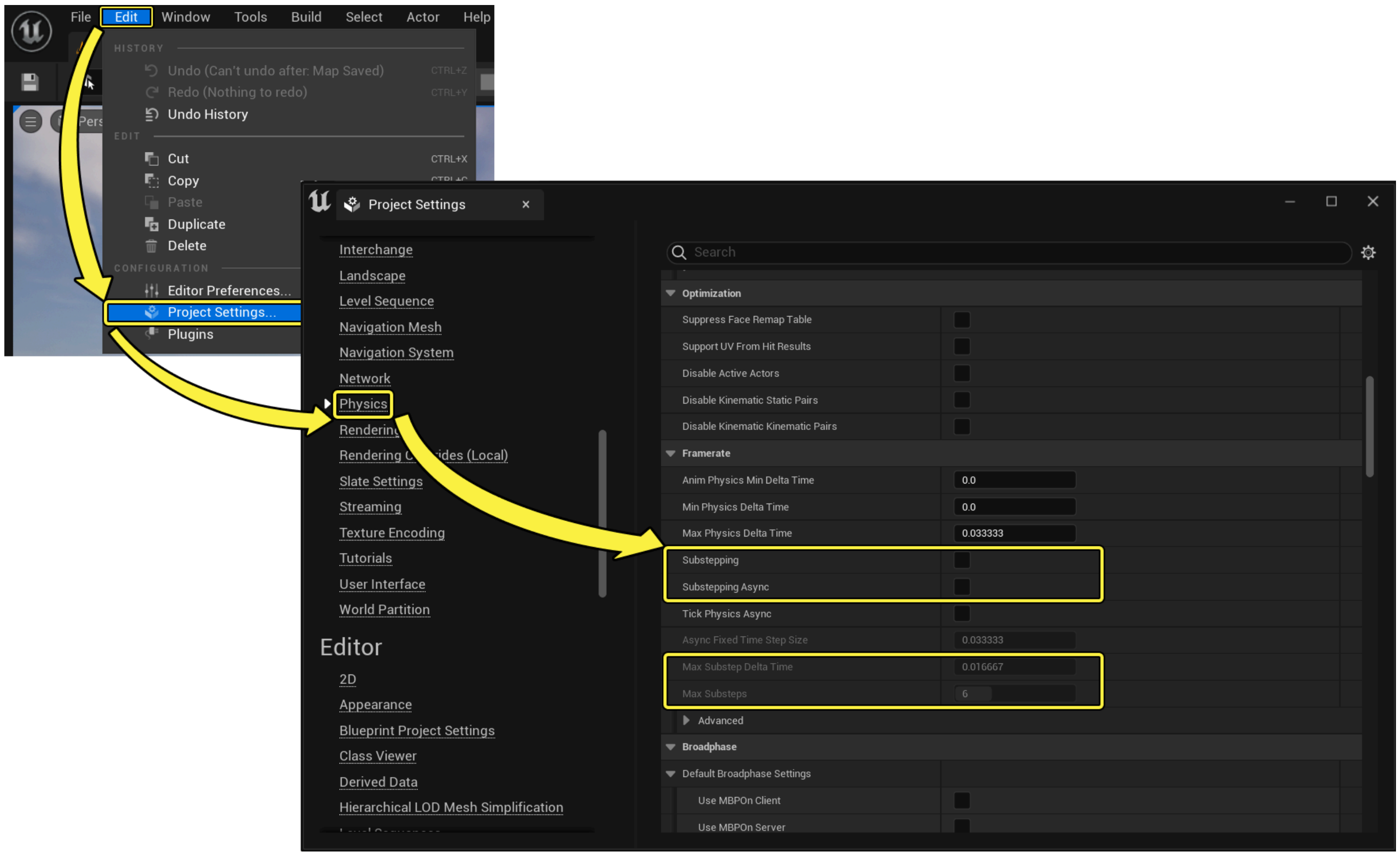
Physics Sub-Stepping

Explanation of what Physics Sub-Stepping is and when to use it.

By using physics **Sub-stepping** you can get physics simulations that are more accurate and stable, however, this comes at the expense of performance. The most noticeable improvement will be with the ragdoll jitter and other complex physical assets.

Enabling Sub-Stepping

Sub-stepping can be turned on in the **Edit** > **Project Setting** > **Physics** tab:



Optimization	
Suppress Face Remap Table	☐
Support UV From Hit Results	☐
Disable Active Actors	☐
Disable Kinematic Static Pairs	☐
Disable Kinematic Kinematic Pairs	☐
Framerate	
Anim Physics Min Delta Time	0.0
Min Physics Delta Time	0.0
Max Physics Delta Time	0.033333
Substepping	☐
Substepping Async	☐
Tick Physics Async	☐
Async Fixed Time Step Size	0.033333
Max Substep Delta Time	0.016667
Max Substeps	6
Advanced	
Broadphase	
Default Broadphase Settings	
Use MBPOn Client	☐
Use MBPOn Server	☐

This feature is still a work in progress and is not fully supported by APEX Destruction. It is also currently not supported on mobile devices. However, we are working towards this as it can greatly improve the look and feel.

Property	Description
Substepping	This enables or disables sub-stepping in your current project.
Substepping Async	Whether to substep the async physics simulation.
Max Substep Delta Time	This is the maximum time, in seconds a sub-step is allowed to take. So, if a full step takes .05 seconds, and **Max Substep Delta Time** is set to .025, the full step will be split into 2 sub-steps. Now, if the full step were to take a substantial amount of time, say 2 full seconds, then the step will be evenly split based on the value in **Max Substeps**, and not **80** `(2.0/0.025=80)` sub-steps. It is worth noting that **Max Physics Delta Time** limits how long a physics step is permitted to take.

Property	Description
Max Substeps	The maximum number of sub-steps a full step is permitted to be broken into.

Technical Details

Unreal Engine 5 uses a variable frame rate. While this is good for hardware scalability, it creates a challenge for the physics engine, which works best with small fixed time steps. Sub-stepping takes the total frame time and divides it into sub-steps. The physics simulation is then ticked multiple times per frame. The number of sub-steps taken will depend on how small your max sub-step delta time is set to. The smaller the max sub-step time the more stable your simulation will be but at a greater CPU cost.

Sub-stepping is completely hidden from the user, which means that some calls to the physics engine have to be interpolated or maintained. For example, if a user applies a force to an Actor for a single frame, and the frame is internally sub-stepped N times, we need to apply the force for N consecutive simulation steps to achieve the same acceleration. Similarly, if the user sets the target location of an Actor, we must interpolate the target location over multiple sub-steps to maintain the desired speed. All of these details are already handled internally by Unreal Engine, but there are some CPU and memory costs associated with the required bookkeeping.

Another technical detail to be aware of is the way collision callbacks behave while sub-stepping. Unreal Engine runs the physics sub-stepping in a separate physics thread, allowing the game thread to continue doing work. To get the best performance, we delay the collision callbacks until the final sub-step is done. This means that you can get several callbacks for the same collision. For example, if A collides with B and bounces off, you may get a callback for A and B overlapping AND a callback for A and B no longer overlapping in the same frame. Internally all callbacks are pushed into a queue and so callbacks from sub-step 1 will be processed before callbacks from sub-step 2.

Uses

Sub-stepping will significantly improve the quality of complex physics assets, such as those used for a rag doll. However, you should always weigh the improvement in quality to the CPU cost, and what is truly important to your game.