Blender Animation of Spacecraft Docking in Microgravity

COSC3000 Graphics Project

Ryan White
s4499039

9th of June, 2021

Contents

List of Figures

1 Introduction

Almost since the beginning of space travel, spacecraft have performed orbital rendezvous and docking in order to increase the living space of astronauts, and transfer materials and fuel. While this is routine for rocket scientists, for the general public this maneuver is shrouded in mystery. On the cusp of a space travel boom, it's beneficial for the population to understand how spacecraft work and what processes are involved in their use. A more educated populace would no doubt result in more a technologically savvy society, in turn further accelerating ideas and technology development.

For two spacecraft to dock, they must first rendezvous in orbit; that is, minimise the distance between them to a suitably low number. Then, once all relative velocity is cancelled out, orbital maneuvers can be performed to orient the spacecraft relative to each other and completely cancel out the distance until two docking ports interact and couple to each other - interconnecting the interior space between them (allowing astronauts to move between compartments) and locking the two crafts together to move as one.

Blender, the animation software used for the project, is famously user-friendly and was learnt specifically for the purpose of the project (Appendix 1). The software has dedicated modelling capability, with built-in animation features all while performing back-end lighting and transformation calculations automatically, some of which were explained in section 3.

2 Aims

The goal of the graphics project was to simulate a simplified (and extremely quickened) docking procedure between two spacecraft already rendezvoused; the first modelled after SpaceX's Dragon spacecraft (used extensively in low Earth orbit to ferry supplies to the ISS), and the second a conventional transfer-like spacecraft used to ferry Dragon to some other destination. Using Blender animation and modelling software, spacecraft rotation and translations would need to be performed ideally with the use of RCS (reaction control system) thrusters as methods of linear transformations. This would demonstrate key knowledge of multiple light sources, object transformations and animation. As for the animation aspect, camera zooming and transformations would be necessary to properly capture the extent of which the spacecraft move.

3 Methodology

For the majority of the graphics project, animation and modelling software Blender was used. Universe simulation software Space Engine was used to generate a skybox, the details of which are in section 3.3.1.

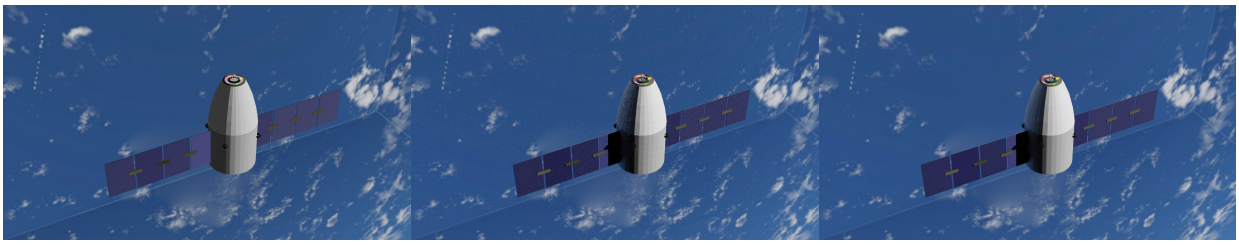
As a basic process, the models were each individually modelled, textures (where applicable) were then applied. The light sources were then created and bound to the corresponding spacecraft. A fire and smoke model was then created and tweaked to resemble rocket exhaust, and moved to the engine bell behind the transfer spacecraft. After this, 6 planes (each of side dimensions 400m) were assigned at 200m from the world origin (in each positive and negative XYZ direction) and assigned the associated skybox texture. Then, the 'sun' lightsource was situated in an appropriate position, and keyframes were created for each of the models rotations and translations. The animation was then rendered (using cycles lighting) overnight for approximately 5 hours.

Each subsection below details what choices were made, and the relevant information that informed said choices (with details of what was done).

3.1 Render Lighting

Blender, by default, has two main render lighting options: Eevee (rasterization via OpenGL) and Cycles (path tracing). Rasterization is a (relatively) inaccurate lighting method in which model faces are projected outwards to the camera, where shaders are then applied to render to coloured image. Path tracing works by shooting out particles from the camera (in a random area distribution), which then bounce off of (or transmit through) the faces of models in the scene. It most accurately models the way in which photons act in real life, without modelling each individual photon [2].

Figure 1 shows the visual difference between each method of lighting. Due to the path tracing shooting out particles randomly, some pixels never receive information about the scene regardless of how many samples are taken. Due to this, denoising shaders were created to fill in the gap of empty pixels.



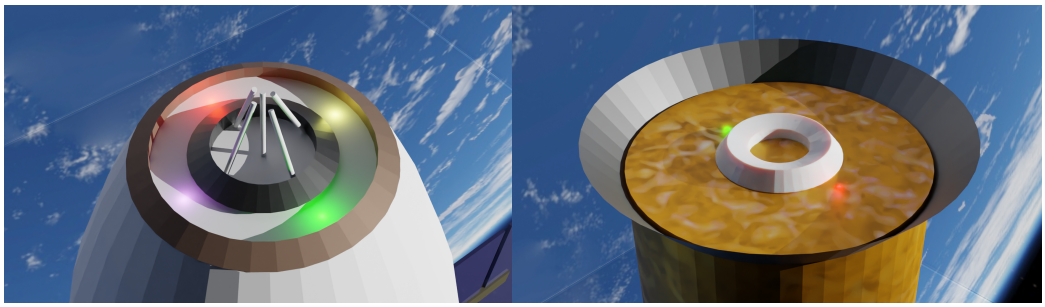
(a) Eevee (b) Cycles (Path Tracing) (c) Cycles with Denoise Shader

Figure 1: Comparison of Lighting Techniques

Since the project was to pre-render an animation (and not have real time interaction), cycles with denoiser was chosen as the lighting method for a more accurate look even with the increased time to render inherent with path tracing ($\approx$ 20s with cycles vs $<$ 5s with eevee). As Figure 1c shows, the denoise shader slightly distorted some edges of model faces. For the project, this was deemed a necessary evil as the positives of removing empty pixels outweighed the negatives of slight edge distortion. As an alternative, more samples could have been taken per frame (e.g. 100+ samples) but as it stood, 32 samples in cycles took 20 seconds per frame. While this is trivial for a single image, an animation of 820 frames takes much longer, and the render time becomes much more of an issue. This is exactly why denoising was chosen: to simulate an image with much more samples while reducing computation time.

3.2 Modelling

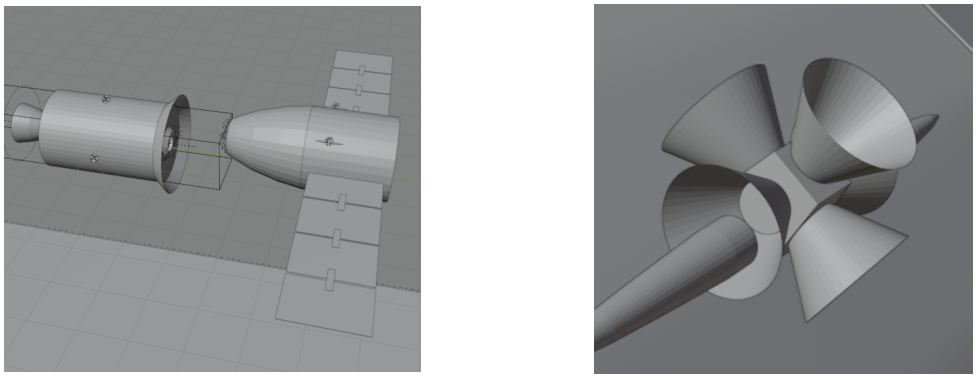
To show how spacecraft dock in orbit, models of the spacecraft themselves first needed to be created. In blender, this is done by creating basic shapes (cubes, spheres, cylinders, etc) and then manipulating the vertices to resemble an object. To create the Dragon spacecraft (right model in Fig 3a), two stacked cylinders were created. The top of the upper cylinder was tapered to a circle of much smaller radius than the base to somewhat resemble a docking port (Fig 2a). Then, multiple low-segment toruses were added on top to resemble locking mechanisms, with multiple light sources (see section 3.4) to imitate guiding lights. A similar process was done with the transfer spacecraft (Fig 2b - with an added engine bell at the bottom). A notable difference is that with the transfer craft, the docking port was modelled as a foam-like substance, and features subsurface scattering. That is, the surface is slightly translucent and scatters light that had entered so that it appears almost like foam.



(a) Dragon Docking Port (b) Transfer Docking Port

Figure 2: Docking Port Renders

In space, very small rocket engines are needed to rotate and translate spacecraft: reaction control system thrusters, or RCS thrusters for short. This consists of several engine bells orientated in 5 of the 6 possible directions (the sixth direction is towards the spacecraft on which it is attached). To rotate in one direction, only one of the RCS bells fires at one time which creates an unbalanced rotational force and orients the spacecraft. A typical model of an RCS thruster is seen in Fig 3b, with a firing RCS (in two directions) seen in Fig 6a. Modelling this consisted of one central cube, 5 hollow cylinders tapered at the inner edge, and two RCS 'exhausts' in the $\pm$ Z direction with high light emission (rendered only at certain points in the animation to simulate engine firing).



(a) Transfer [left] and Dragon [right] Spacecrafts (b) RCS Model

Figure 3: Greyscale Model Snapshots

3.3 Texturing

While a large amount of the objects modelled in this project were simply solid colours (with surface characteristics tweaked), two distinct 'objects' had defined textures; the skybox, and the transfer spacecraft.

Naturally, a skybox was chosen to have a predetermined texture for ease of computation and aesthetics. A plain black background could have worked, but using the Earth for a point of reference helped in showing the linear transformations of the spacecraft. A scale Earth could have been modelled, but would introduce (at the very least) thousands of vertices that would need to be calculated on each frame. As such, 6 planes (totalling 24 vertices) were created and textured and the result was viable.

The transfer spacecraft was chosen to be textured in order to show the capabilities of Blender and the extent to which I'd learnt it. Once the procedural texture was applied, the metallicity of the spacecraft surface was set to full to contribute to the metallic appearance. This property affects primarily the intensity of light reflected off of the surface.

3.3.1 Skybox

Using universe simulation software Space Engine (as previously mentioned), a skybox cubemap was generated. This tool comes by default in the application, and is available to use for any personal or education purposes free of charge or credit [1]. Figure 4 shows all 6 faces of the skybox cube generated, composited into one image. A fully lit Earth (at distance 408km) was chosen as the background as that would allow the animation to showcase the lighting and different angles to the fullest. A distance of 408km was chosen as that is a typical orbit altitude for the International Space Station.

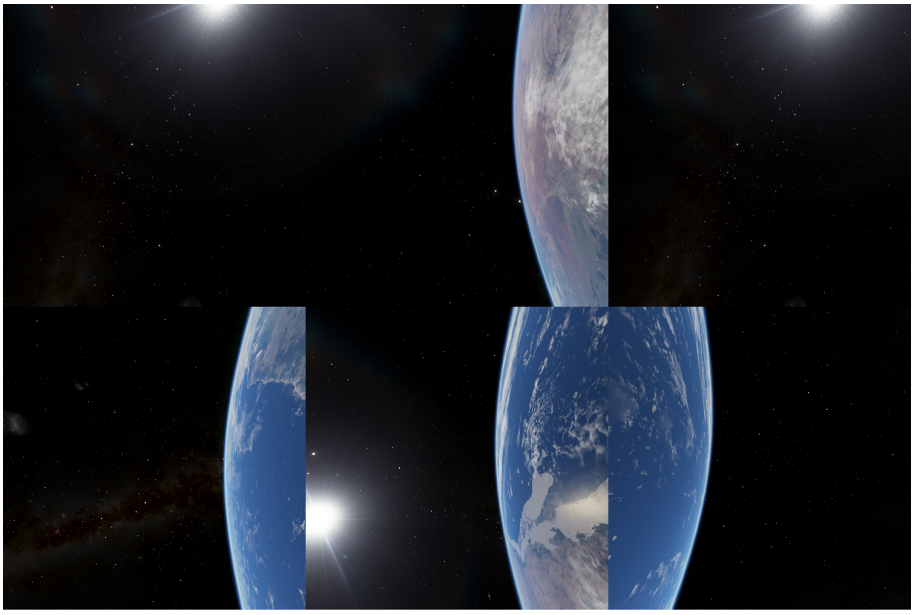


Figure 4: Composite Image of Skybox Axes

From left-right, up-down, each square texture represents the -X, +Z, -X, -Y, +Y, and -Z axes. Instead of mapping the whole cubemap texture to a cube in Blender, each of the 6 face textures were assigned to a single plane (of side length 400m), and placed at 200m from the origin. This was done so that colour correction could be applied to each side individually to account for the angled light source that was inside the 'cube'.

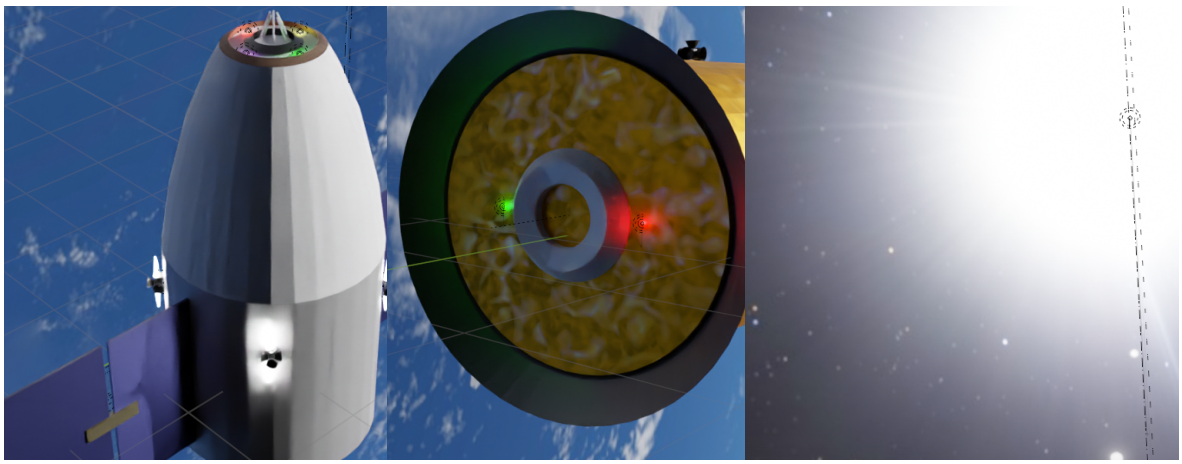
3.3.2 Procedural Textures

Figure 5: Procedural Foil Texture

Using Blenders 'Shading' tab, a procedural texture was created to mimic gold foil often used on spacecraft. This was achieved by assigning a texture of type 'Distorted Noise' onto the cylinder, and then painting it a vibrant yellow in the 'Texture Paint' tab. A lighten mask of pure white was then added on top to accentuate the brightest parts of the procedural texture, simulating light reflection. This texture was saved and is shown in Figure 5.
To apply it to the transfer spacecraft, an image texture was applied to the Principled BSDF as a Base Color, and then the BSDF applied to the surface of the material output. Due to time limitations, a normal map was not created (to simulate actual height differences in the foil), and the texture was left flat.

3.4 Fluids and Light Sources

To show the capabilities that Blender has, multiple light sources and fluids were used for a range of purposes. This included light emission to simulate the RCS gas (Fig 6a), lights around the spacecraft docking ports (Figs 6a & 6b), a sun light source (Fig 6c), and area light sources to correct the lighting on the skybox planes.
As part of the surface characteristics of Blender objects, emission strength can be easily changed using a simple number value. For the 'RCS gas', this number was increased until a suitably bright aura was seen, and it's colour changed to pure white. White was chosen as RCS thrusters typically use a monopropellant as fuel, which reacts exothermically via catalyzation and *not* by oxidisation such as with fire or conventional rocket engines which produce a bright orange flame.



(a) Light Sources on Dragon (b) Light Sources on Transfer (c) Sun

Figure 6: Examples of Added Light Sources

Docking port lights were added to help show the orientation of the two spacecraft, and the distance between them when docking (less distance equated to higher light intensity on the surfaces).

The skybox textures from Space Engine included the softwares sun (Fig 4), and so the scene's light source was placed approximately at the centre of where the sun texture was (see the dotted circle in Fig 6c). Since the source was so far away (about 320m from the world origin), the intensity of the light was increased to 10MW to accurately depict the contrast at the origin.

Once the two spacecraft had docked in the animation, the engine on the transfer spacecraft turned on and accelerated the craft in the $+y$ direction. Blender has fluid simulation built in by default, and the process of creating the fire and smoke was reasonably simple. First, a sphere object was created, scaled down and placed within the engine bell. At frame 720 of the animation, it begins emitting smoke and fire in all directions. The flame colour was set to a white-ish orange, with a high temperature value. To push the fluids in the $-y$ direction, a 'Wind' force field of strength 50 was created (see the circles around the fire in Figure 7, where more spacing between the circles represents higher field strength).

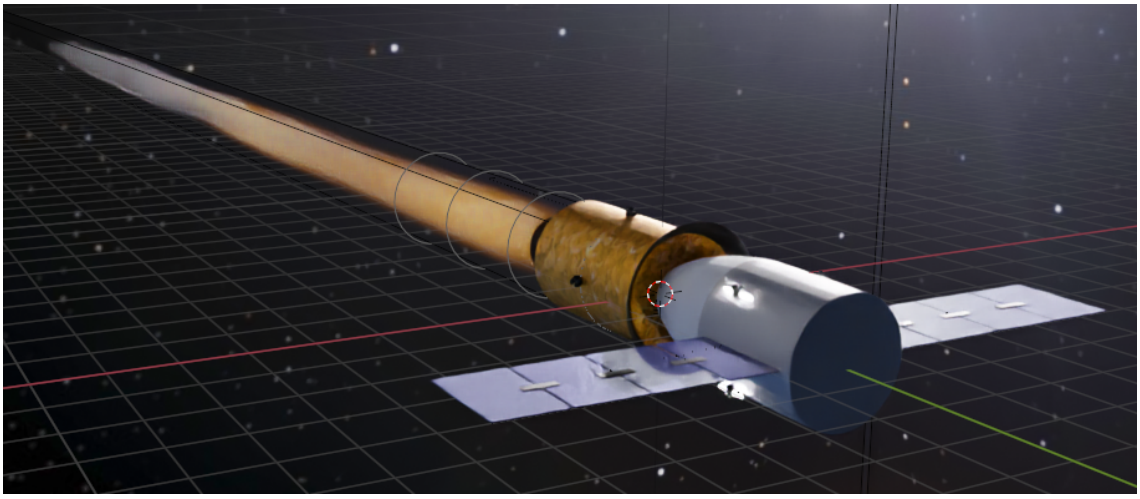


Figure 7: Rocket Exhaust Fire and Smoke

To limit the spread of the flame in the x and z directions, a cube was created with the side length of about the diameter of the transfer spacecraft. This cube was then stretched in the y direction, with the start of the prism just behind the engine and the end of the prism extended to where the end of the smoke is in Fig 7. This cube was assigned the 'Domain' type in physics properties, which assigned the limit to which the fire and smoke could radiate.

3.5 Keyframing and Camera Manipulation

Throughout the course of the animation, the camera rotated and translated over the scene to appropriately show what was happening to the spacecraft. Figure 8 shows the approximate positions of the camera over the whole animation (neglecting the rotation properties due to complexity).

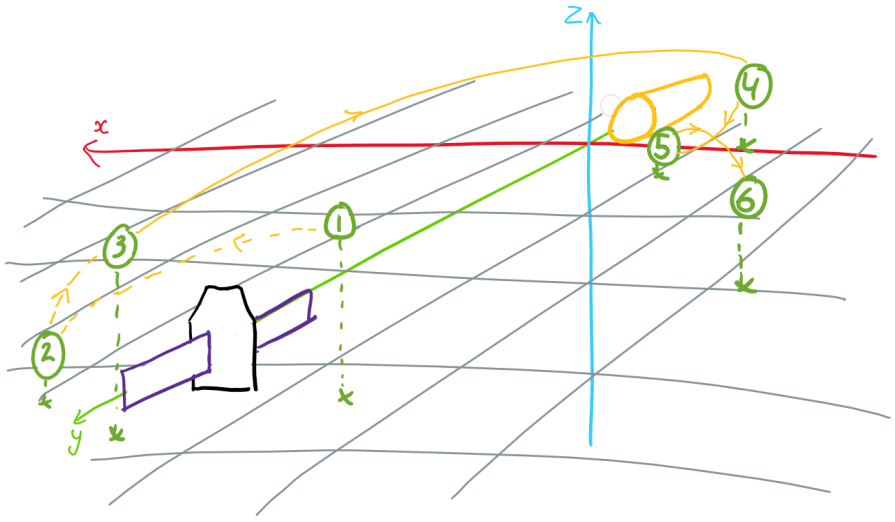


Figure 8: Camera Positioning Over Animation

Blue, Red and Green lines represent z, x, and y axes respectively, with grey squares showing grid at $z = 0$. The intersection of axes represents the world origin. Each numbered circle represents camera position end points before moving to another position, with the animation starting at 1. The green dotted lines represent height from $z = 0$, with the yellow dotted lines showing approximate path between camera positions while Dragon is rotating. Solid yellow lines represent approx. path of camera while Dragon is translating.

Blender's keyframe functionality works by instancing object location and rotation for key points in time. Once two keyframes are set, the object transforms from the first to the second over the length of time between them. The speed at which it transforms are customisable, but for this project linear transformations were chosen due to simplicity and accuracy. Since there's no atmosphere in orbit, no drag forces impact the inertia of the transforming spacecraft and so they're free to transform at a constant (linear) velocity.

4 Animation

The completed animation is available at (with a standalone file available on request):

https://drive.google.com/file/d/1k0dbL2CBjNH3VXPCorVCRNP4DSRKj8DW/view?usp=sharing

Due to a mistake in rendering, frames 820 to 1000 are static. The animation end frame was set to 1000 before rendering by accident (despite the last keyframe being at frame 820), and then rendered overnight. Since the rendering took approximately 5 hours, the project was not re-rendered. No video-editing software was available, and consequently the blank 180 frames remained in the render.

As an output for the animation, either all 1000 frames as individual images or a .avi file was available. As mentioned previously, no video editing software was available, and so the .avi was chosen as the render output which is available in the above link.

5 Discussion

The created render clearly showed orbital maneuvers to a reasonably accurate degree. The quickened pace of the docking worked well for the format, and would not appear out of place from official mission promotion media (of course with some editing and extra refinement). Arguably the most prominent errors were the rocket exhaust plume and the rate at which the objects moved.

In space, the plume of rocket engines expands rapidly past the engine bell due to the lack of atmospheric pressure to compress it into a uniform cylindrical shape. To model this, a much larger domain would be needed, with some extensive customisation of the default blender fire and smoke fluids. Particularly towards the end of the animation, the smoke trailing the fire is prominently visible. Once again, this would not be the case in a more accurate simulation both due to the density of the gas particles (or lack thereof) and the velocity in the opposite direction of the spacecraft. For animation purposes, the smoke velocity was set to 60m/s in order for the viewer to clearly see it as they would expect from a rocket launch on earth. In reality, the exhaust would be moving on the order of hundreds or thousands of meters per second in the opposite direction of the craft. All that said, however, the animation was serviceable and these errors wouldn't be evident to the layman.

The rigidity and linear movements of the objects were clearly more noticeable to a layman, however. With a bit more time, there's no doubt that this could have been remedied by fine-tuning the rate of rotation/-translation to gradually increase while the RCS was firing (see: the brief periods in the animation where bright lights were emitted from the RCS thrusters on the side of Dragon).

6 Conclusions

Through the use of Blender modelling and animation software, a detailed and relatively fluid animation of two spacecraft docking was achieved. After modelling each spacecraft individually, creating a skybox and importing textures where appropriate, a scene similiar to that of low Earth orbit was simulated. After defining the keyframes and camera positions, the animation was rendered overnight and took approximately 5 hours, and finally saved as a .avi file.
Various computer graphics techniques including but not limited to path tracing, multiple light sources, object modelling, texturing, fluid simulation, object translation and rotation, and camera transformations were demonstrated to a high degree. Limitations were discussed, and suggestions for improvement were detailed with reasons to do so.

Animations like the one created for this report would undoubtedly help inform the layman as to how and why orbital maneuvers are calculated, and potentially help nurture an appreciation for rocket science and physics in general.

7 Self-Evaluation

All in all this graphics project exceeding expectations of both myself and my peers. Through the use of blender, a visually impressive final product was created that wouldn't have been possible to such a degree in the timeframe if python and OpenGL were used as per the project proposal. I distinctly remember one of the tutors saying "it won't look as good as you think it will" when I had mentioned my graphics idea in a tutorial, and I would agree with them if I had used OpenGL. That said, I am well aware of errors within the animation; the rigidity of the objects, the lagging in the engine exhaust when the spacecraft translates, the visible borders where the skybox planes intersect with each other, etc.

With more time, I am absolutely confident that a better animation could have been developed. Over the course of creating this graphic, I had learnt more than I had thought I would - this was my first use of Blender and animation software ever.

Finally, the grade I believe I deserve is a P+14. I think that the P+7 criteria is satisfied in it's entirety, and points [2, 3, 4, 5, 7, 8, 9] on the 'Further Additional Marks' criteria were satisfied. I guess points [1, 6] are on the marker to decide whether I had satisfied the criteria or not, as I am at a half-way point on criteria 6 and quite unsure as to the brevity of the report (potentially not satisfying point 1).

8 Appendices

8.1 Blender Learning Process

As part of learning how to use Blender within 2 days, I had searched YouTube for the quintessential 'Donut Tutorial'. After finding a brilliant series of videos from the self-proclaimed BlenderGuru, I had created this delicious masterpiece below.

In all seriousness, I am just looking for an excuse to show off the donut, although this does show how I learnt how to use procedural textures (see the crumbing and distribution of sprinkles) which were subsequently applied to the report both for texturing and fluid effects.

References

[1] Romanyuk, V, 2021. *LICENSE & CREDITS* Available at: http://spaceengine.org/manual/license/

[2] Lampel, J, 2019. *Cycles vs. Eevee - 15 Limitations of Real Time Rendering in Blender 2.8* Available at: https://cgcookie.com/articles/blender-cycles-vs-eevee-15-limitations-of-real-time-rendering