13-01-18: First thoughts

The five-part reading on lighting theory in video games by Robert Yang pushed for me to think of light formally which I realize I had never really done before. He bypasses the predominant use of lighting in video games to create a mood defined as *tactical violence* and explores it from a theoretical viewpoint. He describes how different forms of light fixtures go beyond their physicality and create different messages influenced by their context and usage. Yang posits that those evolve through time as technology progresses, his best example being the Edison light bulb, which was at first quite an experience, but quickly became outdated and now is more of a luxury item for collectors.

This gave me strong Marshal McLuhan vibes, with the idea of light being as much its medium as its message, i.e. all the properties enumerated by Yang and the connotations they create. Seeing one part dedicated to three-point lighting – which I have experienced in video production – was interesting to put in a video game setting. The light placement really is dependent on the viewpoint and the video game environment, which is meant to be more of an explorable space than a set of scenes. Since I am more interested in exploring lighting than camera right now, I am wondering how to take this idea of multiple lights setting while also finding a way to make it more aware of it being part of a video game, an explorable setting.

20-01-18: Narrative & ludic, function &. mood, passive &…?

Skimming through some more of the Radiator blog posts, I was reminded of a concept seen in Intro to Game Design about the ludology vs. narratology debate. It feels worthwhile for my research to explore it with lighting theory thrown into the mix. The “debate” pits two opposing viewpoint on video game theory, with games coasting along a spectrum between focusing mainly on the gameplay or the plotline of a game. Some will be on the far end of ludology (Tetris), others on the far end of narratology (most If not all Japanese visual novels) and most games in-between. With lighting in mind, it can be said that this feature can and has heavily been used in both sides: like shown in Magnar Jenssen’s "Functional Lighting", a lot of games use it in a ludological(?) context, giving directions and hints to help players read the map and move along efficiently, especially for face-paced games; and, in narratological context, light is also heavily used to create a narrative context and mood.

But one thing seems to be like it is barely considered with the lighting scheme, and that’s the input of the player on it. Though many games allow us to superficially affect it (open or close light, day and night cycle through playing, picking up a light source), there is never a full control of it as might be seen in a different medium. This is interesting considering the fact that lighting in video games is much easier to achieve and play with than in the physical world. As the ludology vs. narratology debate is, to me, chiefly about the interactive nature of video games and the contrast between passive and active modes, I want to have light an exclusively active role, as its role feels to be exclusively passive due to it being designed usually by the developers. By active here I mean that is affected directly by the viewer -- as I realize some games such as Limbo have light at the forefront of their game design sheet but without relinquishing its control over.

Thinking back on the three-point lighting technique, I like the idea of exploring other art forms’ lighting concept within this Tanks game, and putting the focus away from its *tactical violence* genre and more on the medium of the video game as a possible theatrical scene for war propaganda. The idea would be to have several spotlights that are influenced by the input registered by the players, allowing them to change the light scheme depending on their current action.

20-01-20: Implementation

Now that I have an idea of where to go, I need to start thinking about the way in which I’ll implement it. I basically need to have a way of recording all the inputs from the players (probably in an array) and recording in another Boolean array which are active and which are not. This array can then be used to tell an array of lights which should be lit and which shouldn’t. Naturally, I divided those tasks in two scripts to increase readability.

InputSorter: This class would need to know what are the player inputs. Since the tanks are only created on launch, this needs to be set after compilation. Since the tanks have a manager, this is from where those movement should be taken. An of TankMovement is what I ultimately need. Once that is loaded, I can created a corresponding array with as much Booleans that are initially all off. Then its main method should be about turning it to true whenever a TankMovement is registered.

LightManager: This class would contain links to all the lights and tell them when to be on and when to be off from the InputSorter’s results. It should also have a link to the Sun to be able to also affect it.

The GameManager will ultimately be updated also to call those previous classes at the right moment.

*ATTEMPT 1*: It took me a while to get the TankInputs array to be successfully created because I had originally put the InputSorter script on a different object and the compilation got mixed up between the GameManager object and the GameManager script. I ended up putting both InputSorter and LightManager on the GameManager instead of creating a new object. This way, I can GetComponent from the gameObject easily. From there it went smoothly and, after fixing a few errors the game managed to launch. The public Boolean array allowed me to see easily the input triggers. Only an else condition was missing to ensure the Booleans turned off when input stopped, so I had to fix this error in CheckInput.

20-01-22: Setbacks

*ATTEMPT 2:* I now try to implement the code for the LightManager. It runs but with some problems, as the lighting keeps flickering on and off. I try to implement a code that would make the light be affected only once per cycle of input (treating the button being pressed and held as a cycle and being held and released as another). Unfortunately, I seem to have created a problematic code as Unity becomes alarmingly processing-draining until my computer crashes and I have to force close it. When I get back to my project, the LightManager script is filled with nulls and the project, broken, can’t be run anymore. I can unfortunately not put more time into this today so I will have to get back to rewriting LightManager tomorrow.

20-01-23: 2.0

*ATTEMPT 3:* I ended up having to redo the ScriptManager. I used this occasion to redo the implementation in GameManager more rightly, as I wasn’t originally able to properly switch from the spotlights mode (with the input) to the normal mode (without input, turning spotlights off and turning sun on). This time, instead of accessing the attributes of any method from the exterior, I implemented more small methods to do all the necessary actions from within and without, giving them the task of changing attributes; I feel this second attempt was a better step towards efficientf OOP. I also added in the InputSorter another Boolean array so the previous sorted input could be compared to the new one and react accordingly (this is to simplify later coding); as well as broke down some of its CheckInput methods into other methods to be able to check whether there was input or not (and avoid unnecessary computation), whether it changes or what to do when it just did change. The game now works as intended, though it took some time to get the Sun light (directional light in level art) to respond properly. Managed to make sure all lights are only turned on once per cycle.

**What’s next?** Now that it works, I want to really work on the composition of the project as it still very far from what I had in mind visually. I also want to push further the implementation to combine player inputs together and create lights schemes impossible alone.

20-01-25: Experimentation with aesthetics

While working on the coding for the light scheme, an image started forming itself in my head about the look for this. It might be the winter getting to me but I really want to play with a completely white playground. It looks somewhat like marble, very pristine and “museum-esque”, in my mind’s eye. I thought that, combined with the idea of rendering only the shadows (thanks to Pippin sharing this experiment), they would give a singular look to this piece that would heighten the theatrical component I am focusing on with *Mis-en-scène.*

*ATTEMPT 1*: I struggled a bit at first before finding the option the render only the shadow, having started with changing the material to transparent before realizing it was a built-in option past v5. I then went through all the level art and saw the objects that I wanted to affect: ultimately, only the inorganic building and tanks got switched to only shadows rendered.

The rest got at first all covered in white material, with some element in the oil drums material to create slight contrast. Since white reflects light differently, adjustment had to be made to the color tone of the spotlights (previously bluish to avoid a overly yellow light). I also dimmed the “sun” light and switched from disabling it to merely reducing its intensity; otherwise, the shadows that now constituted the tanks and the building were lost with or without spotlights on, when the spotlight was off.

In the end, the white material, for the plane especially, felt a bit too much like snow. Though it didn’t look bad, what I had had in mind had been more reflective, almost plastic-y. I found two marble textures, applied them to materials and used them throughout the scene. I am happier with the result for now but would still like to make further tests with it, especially with refraction.

20-01-29: Light scheme switcharoo

I am feeling a bit stuck with project at the moment, being happy with the InputSorter and LightManager scripts but somewhat disappointed with the look of the project. Despite having played with the aesthetics and having them closer to what I had in mind, I still find it is lacking a little bit. The marble texture, which I initially like, now feels somewhat tacky I think the current lighting scheme/setup is boring, but being unsure how to fix it I decide to get some inspiration from a little inquiry on lighting setup for the stage. Through this research, I have found some that were closer to what I had in mind and I set-out to try and replicate those through my project.

My first aim was to get one light to be able to send out multiple instance of itself in all directions, kinda like a disco light. I did some research on that but had a hard time finding any Unity tutorial for that; more often then not, my search returned unrelated things and it was difficult to get anything that was linked to what I was looking for. I attempted to use the water and glass refraction materials to try and get this effect but both failed. In the end, I decided to let go of this idea and focus on making the light scheme more visually interesting.

Having placed the light one after the other in the code with the input (e.g. Light A impacted by Tank 1’s forward movement, Light B impacted by Tank 1’s backward movement…) I played around with this a bit and exchanged light names and played with their intensity, rotation, position, range and spot angle. With my final setup of the day, I am much more satisfied with the more varied light placement. I hope to make further experimentation with this and see how to strengthen it.

20-02-03: Assessment on Lighting experiment

Though I had originally planned to have a more complex lighting scheme, I am realizing that it might not necessarily make my project stronger, nor make my final prototype stronger. As I am realizing the implications of programmatic experimentation – namely, that programs will, through development, bug and crash often and regularly – I choose to leave it as is, in its current state which is the best version of the project so far. I hope to keep playing with it in the future, especially for the final version of this class where all the experimentation will be put together into one project. Despite not having the time yet to go further with this experiment, I will still lay down here my further plans for this project. To get closer to the idea I had in mind, some minor additions would have to be made.

The InputSorter would need to keep track of the amount of lights currently activated through an integer variable. This would allow the program to only start using the new lights when more than two inputs are being triggered – so that the new lights are only activated once the tanks use at least three types of inputs. Another necessary (boolean) function, to go with said variable, would need to check for when the variable becomes >= 2; the function would then return true from than onward, which would change the functioning of the LightManager.

One major adjustment for this script would be to update the CheckInput function, which turns to true or to false each boolean assigned to an input/light. Here, the idea would be to make the setup more complex -- I am still unsure whether this should fall into the LightManager or here, but the idea would be as following: The current 8 lights (A to H), which are in the 1st cluster, are each provoked by one particular input. Once more than three inputs are registered (meaning at least one of the tanks has two inputs), one light from the 1st cluster is turned off while one or two lights from the 2nd cluster are turned on. So, for example, if lights A, C and H are activated, light C ends up being turned off while lights J and K are turned on; but, if lights A, B and H are activated, it is light A that is deactivated and lights I and K that are activated. The point would be to create visually pleasing/interesting setup from the various combination so that each of them gives up a different result. To push it event further, I would possibly see a 3rd cluster, which is only activated once inputs are being kept long enough. These cluster would work differently in that the lights would each (or all at the same time) be activated once the minimum required amount of time is registered. This would also activate lights from cluster 1 and 2, meaning the scene would be overlit from all sides. This might end up problematic with lighting being a heavy component to build; I’d have to test it and see how it turns up.

The LightManager is ultimately what would require more heavy work (depending on whether the previous implementation would have the bulk of it in here or in InputSorter), with it necessitating more light objects while implementing a more complicated light scheme that would check the previously checked images (from the 1st cluster) to determine which of the lights from the 1st and 2nd cluster are being modified. These modifications would all add a new layer of interactivity to my game which in my opinion would make it more interesting to play for a while, making the light scheme less predictable and pushing the player longer in their attempts at mastering the functioning behind the spotlights. Until then, I am happy with my current program and will build it as is for submitting.