DIGCRE Documentation

«A Cozy Winter Night»

GIT Repo Link

Contents

1 Vision

AI is rapidly evolving and new options to use AI in creative ways is a very relevant interesting topic. Generating media like stories, images or videos is already done on a daily basis and a lot of users find joy in creating these assets for personal and professional uses alike. Personally, I want to try using 3D models to create a nice scene in Unity which is compared to the methods not as much explored.

2 Tool Evaluation

I tried two main options. Shap-E from OpenAI and Luma AI from Luma Labs.

2.1 Shap-E

Shap-E is developed by OpenAI and also uses the huggingface API to create 3D models based on prompts or images. By experimenting with Jupyter notebooks I achieved different results.

```
render_mode = 'stf' # you can change this to 'stf'
size = 256 # this is the size of the renders; higher values take longer

cameras = create_pan_cameras(size, device)
for i, latent in enumerate(latents):
    images = decode_latent_images(xm, latent, cameras, rendering_mode=r
    display(gif_widget(images))
```

Figure 1: A feather generated with Shap-E using prompts.

The feather looks decent, but this quality was only achievable by using prompts. The image-based generation seemed to require very specific image properties and background colours to work. Even with a lot of trial and error the results were not suitable for this project. After creating some prompts I imported them into Blender to inspect the models further.

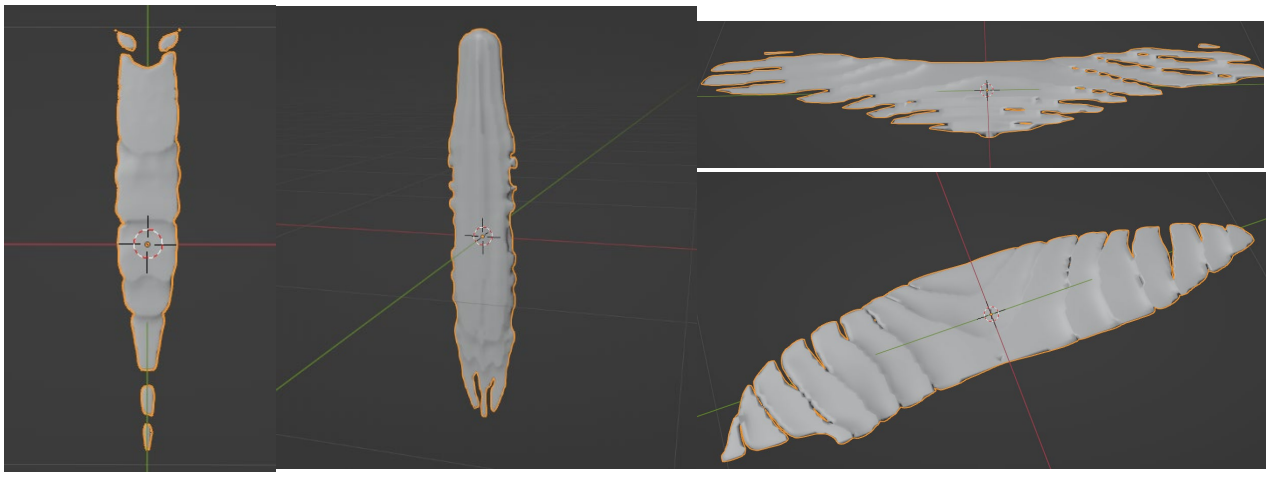


Figure 2: Additiontal feathers in Blender

A big problem with these models is the detachment particles from the base model. This is observable in the top right model right at the end of the model. This costs performance since the model generally will end up with more vertices and edges and it usually does not look very pleasing visually.

Figure 3: GIF files of the generated 3D models

Next, I tried a bird which also did not perform well. Some details on the wings are usable but they again have a lot of detached masses which is a big problem.

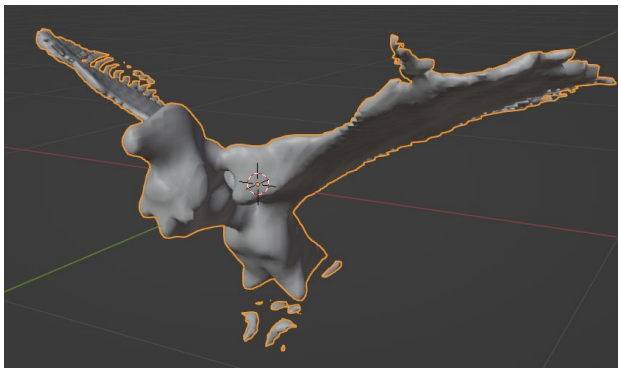
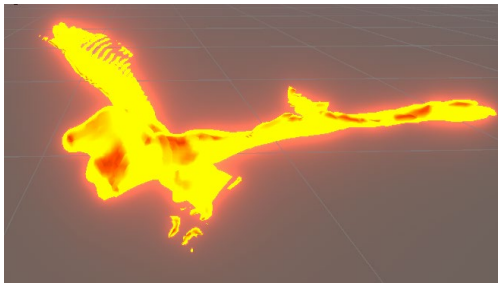


Figure 4:Bird model in Blender



I also tried to enhance the model using a shader and Unity postprocessing technology but even they were not able to save the bad texture and weird body parts.

2.2 Luma AI

Luma AI is a powerful tool which can process and refine prompt based queries. The results can then be exported into common 3D file formats and be displayed inside Unity. This system had much better results and was even able to create textures which made sense.

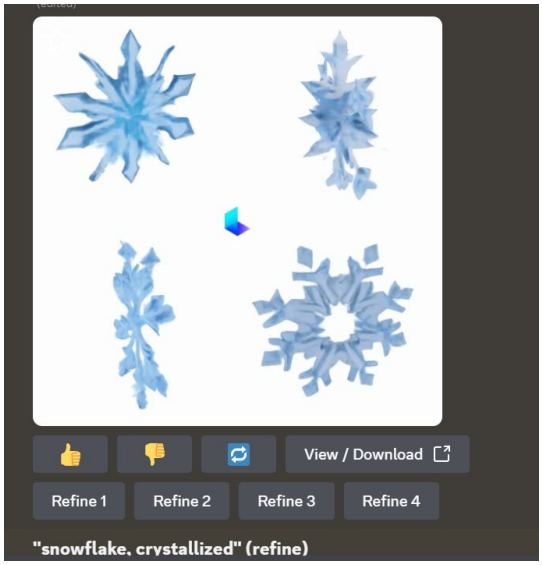


Figure 5: A Luma AI prompt result

These results can then be refined, which also happens through AI. This added more details to the textures and increased the resolution of the model.

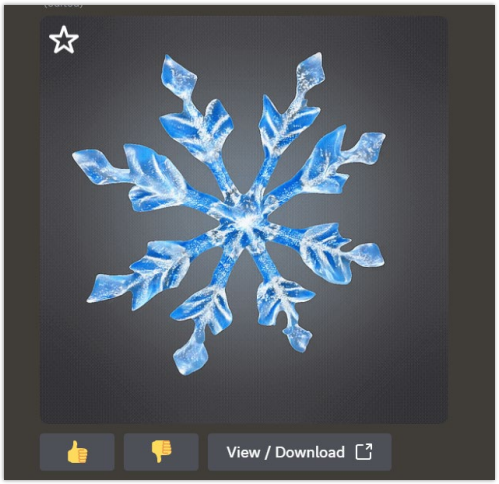


Figure 6: Refined Luma AI model

3 Unity Engineering

Based on this acquired knowledge from my research I was able to start building a prototype for my Unity scene. Fitting for the season, I planned to go with a winter themed landscape. After generating the models through Luma AI I did some scene modelling in Unity until the result was satisfying

Figure 7: Unity scene prototype

However, I still was not happy with the result. Even with some light sources and no global light (to imitate nighttime) it looked quite grim and not cozy at all. So, I did some manual Unity engineering with post processing. (not using AI) The goal was to make this a lot more cozy and less dark.

Figure 8: The scene after adding lighting

3.1 Shader programming

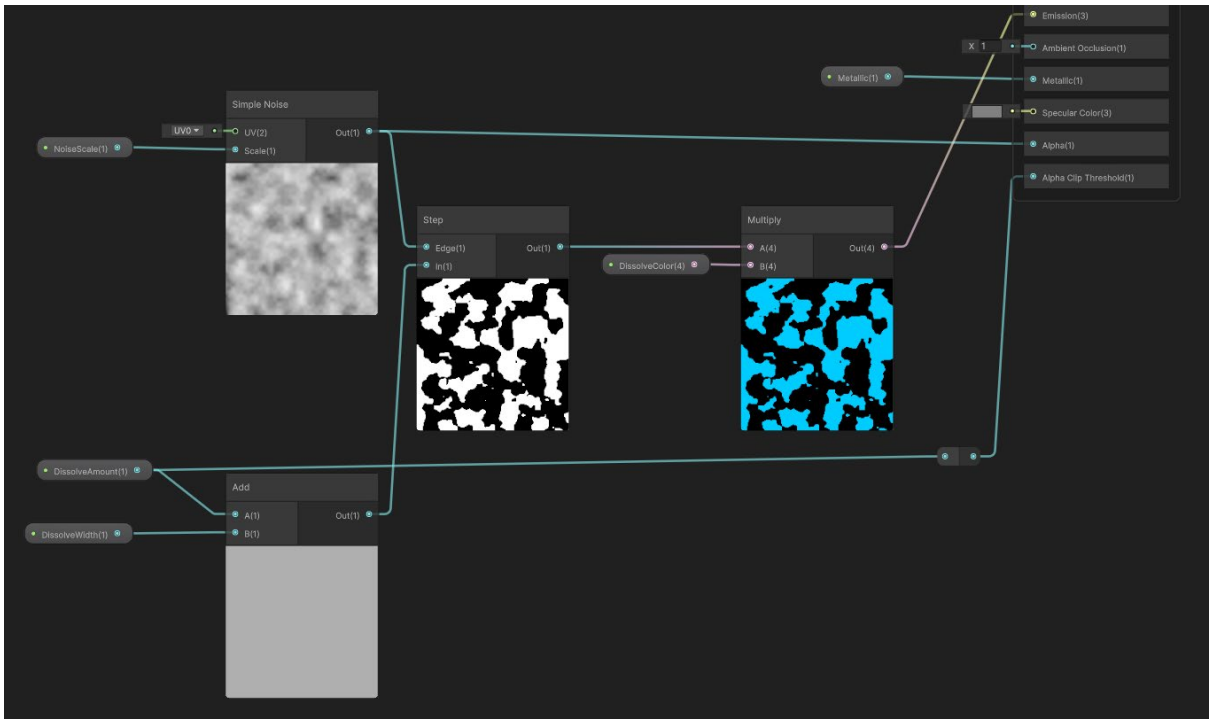


Figure 9:Shader Graph for Dissolve

Using Unity Shader Graph technology, I created a shader to try to animate the objects inside the scene. I will not go into detail here but generally we use a gradually increasing noise effect to make a 3D model dissolve or materialize based on the use case. In my case the models should appear out of nowhere and gradually take form, so we need a materializing effect.

3.2 Coding

```
vfx.SetFloat("duration", vfx.GetFloat("duration") / dissolveRate / 25);
if (reverseEffect)
{
    counter = 1;
    dissolveRate = dissolveRate < 0 ? dissolveRate : -dissolveRate;
}
else
{
    counter = 0;
    dissolveRate = dissolveRate > 0 ? dissolveRate : -dissolveRate;
}
```

Figure 10: Dissolve code snippet 1

The particles duration will be set here according to the value of the duration field. It may be confusing that strings are used here but it seems this is how Unity communicates with the exposed properties of shader graph and visual effect graph. The duration will also be adjusted according to the value of the DissolveRate property and a constant value. This ensures that the gradual disappearance of the model is synced with the duration of the displayed additional particles so that both visual effects look better when displayed simultaneously.

Additionally, depending on if the user wants to make an object appear or disappear the counter need to go from 0 to 1 or from 1 to 0. This Boolean property called “reverseEffect” will determine the according outcome which is relevant in the loop later.

```
while (materials[0].GetFloat("_DissolveAmount") <= 1 && materials[0].GetFloat("_DissolveAmount") >= 0)
{
    counter += dissolveRate;
    ChangeVfxColor(useDifferentVfxColor ? vfxColor : modelColor);
    for (int i = 0; i < materials.Length; i++)
    {
        ChangeModelColor(modelColor);
        materials[i].SetFloat("_NoiseScale", noiseScale);
        materials[i].SetFloat("_DissolveAmount", counter);
    }
    yield return new WaitForSeconds(refreshRate);
}
```

Figure 11: Dissolve code snippet 2

Here we will adjust the Dissolve amount between 0 and 1 depending on if the model should appear or disappear. To enable changing the color of the particles during the display of the effect there is a color changing method called inside the loop. A performance problem has not been observed during tests. The refresh rate will determine how fluently the effect will display. This can be increased to increase performance which gives the user additional control if they use this effect on many objects at the same time.

4 Result

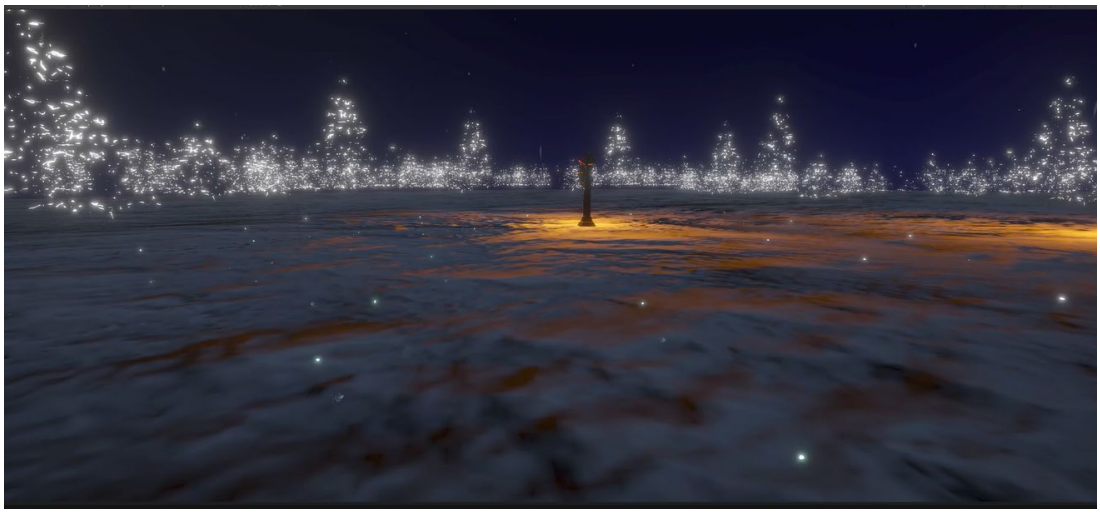


Figure 12: Animation starting frames

With the visual effects of my Bachelor thesis, we can add some animations to these models to make them appear more lively.

Figure 13: Animation in play

Figure 14: Finished animation

For full animations and more clips please check the attachments folder.

5 Conclusion and Reflection

I believe that I made an interesting personal observation during this project. Even though we are already able to develop quick results and powerful tools I believe that there is still an artistic difference or by extension, a possible clash of vision between AI and the human user. This is why I think manual engineering which I did with Unity in this case will still be relevant in the short term and middle term future to really create something that fits our narrative of the designs that we make. While the AI was maybe trying to create something more photorealistic, I rather wanted to go with something aesthetically pleasing to the eye without giving realism too much attention. I was able to achieve this using the Unity tools.

6 Table of Figures