints between two objects. Using the pre-built physics constraint actor provided by Unreal allows easy connection between two actors with static meshes in the world map, with the point they are constrained between being where the physics constraint is placed. While not allowing as much customization as alternatives, this provides a quick and easy way to rig up actors in a scene. For added functionality, the physics constraint component can be placed inside of an actor’s blueprint and allows for connections between components. Implementing this way also provides which socket you’d like to specify if your components have skeletal meshes and are parents of the physics constraint.

Physical Animation is a component that can be added to actors who have a skeletal mesh. This is done to achieve a realistic weighted effect when collision is detected with another mesh or when force is applied. To do this, first a physics asset blueprint will need to be created, and inside of it all bones will need to be assigned to a physics profile. Then by using “Set Skeletal Mesh Component” Apply Physical Animation Settings Below” you can apply force to create a rag-doll effect on the actor that is dependent on which bone you specify as the root of the changes. This allows for dynamic animation that changes depending on which forces are being applied to the actor at any given point during the runtime of the game.

This engine is responsible for many of the components/functionality provided in UE4 and allows for the simulation of physical force being applied in various ways and for complex collision detection between actors. The understanding of how these are being utilized is vital for any project and proper use of the methods can yield truly stunning, realistic results.

References

1. [Unreal Documentation on Physics Simulations](https://docs.unrealengine.com/en-US/Engine/Physics/index.html)
2. [Unreal Documentation on Physics Constraints](https://docs.unrealengine.com/en-US/Engine/Physics/Constraints/ConstraintsUserGuide/index.html)
3. [Video course on Physical Animation component](https://www.youtube.com/watch?v=6pG9SZQ0o7w&feature=youtu.be)
4. [Video course on Physics Constraint component](https://www.youtube.com/watch?v=bMEp9tAcisU&feature=youtu.be)