Research

Research done for IMT&S by Tycho Tuitert – 2023-2024

Introduction

During IMT&S research has been conducted to allow a smooth workflow and to accelerate the progress opposed to purely relying on trial and error. Many topics covered during IMT&S, most of which are covered in blogs, were new to me and thus required some background research. This post/document will give an insight into the most important topics covered. Some of them might seem small, but still took a lot of time to research and are thus worth mentioning, as it may prevent encountering these problems in the future. This post is created to fill in the research gaps from the previous IMT&S posts, as research (and more importantly the sources) were not covered to the extent that IMT&S requires.

Methods

Multiple research methods have been conducted during the period of IMT&S. The main methods were desk research & experimentation, a combination of secondary and primary sources. Under desk research I mainly categorize the different tutorials used, the forum posts referenced and all the documentation Unreal offers itself. Experimentation here entails trying to fix an issue by either trial and error, or by referencing the code in the engine. The latter can not be covered by stating sources and is thus only mentioned if it solved the problem. The final research method could be categorized as both “Observation” and “Interview”, but both are technically just play-tests. We did playtesting with strangers from our target audience, but mainly did in-house playtesting (we also fit the target audience). From these playtests many issues came to light, but only some required further research, thus not all are mentioned here. As for the research models I applied the double diamond throughout most of the project, in combination with applying design thinking in the project as a whole. This document will mainly be created in a journal format, whereas my blogs are more shaped as the double diamond. The research topics in this document do all keep the general flow of the double diamond which consists of: discovering and stating the problem, defining the needs, developing different solutions, or testing out ways on how to fix it and the final solution which could be viewed as deliver.

Research

This research will entail the most important and though to find solutions to problems. Some were solved, some did not have any solutions. I still mention those problems as they were crucial in my experience with Unreal and were still solid research points. I will state most sources referenced for each research post, but some were either not used or did not pose any solid solution and will thus not be included.

Research – IK rig

The IK (inverse kinematic) rig was one of the first things researched, and one of the final major aspects finished. The IK rig serves as a full body which can be controlled with the controllers, thus adding a body with the only input being the head and the hand positions. Compared to Unity, the implementation of a full body IK rig in Unreal is more complicated and works way less reliably. I tested out many different plugins and methods, although some only had demos available publicly. The main plugins tested were UBIKSolver [3], PowerIK [5], the built-in FBIK tool from Unreal [1] and finally UnrealBodyPlugin [11]. Each had their own issues, with most of them being very outdated and

not giving a proper IK rig which can be altered easily. Please note that a lot of research was done in September and was thus in my early days of Unreal and the project, so some plugins might work now that I have more experience in the program (and with IKs). I also tried out several Youtube tutorials [4,6,7], but each had their own issues with either head rotation being wrong, snapping rotation or no animation inclusion.

After many attempts I got something to work with the FBIK from unreal itself. But it still had many issues, with mainly rotation being unreliable at each angle. Instead of trying to solve it, I worked on other prototypes and picked up the subject at a later stage.

It was continued back in November, which was also when I found the UnrealBodyPlugin [11]. It was already made for unreal engine 5, and seemingly kind of worked well. It also had some documentation for the VRE (vr expansion plugin), which was a bonus. I implemented it loosely and immediately saw issues. This time however, I had more experience in Unreal and figured I would fix the plugin myself.

This was a great point for two of my learning outcomes, as I both got to work with C++ and IK-rigs, both of which were important to me. The main issue was that the body was placed in world position and was thus detached from the main player at the start. This would lead to the body “floating” behind the player, as it could not keep up with the quick movements. There were variables to solve this, but they caused everything to get choppy (as it would just move it to the player faster). I rewrote everything to work locally, which ended up to me rewriting most of the backbone of how everything moves. I furthermore added some extras to make it more tweakable, such as adding variables to offset the body compared to the head to make it work with the PC controller (non-vr).

The plugin was afterwards sitting in a separate project for a while and was finally implemented properly in January 2024. Upon implementation however, many more bugs were discovered and were fixed by implementing a combination of trial-and-error and experimentation. The issues, also covered in my blog on the matter, consisted of rotation issues where everything would snap as they would reach from -180 to 180 degrees and other issues where rotations were incorrect or behaving weirdly.

I tried debugging everything with the help of prints and other manners and tried rotating it differently by for example using look ats instead of direct 1-to-1 rotation setting, but they seemed to not work. I ended up making something work by fixing the small issues and leaving out some other features. The rotation snapping was fixed by simply using a rotation lerp function instead of lerping the angle, as the rotation slerp has a built-in closest rotation setting. The incorrect rotation was not fixed completely and was instead upon a trial-and-error basis brought to a point which worked but was lacking features. The plugin offers looking side-to-side, which is nice as the body would then not rotate instantly. The instant rotation however was brought back and fixed to the head rotation with some extra lerping added, as that would fix the yaw rotation of the head. The pitch and roll still gave issues and were ultimately fixed by ditching the roll entirely and only using the pitch. The result is relatively reliable and works in multiplayer scenarios.

Research – Freezing Skeletons

One of the most worked on prototype(s) during IMT&S for me are the bow and arrows. Each arrow had different effects, some of which affecting enemies and some only the environment. One arrow, the ice arrow, had a quick first prototype where the enemy skeletal mesh was frozen in place and would require to fall over. This however, resulted in a mixed result as while it did *kind of* work, it still had many issues. The main issue was the collisions, as once the skeletal mesh was frozen by freezing

the joints, the collisions were extremely inaccurate and would result in the mesh clipping through the ground at all times.

Research was conducted to figure out a way of making skeletal meshes simulate (thus apply gravity), solve collisions and still be frozen in place. This resulted into many failed attempts, and was eventually tossed.

I even did some research into active ragdolls at the time, but due to the complexity it was put on hold and ended up never being continued.

Finally, we came to a different solution where the enemy would instead be frozen in an ice cube. This worked fine and was later expanded to also include already simulated (ragdolls) to be frozen into ice. This was achieved by using pose snapshots, which allow the current state to be frozen while still animating. This means the skeletal mesh does not need simulate turned on, and only requires small additions to the animation blueprint. It still did not have what we initially wanted, but turned out to be a nice addition as it led to the puzzle where you need to build a bridge with dead bodies.

Research – Widget Input

Widget input is the UI input. In VR all widget input is in 3D space, meaning they are handled differently from normal UI. In multiplayer sessions, there is a weird bug that stops you from inputting any commands to the UI. This problem is one of the smaller ones that is covered in this blog, but is still worth mentioning the solution. After referencing many forum posts some talk about the reason why it is not working. Apparently, all UI is handled in a separate thread in the editor. This means that any fake input such as the VR input is sent to the thread and handled there. When running two sessions on one machine, they fight for the thread and start triggering the events every update tick. [23]

This is, for the editor, not fixable from what I have found. This issue does luckily not exist in builds but is mainly an issue in testing with the editor. You could force it to run in separate threads, but this takes more time to run each test.

Research – Altering Terrain

One of the problems encountered in the project was changing the collision settings of terrains in real time. Back in November, glue arrows worked by shortly changing the collision response and allowing particles to be hit, and then changing it back to ignore it. This allowed glue particles to only hit the object that was hit with the arrow, and not spread onto objects behind it.

Terrains caused a problem as they could not be changed in real time through blueprints. This served as a nice challenge, so I took up the problem and tried to solve it with C++. I attempted to find the primitive components of the landscape that make up the collisions and force change it. This was however blocked in real time. After debugging and sitting with our Unreal teacher; Daniel we concluded that this approach was not possible. We instead opted for changing it from the point of view of the glue particles, where the glue would look for terrain instead of the glue channel if a terrain was hit. This solution worked perfectly and solved the problem.

Research – Foliage Fixes

After the terrain issue was done I took another research job upon me; debugging the build. There was a strange crash each time we tried to package the project, which resulted in us unable to make a build which was required for the playtest/demo deadline we had set for the week after. While solving the issue I figured out new debugging ways, such as delving into the logs and actually stepping through the editor. I ended up fixing the issue by referencing the error code and narrowing it down

to the foliage, and by a trial-and-error basis of removing the foliage one by one. The issue was that one of the foliage type was a blueprint which combined the trunk and leaves. This caused issues, and thus had to be combined into a single mesh. After this the problem was solved.

This was again a small research topic, but felt worth mentioning due to the skills it gave me in debugging and it was still an important fix.

Research – Falling physics objects

This is another failed research topic which caused many headaches. The problem consists of objects that fail to apply correct angular velocity upon being placed on one of the edges. The gravity would fail to work, and would result in the object simply falling down very slowly until it finally hits the ground. Different forum posts with the same issue reported many different values to tweak, none of which resulted in the objects falling down normally. I figure it's something with how sleeping is handled in the physics, but nothing seemed to improve the behavior of the objects.

I sat down with Daniel and he also had no clue what could cause it, and the problem was easily replicated in different projects. Although much research was done into the topic, I (/we) were unable to solve it. I still feel it's important mentioning failures, especially as this one took quite a lot of research.

Research – Edge bouncing

During the mid stages of development while I was working on the ice cubes we stumbled upon an annoying bug during a playtest. This resulted into the problem of Edge Bouncing with Ice Cubes to give it a term. The problem was that whenever players wanted to walk upon the ice cube with the enemy inside, a core element that's crucial for one of the puzzles, they would bounce away from the edges. It was possible to land on the center of the cube and walk without any issues, but the corners had some weird properties that caused bouncing.

This first led me to experiment with different solutions, mainly in the physics material settings. It has several tweaks for bounciness and the like, and seemed like the obvious culprit. After attempting to change each value however, this did not seem like the problem.

I then went back to browsing forums trying to find solutions; desk research. This gave me *some* ideas on what to try, but ultimately did not yield many solutions. I tried again the following day and stumbled upon more forums which did not state my issue but did lead me into the direction of the solution; character can step upon. Back then, the ice cube was still a child of the enemy meaning that the hierarchy was "enemy AI > ice cube > mesh". There was no issue with the ice cube and the mesh, but the enemy ai (parent) of the ice cube was set to player can't step upon. Even though the ice cube was set correctly, this still did not overwrite the parent and some "fighting" seemed to happen mainly on the edges. This caused the player to be able to walk fine on the top but bounce off the edges.

Research – Moving players on moving objects

The first thing the player encounters in our game is the spaceship. It's a large, fully decorated ship which moves to the planet in which the level is loaded asynchronously during a part of the landing. Moving large objects (while initially fully disputed by everyone) ended up in the game anyways due to design choices and brought some major issues with it. The main issue is lag. Moving the ship is done with an animation sequence. More methods have been tested, as I will state in a later chapter too, but none of them seemed to fix the amount of lag the landing sequence causes. This lag creates a larger problem, as players need to stay on the spaceship.

How players initially stayed on the spaceship was through the Movement Component it possesses. It already has functionality that allows it to move with moving objects, keeping their momentum, and calculating how to move relatively to it. The problem is created by the low framerate. The spaceship moves fast, and if the player controller misses a frame or two of keeping up with the object, the player would shoot away from the spaceship leading them stranded in space (or rather falling infinitely). With framerates as low as 20-40 FPS this would happen consistently enough for it to be a major problem.

Much research in both desk and experimental research has been done to find better solutions to the issue. There were three possibilities: fixing the framerate, rewriting the player movement to be separate from the player controller to have more flexibility or attaching the player to the ship and stop them from moving.

The first solution, again as will be continued in profiling, was unsuccessful. Even moving the ship without any extra things brought the fps down to about 40, which was still unreliable in its movement. We did attempt it for a bit, even in later stages, but it would still throw away the player 1/4 times and in multiplayer 9/10 times. Moving the ship was just a bad design choice, but we were too deep to change it.

Rewriting the player movement seemed like the other popular option online. This would have worked but would have taken a lot of time. Due to the amount of work I still had to do, we prioritized that over fixing this problem.

This leads to the third solution, one I tested out early and came back to very often. Whenever the ship would start the landing sequence, it would first retrieve all players and attach them to the spaceship. This introduced issues with the movement controller, as it both tried to keep up with the object the player is standing on and move with the object, causing the velocity of the object to be added to the player itself. After doing extensive research online there was no way of making the movement controller calculate its velocity required locally, thus the only solution was to disable movement entirely. This was only one large downside, the other was that rotation was now no longer taken into account. With movement disabled, rotation would no longer be transferred onto the player, and with VR added the player would always look in the same direction while the ship moves down. Sure, you can look around and physically rotate with the ship, but the rotation of the ship is not seen on the player. This breaks immersion immensely.

Over time I gave multiple attempts to finding a solution to the problem. After fixing some framerate issues I even turned off the sticking to the spaceship, but it was too unreliable in VR so we placed it back. The end result had the first part where not much visually is happening to still allow the player to move (but not jump), and the deny movement for the second part after the level is loaded (and there is a lot of visual content).

Research – Controller velocity

Melee was one of the features we initially were really excited about putting in. Before starting this project, we worked on a different project called Apollo Arena. We really wanted to have arrow melee in that game, but could not implement it due to time constraints. This time we had all the time required, and I implemented several ways of performing melee combat. One of the ways was with the use of arrows, and as arrows did not have many collision aspects themselves, we were required to retrieve the controller velocity.

This turned out to be a problem however as the documentation for the Grip Motion Controllers was very vague and did not talk about retrieving the velocity of the controller. There are specific functions for retrieving the velocity, but they don't seem to work at all.

This led to a small desk research into how to fix it, with many forums either also having no clue or managing to fix it without telling how. I finally discovered a reddit post [60] which gives an easy solution which is to grab the velocity of a sphere component (which is already handily attached in the plugin by default) which is the same as the controller velocity. This worked perfectly and allowed me to implement melee by taking the linear and angular velocity into account when calculating the damage.

Research – Light baking onto movable objects

Baking light is usually done in static objects. It helps increase the FPS by pre-calculating all light and adding light probes to do any fake dynamic lighting. This is very useful as it allows the game to run with barely any light calculations, which usually take up a large portion of the games' frame time. Sometimes, this is also done on movable objects that don't need dynamically moving lighting. The spaceship, being as large as it is, simply can't handle dynamic lighting as it drops the frame time by too much, thus it had to be baked. It's all movable however, so for this we needed a solution.

And initially that solution was not found by me, it was Teun that found a simple setting which causes light to be baked onto movable objects. It worked fine, but when we actually got further into the project and started making daily builds, we discovered that the light actually was set back to dynamic as soon as you launched the build.

This of course was a major issue and brought the FPS down to as low as 10 FPS on VR. This is unacceptable and had to be fixed as soon as possible. I started researching and experimenting with different ways of how to fix it.

Many forums had posted similar issues, but only some provided possible ways to fix it. Sadly, each attempt required a full light re-built and a re-package so it took about 20-30 minutes per try. After trying out all the different solutions, such as for example setting "Lightmap Directionality" to false for all moving objects [63] or trying to remove the cache [62] or trying out all sorts of different settings on the lighting I tried looking for other ways to increase the framerate.

One of the things that worked was removing the number of light sources. By reducing it to a maximum of 2 per room, so about 10 in total, we were able to run the game back to ~40-60 fps (which would go down again during landing). It seemed like a last resort option, but it was an option.

I continued to experiment with different solutions, and finally found something that worked. One of the artists who did all the lighting placed all the lighting in the same blueprint as the models. In unreal, objects can either be movable or static (or stationary, but that's not important here). For light to be baked it needs to be static, for objects to move they need to be movable. Movable objects *can* be a child of static objects, but not the other way around (movable can't have a static child). By having both in the same blueprint, the blueprint was confused about if it should be static or movable. All the models in the blueprint were put on movable, but the lights (and thus the parents of the lights until the top) would be static as static can't be a child of movable. This weird combination allowed it to kind of function in the editor, but as all objects would instantly turn movable as soon as you packaged it (as they were designed to move), the lighting would be lost.

Upon separating the light and the objects baking worked as expected, and FPS improved dramatically.

Research – Profiling

The final research point for me was on how to properly profile Unreal. Over the course of the project I did multiple adventures into the topic, and at the end I really sat down to test several different things to try and improve the performance.

I had already used the Profiling Insights tool before, but I had read some instructions and watched some tutorials near the start of the project to really understand what it does and how it works. I still don't fully grasp everything it offers, as it's very in depth, but it allows you to narrow down what calculations take the most time by going down the call chain.

Furthermore, I found some very interesting commands [80] that help track down what takes the most time while still in the editor without having to rely on other tools. The first most useful one is "ProfileGPU" which takes a snapshot and shows all the timings of the GPU. The second is "stat DumpFrame -ms=0.1" which dumps everything that was run during the tick in the console or command prompt. This one was very useful for debugging builds. The next nice commands were "stat Game" and "stat SceneRendering" which give information on things such as light information, how much ticks took and other aspects such as how much physics is taking.

Finally, I also was able to do some profiling for things such as Occlusion Culling [81]. It initially seemed like there was none happening, but upon closer examination with the commands "stat initviews" and "r.VisualizeOccludedPrimitives 1" it was clear that it was working as expected.

Profiling at the end allowed me to cut down quite a bit of lag by optimizing settings in models, mainly at the spaceship. All models still had their overlap events turned on, which resulted in about 8-10ms of calculations each frame. Cutting this down brought it back to ~2ms which was a massive improvement. Turning off all the "affects navigation" on objects that did not require it also helped a lot, cutting down a further 3-5ms. Overall it's still laggy due to the spaceship moving, for which I tested out multiple ways of movement while profiling (for example moving it with timelines, or using the interpmovement controller or using tick) but all resulted in the same amount of lag. Ultimately it only gave some improvements for VR, but it was not enough to make it a great experience in terms of framerates.

Results & Reflection

Overall, I am content with what I have covered over the time of IMT&S. Some real insights were gained into several different aspects of unreal which will ultimately really help in me using the platform in the future. Although a lot of smaller aspects weren't covered in this research blog, it still demonstrates the amount of research simple or large issues can have for a programmer. I think ultimately most problems can be solved by applying the "experimentation" research method which is hard to demonstrate in text. I am happy with all the progress made, and the different research methods applied. I do think playtesting could have been a larger aspect of the project, mainly with strangers as they give extremely valuable information, but that is something to take into the next year (as I want to make user testing a large part of my graduation).

Not all issues in this research document have been solved, but that is fine. Not all times research leads to a valid solution, even when trying for a long time, and it is important to recognize those situations as important as well. I am happy with the end result, and a lot of the research during this project led up to it (as initially I did not have a lot of experience with making actual games in Unreal).

Sources

All sources are categorized under their respective problem. They are not APA sources, as they are all very informal sites such as forums, youtube videos, github or documentation. Each source is numbered to allow in text referencing, although most were too insignificant to the end solution to be referenced and are thus not referenced in text.

IK Rig

[1] https://forums.unrealengine.com/t/full-ik-body-in-vr-with-unreal-engine-5-0-3/648431

[2] https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/SkeletalMeshAnimation/NodeReference/SkeletalControls/TwoBoneIK/

[3] https://github.com/JonasMolgaard/UBIKSolver

[4] https://www.youtube.com/watch?v=EKR8ogonD68

[5] https://www.unrealengine.com/marketplace/en-US/product/power-ik

[6] https://www.youtube.com/watch?v=6Xiq6w9mJaI

[7] https://www.youtube.com/watch?v=qBooEZnlAA4

[8] https://www.unrealengine.com/marketplace/en-US/product/full-body-ik-plugin-for-vr?sessionInvalidated=true

[9] https://docs.unrealengine.com/5.0/en-US/control-rig-full-body-ik-in-unreal-engine/

[10] Finding existing implementations on discord in: https://discord.gg/7mRnH9Vx

[11] https://github.com/kvoeten/UnrealBodyPlugin

[12] https://forums.unrealengine.com/t/how-to-use-lerp-rotator-in-c/547921/3

Freezing skeletons

[13] https://forums.unrealengine.com/t/possible-to-freeze-joints-on-skeletal-mesh-while-simulating-ragdoll-physics/82954

[14] https://forums.unrealengine.com/t/freeze-certain-bones-in-animation/83817/3

[15] https://www.youtube.com/watch?v=3O1o_-VP2qk

[16] https://www.youtube.com/watch?v=CMOJrnvROS4

[17] https://forums.unrealengine.com/t/how-to-have-skeletal-mesh-be-affected-by-gravity-but-not-ragdoll/156463/2

[18] https://www.youtube.com/watch?v=1OcGAGT2opU

[19] https://www.youtube.com/watch?v=b9EmFnklpLk

[20] https://youtu.be/3O1o_-VP2qk?si=9DGxUIXHvGGwxnNL

[21] https://www.youtube.com/watch?v=0H6w3YtLr2Y

[22] https://www.youtube.com/watch?v=nkj6PAbGYtM

Widget Input

[23] https://forums.unrealengine.com/t/3d-widget-interaction-not-working-in-multiplayer/1246334

[24] https://forums.unrealengine.com/t/bug-widget-interaction-in-multiplayer/402548/5

[25]https://www.reddit.com/r/unrealengine/comments/dvc4im/multiplayer_when_a_player_populates_a_widget_with/

[26]https://www.reddit.com/r/unrealengine/comments/cxo621/3d_widget_problems_with_interaction/

[27] https://forums.unrealengine.com/t/3d-widget-interaction-button-highlights-but-cant-click/382803

[28] https://forums.unrealengine.com/t/widget-interaction-multiplayer-doesnt-click/446000

[29] https://forums.unrealengine.com/t/weird-widget-interaction-un-hovering-behavior/84203

[30]https://www.reddit.com/r/unrealengine/comments/ydbr2q/when_i_hover_with_widget_interactor_it_both/

Altering Terrain

[31] https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/Landscape/Collision/

[32] https://forums.unrealengine.com/t/how-to-change-collision-of-my-landscape-with-blueprint-4-18/417943

[33] https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/Collision/SetCollisionResponsetoChannel/

[34]https://www.reddit.com/r/unrealengine/comments/vhxs5b/change_collision_response_to_channeldisable/

[35] https://docs.unrealengine.com/5.3/en-US/collision-response-reference-in-unreal-engine/

Foliage Fixes

[36] https://forums.unrealengine.com/t/ue5-2-crash-whenever-i-place-foliage-in-certain-places/1203499

[37] https://forums.unrealengine.com/t/ue-5-1-is-actor-foliage-depricated/722546

[38]https://www.reddit.com/r/unrealengine/comments/zrhhdq/performance_merging_actors_vs_making_them_foliage/

[39] https://forums.unrealengine.com/t/assertion-failed-error-with-4-27/792355

[40] https://forums.unrealengine.com/t/4-10-4-cant-load-project-anymore-unknown-exception-code-00000001-first-second-chance-not-available/348657

Falling physics objects

[41] https://forums.unrealengine.com/t/stuff-is-falling-way-too-slow/77274

[42]https://www.reddit.com/r/unrealengine/comments/63ojk6/destructible_component_falling_too_slowly/

[43] https://forums.unrealengine.com/t/angular-velocity-is-being-dampened-despite-0-0-angular-damping/259217/2

[44] https://forums.unrealengine.com/t/bug-max-angular-velocity-doesnt-work/408972

[45] https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/FrictionRestitutionAndDamping/

Edge bouncing

[46] https://forums.unrealengine.com/t/how-to-stop-character-bounce-off-when-it-jumps-and-hits-an-object/1190242/2

[47]https://www.reddit.com/r/gamedev/comments/ubvxl4/i_was_wondering_how_i_can_make_my_character_stop/

[48] https://forums.unrealengine.com/t/movementcomponent-pawn-capsule-collision-bounces-randomly-when-sliding-off-a-90-degree-ledge/90421/4

[49] https://www.reddit.com/r/unrealengine/comments/11ub6rf/how_to_prevent_collisions_from_pushing_each_other/

[50] https://forums.unrealengine.com/t/character-is-bouncing-when-moving-on-inclines/245490

Moving players on moving objects

[51] https://forums.unrealengine.com/t/how-to-stop-moving-platforms-affecting-jump-momentum/85727

[52]https://www.reddit.com/r/unrealengine/comments/13buxjg/the_character_falls_off_the_moving_platform/

[53] https://forums.unrealengine.com/t/character-falling-off-fast-moving-platform/401621/4

[54] https://forums.unrealengine.com/t/character-falling-through-floor-and-ground/403395/2

[55]https://www.reddit.com/r/unrealengine/comments/pabjjk/hi_my_player_keeps_shaking_while_standing_on_a/

Controller velocity

[56] https://forums.unrealengine.com/t/how-to-i-get-motion-controllers-velocity/97140/2

[57] https://communityforums.atmeta.com/t5/Unreal-VR-Development/How-can-I-get-the-velocity-speed-of-the-Motion-Controllers-in/td-p/1060721

[58] https://blueprintue.com/blueprint/7vo667e9/

[59]https://www.reddit.com/r/unrealengine/comments/132702o/is_there_a_way_of_getting_motion_controller/

[60]https://www.reddit.com/r/unrealengine/comments/5rykxo/i_could_never_get_velocity_from_motion/

Light baking onto movable objects

[61] https://forums.unrealengine.com/t/baked-lighting-on-moving-object/304301/2

[62] https://forums.unrealengine.com/t/lighting-build-keep-saying-unbuilt-objects-after-build/41996/9

[63] https://forums.unrealengine.com/t/problem-with-the-rotation-of-the-mesh-on-which-the-light-is-baked/242150

[64] https://forums.unrealengine.com/t/how-can-i-see-all-the-light-sources-that-affect-selected-actor-mesh/108174

[65] https://forums.unrealengine.com/t/listing-all-objects-affected-by-direct-lighting/316045

[66] https://forums.unrealengine.com/t/baked-light-lost-on-force-surface-mesh-after-rotation/127822/3

[67]https://www.reddit.com/r/unrealengine/comments/6c5mr4/help_optimization_issue_shadowdepth_atlas_taking/

[68]https://www.reddit.com/r/unrealengine/comments/ffox39/not_sure_how_to_fix_this_lighting_problem/

[69]https://www.reddit.com/r/unrealengine/comments/s51pb5/is_any_way_to_have_a_static_light_that_effects/

[70] https://forums.unrealengine.com/t/how-to-build-lights-on-movable-objects/670779

[71]https://www.reddit.com/r/unrealengine/comments/go5740/lightmaps_for_movable_objects_in_the_unreal_engine/

[72] https://youtu.be/qytono6ft8w?si=aDRWWQz_hF36U6IS

[73] https://youtu.be/8I0IfhBjMIw?si=ztegFDSa072lSQdB

[74] https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/Rendering/Components/Light/SetForceCachedShadowsforMovableP-/

[75] https://forums.unrealengine.com/t/movable-objects-bake-light-problem/135611

Profiling

[76] https://youtu.be/GuIav71867E?si=US5SFcJr2ijM8_1v

[77] https://www.youtube.com/watch?v=HOcXWZY044E

[78] https://www.youtube.com/watch?v=H9Yb8Y2-Kng&list=PLF8ktr3i-U4A7vuQ6TXPr3f-bhmy6xM3S

[79] https://docs.unrealengine.com/4.26/en-US/TestingAndOptimization/PerformanceAndProfiling/GPU/

[80] https://docs.unrealengine.com/4.26/en-US/TestingAndOptimization/PerformanceAndProfiling/CPU/

[81] https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/VisibilityCulling/

[82] https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LightingAndShadows/DistanceFieldAmbientOcclusion/

[83] https://docs.unrealengine.com/4.27/en-US/TestingAndOptimization/PerformanceAndProfiling/Profiler/