An Authentic Physics-Based Traction Trebuchet Simulation in Unreal Engine 5

Amy Broeders

***Abstract*—History is commonly perceived as boring and irrelevant to modern youth due to the inaccessible and often dark nature of literature surrounding it. Video games provide a unique medium to retell stories of epic battles and survival against threats that we rarely worry about today, allowing a broader audience to engage with and learn from our rich history. Traction trebuchets were used for centuries in medieval siege warfare, yet the researcher didn't know of their existence prior to starting this project due to the lack of representation in modern media. This project sought to use this modern medium to simulate the machine that would have saved and destroyed many lives with accurate physics and create a fun experience to discover history in a comfortable and accessible setting. From working on this project, it has become clear that the initial vision for this project was ambitious and the game has not lived up to its full potential, however, the physics simulation and accuracy of the trebuchet have been continually prioritised as this will be the most useful artifact to use in future work in this area. My initial work has provided context on some of Unreal Engine's current physics capabilities, documentation of solutions to bugs created during development that may trip up new learners to Unreal Engine, and a prototype of a traction trebuchet simulation for users to interact with.**

***Index Terms*—Software Engineering, Trebuchet, Simulation Game**

I. Introduction

A. The Problem

The traction trebuchet, also known as a mangonel, is a medieval siege weapon invented in China between the 5th and 3rd centuries BCE, used in Europe from as early as the late sixth century (CE), and estimated to have been in use until at least the 12th century when it was superseded by the counterweight trebuchet [1] [2] [3]. The traction trebuchet was a popular anti-personnel weapon, yet it is not particularly well-known in the 21st century, and it is often overlooked or recreated inaccurately in modern media. As such, this piece of history goes largely undiscovered by the public. The construction and use of a physical traction trebuchet requires a large area of free land as it can launch projectiles up to 120 metres (while also being able to misfire 50m backward and 10m to either side, totalling a minimum of 3400 square metres of space required for safety) [2]. So, for enthusiasts to authentically recreate the traction trebuchet, they face safety concerns, require permission to fire projectiles at high velocities on a large empty piece of land (which will likely damage the land), and require resources such as wood, rope, and leather. With empty land becoming rarer due to population growth and expansion of housing, opportunities to achieve this will also become rarer with time. These barriers won't stop everyone, but it does make it infeasible for many and will hinder others, and will certainly keep anyone from the wider public away from this niche activity.

So, how can we provide a safe and accessible alternative to building a traction trebuchet? This project suggests that this problem can be solved by leveraging modern technology to create a digital equivalent for people to interact with in the comfort of their own homes, instead allowing individuals to explore and experiment with the tool and deepen their curiosity by discovering more resources on this topic available through the internet, with accurate physics and authentic structure to contextualise the trebuchet within its era. This would include a 3D model of a traction trebuchet with a structure that is similar to historic diagrams or pictures of traction trebuchets (although there were several variations on the structure, the core ideas remain the same), which would have moving parts that could throw a projectile, a hook that could be altered separately to change the angle at which the sling is released, functional ropes and a stiff sling to imitate leather. With this digital tool, history like the traction trebuchet can be easily brought to a broader range of people, as well as serving to entertain enthusiasts who may already understand the function of the trebuchet, or even being used in an academic setting such as classrooms.

B. The Solution

This project developed a digital physics simulation of a traction trebuchet. This simulation is made up of a low-poly 3D model with separate parts using physics abilities in Unreal Engine 5 (UE5) to throw a projectile. The simulation can be viewed from all around and fired by the user, and by tweaking the hook or the code in the UE5 editor, the behaviour of the trebuchet can be altered as it would be with the physical version. Alongside this, other deliverables created include a paper discussing Unreal Engine 5's physics capabilities and its advantages and disadvantages in the context of the project, and documentation of the fixes to some big issues faced in 3D modelling and physics simulation.

Due to the project's deliverables all being in digital format, it is an environmentally friendly solution (especially over building trebuchets, which requires construction materials, and more specifically cutting down trees, which negatively affects Goal 15 from the Global Goals). Low poly art styles also offer lower computational requirements due to the

This project was supervised by Simon McCallum

computational cost associated with rendering each polygonal face on the screen (the computer has to calculate what is seen/obscured and where each face is displayed on the screen) which reduces the impact the project has on climate change and thus improves its effect on Goal 13 [4]. However, it is worth noting that the physics calculations for some parts of the project, while minimized, are computationally intensive themselves. Unfortunately, this is a natural requirement for a physics-based project. Future iterations of the project will also be sustainably aligned unless the scope broadens (e.g. physical components being added). The tool's educative nature also relates to Goal 4 from the Global Goals, as it can be used to educate people in history.

Due to the change in outcome from the original goal presented in the preliminary report, the original user evaluation metrics are no longer applicable to this project. Instead, three metrics are presented: the throwing distance of the projectile, which can be converted to metric measurements and checked against maximum distances in the existing literature; the number of bugs/misfires out of a total number of throws; and the framerate at which the application runs. The projectile was thrown 89.61 metres on average, though there were multiple instances of it going over the possible maximum suggested by Hjesvold et al. [2]. 35% of trebuchet firings had a misfire or bug occur, and the frames per second remain at 120 consistently during the simulation.

II. Related Work

A. Existing Solutions

There are very few existing traction trebuchet simulations currently available, and existing solutions are less accurate and comprehensive than the proposed solution. Age of Empires IV is a game that offers gameplay including a traction trebuchet as a unit unique to the Mongols. Age of Empires is a strategy game series where you play as a civilization and compete against AI or other players to be the first to reach one of several win conditions, such as including destroying all other civilizations' units and buildings. The model of the trebuchet is decent, although its structural integrity is questionable (see Fig. 1), with the main axle attached on either side to separate pieces of wood which means great stress will be put on the connection to these side pieces as opposed to the axle sitting directly on top of tall vertical struts as in Hjesvold and McCallum's paper [2].

The traction trebuchet in Age of Empires 4 belongs exclusively to the Mongols, even though it originated in China, was instrumental in the expansion of the Islamic empire, was brought to Greece by the Avars, and became widespread in the eastern Mediterranean [1] [2]. In the scope of the game where having units unique to different civilizations provides variety to the gameplay, it makes sense to limit access to the weapon to one civilization, however, it does misconstrue the history of the weapon. It is listed as being weak against melee units, however, in reality, traction trebuchets were primarily excellent anti-personnel weapons as well as being used as siege weapons [5]. Finally, as there

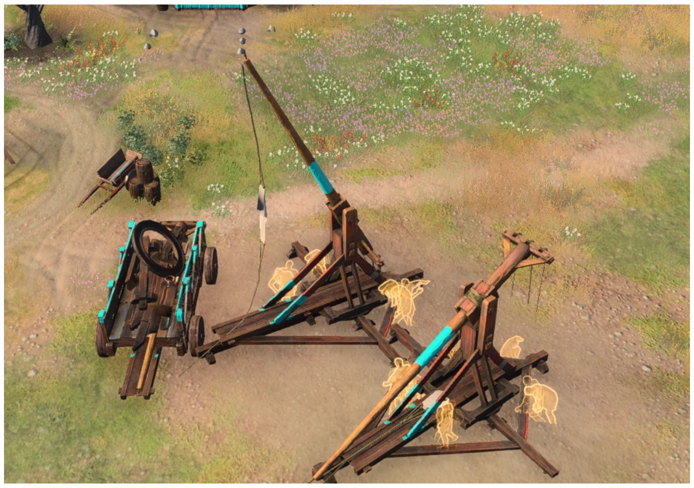
Fig. 1. A traction trebuchet simulation from the game Age of Empires IV. It has thin sticks holding up the axle on either side, which is most likely for aesthetics as it is not practical from an engineering perspective. Source: [5]

are no consistent units of scale in the game, it cannot be said to accurately depict the physics of a real-life traction trebuchet as the distance it can fling projectiles is unknown (though estimates would say the in-game throwing distances are inaccurate).

Another strategy game with very similar gameplay to Age of Empires called Empire Earth III also has a traction trebuchet unit that suffers from the same problems with the structure of the trebuchet and the lack of known distance units, if not worse (see Fig. 2) [6]. Empire Earth III has three civilizations in contrast to Age of Empires, and the traction trebuchet is a unit available for the "Far Eastern" civilization, which is based on East and Southeast Asia, again partially incorrect in its limitation of the unit, with the counterweight trebuchet instead being attributed to the "Western" civilization. This project fills this gap in the market by providing an accurate depiction of a structurally sound trebuchet structure and the physics involved and also avoids misconstruing the history of the weapon.

B. Theory

Aside from simulations, there is plenty of research surrounding traction trebuchets in both a historical and practical engineering context. There are several components to the traction trebuchet to consider: the arm, the axle, the sling, the ropes connecting the sling to the arm, the pulling ropes, the hook, and the base. Each has unique or varying attributes to consider in the functionality of the trebuchet.

The ropes attaching the sling to the trebuchet attached to the trebuchet in different ways - one "attached firmly" perhaps tied on or nailed into the arm, and the other "looped over a metal prong" (the hook), the idea being that the loop would come loose at a certain angle, and thus the kinetic force pulling the projectile with the arm of the trebuchet

Fig. 2. An existing traction trebuchet simulation from the game Empire Earth III. Again, the strut does not look very secure. Source: [6]

would be released and it would travel forwards at its existing velocity [1]. The adjustment of the prong and the length of the ropes attaching the sling thus causes variance in how far and high the projectile flies (the longer the rope the later the release point) [2].

More ropes were used by the crews manning the traction trebuchet, as they produce the kinetic force that has to be greater than the force of gravity acting on the arm and projectile to cause the forces to be unbalanced and pull the heavier end of the arm into the air with the sling [1].

To accommodate the unbalanced pulling force generated by the crew standing at the back and sides of the trebuchet, Hjesvold et al. reinforced the bracing behind the ”uprights” to ensure stability during the arm’s forward swing (see Fig. 3) [2]. The base is not expected to move significantly, with these bracings absorbing the forces that aren’t transferred to the projectile.

The arm of the trebuchet is generally slightly bendy, as it is ideally constructed from green wood which is less stiff than dried wood due to its additional moisture, and this provides protection from breaking and some spring forces to absorb any uneven forces from the strength and timing of the crew to allow a smoother movement of the projectile [2]. It is also worth noting that there are two main styles of bracing on the arm itself: an axe style like in Fig. 3, or a rake style, where the ropes come from a horizontal piece of wood attached at the end of the arm more akin to the style of the Age of Empires traction trebuchet in Fig 1. [2]. Due to the 2-dimensionality of the pictures from the era of the traction trebuchet’s usage, it is difficult to tell which was used in many pictures, however, the

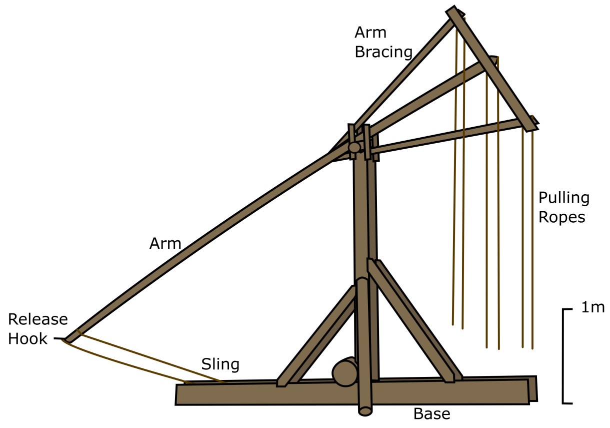

Fig. 3. A diagram of the traction trebuchet. Source: [2]

Maciejowski Bible depictions seem to show vertical bracing [7]. The axle as suggested by Hjesvold et al. sits between the arm and one of its bracings and rotates with the arm as needed.

C. Tools and Methodology

The main tool used in this project that needed to be chosen was the game engine. This was important to the project since the features available in a game engine will determine the way that the game is developed, and potentially influence the quality of the resulting product. There are many options nowadays, such as Unreal Engine, Unity, Lumberyard, and CryEngine, all of which are solid options for 3D graphics. Of these options, the researcher has existing academic practice with Unreal Engine and some experience with Unity as well, so they were naturally the top choices for this project. Due to the wide variety of engines available and the researcher’s existing experience, Unreal Engine was placed as a primary candidate for the project and investigated for potential drawbacks, rather than ruling out all other options. Given that multiple engines could likely support the requirements of the project, factors such as cost and experience significantly influenced the decision-making process.

The latest version of Unreal Engine, Unreal Engine 5, appears to boast superior physics simulation (along with a few other advantages such as superior animation) with its new Chaos Physics engine, although some suggest it is less reliable and complete than the previous engine used, Nvidia’s PhysX, due to less rigorous testing [8] [9]. It is also well known for its flashy graphics, which aren’t necessary for a low-poly game but may nonetheless enhance the experience. Unreal Engine is also free to use for projects of this nature, with royalties owing only after earning over $1 million USD, which makes it advantageous for a project like this. It has reasonable documentation, although due to its overlaps with the previous version, Unreal Engine 4, many tasks can be figured out by reviewing the older documentation instead, which makes the documentation a lot more rigorous holistically.

Physics simulation projects have been implemented within versions of Unreal Engine before, with conversions of distance in Unreal's units to real-life units being calculated to help with measuring and improving their accuracy [10]. This further proves that Unreal Engine is a strong candidate for creating the simulation project with.

The most complex part of the trebuchet system identified and researched for this project was the rope. As opposed to solid objects, rope has tension forces and flexibility, so it behaves very differently. Unreal Engine 5 offers a "cable" component, which is a series of connected particles where only the end particles may be attached to something, and the particles between the two ends dangle freely, using the Verlet Integration technique [11].

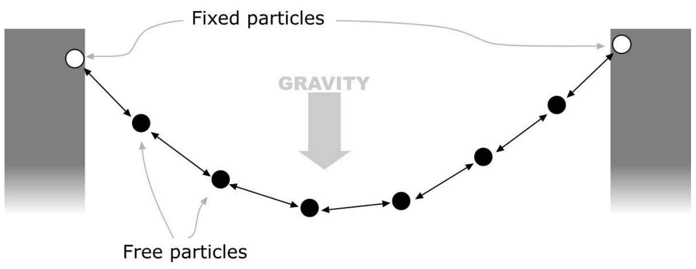

Fig. 4. A diagram illustrating the components of a rope in Unreal Engine 5 Source: [11]

There are alternatives such as the VICO Dynamics plugin which offers its version of cables as well as other soft-bodies such as cloth, or a more custom way to achieve the effect would be to model rope with a cylindrical model and apply bones and physics constraints to create the required behaviour [11] [12]. Unfortunately, the VICO Dynamics plugin is a paid service, and the cost is not warranted in comparison with the features offered by the existing cable component.

The major flaw with the cable component that UE5 offers is that ropes will not collide with one another. This may cause a visual issue in that ropes will most likely swing into (aka through) each other when the trebuchet is fired, however, it is not anticipated that this will make a major impact on the realism of the trebuchet. The Cable component also has great added computational cost from enabling collision with rigid bodies and stiffness, and collisions are still in development and are prone to bugs.

Unreal Engine features two ways to program custom behaviour for objects - blueprints, an Unreal Engine native visual coding system where developers mostly drag and drop nodes to build behaviour from available functions, or C++, a scripting language first released in 1985. C++ offers greater complexity and customisation to control what happens in-game, however, it is also more complex to program in than the visual, dropdown menu style of blueprints where you can search for functionality you may want to use. The researcher has experience with blueprints and minimal experience with C++, so this was chosen as the standard for development on this project, with C++ only being added to supplement blueprints if the required behaviour could not be implemented with blueprints.

GitHub Co-Pilot is a recently released AI tool that can be used to aid programming, for example with C++. GitHub Co-Pilot can generate correct and optimal solutions for certain fundamental problems, however, the quality of its responses varies, and it may produce flawed or suboptimal results that require a fairly skilled developer to identify [13]. However, with minimal C++ knowledge, having an AI pair programmer assist with some syntax and basic functions will likely accelerate development over working alone as less time would be spent researching implementation and bugs for simple features.

For modelling the trebuchet itself, I also needed to find a 3D modelling software to use. Again, there are many, such as Blender, Maya, SketchUp, and ZBrush, and the decision process here was much the same as for the game engine - the researcher has prior experience with Blender, it has strong documentation due to being an open-source project, and it is also free to use.

III. Design

The initial design for this project was multi-faceted. There was the physics accuracy of the trebuchet, the historical context worked into the game through dialogue or menus, and the entertainment value of the gameplay itself. With the scope being scaled back to focus on the trebuchet tool being of higher quality, the requirements have changed to focus more in-depth on the traction trebuchet physics and structural design. The non-functional requirements generally stayed the same, with performance such as high frame rate and minimal bugs encountered being particularly important, but simple UI, inputs, and loading of the game being key for a friendly user experience.

To accurately represent these various facets and the way physics acts on each of them, we wanted to present them in 3 dimensions. Alternative game designs were discussed in early planning, such as a 2D game akin to Angry Birds, however, the researcher believed that a 3D representation would provide a more authentic and holistic experience of the trebuchet.

The trebuchet consists of a 3D model split into several independent components - the base, the arm and the axle, the sling, the hook, and the projectile. The arm and the axle were kept together as a single mesh as a simplification where separating them wouldn't have been likely to change much of the physics operations but would have opened the project up to more bugs/mistakes and increased complexity. However, the hook could have been kept in the same mesh to further reduce complexity and still would have been functional, but this would take away from the customizability of the trebuchet in the simulation (changing the hook angle), so this had significant relevance to the physics capabilities of the project and was kept separate. The base can be reasoned to be one mesh as in reality each piece of wood would be constrained together to act as one object with rope and/or nails, so its behaviour as a single object is appropriate in the given context.

A low-poly art style was selected due to its lower computational requirements and associated reduced energy costs for environmental well-being. This means that there are fewer faces than on a highly detailed model, and is a significantly friendlier art style - for the planned game to be entertaining, the researcher did not want the style of the game to be too serious to appeal to a wider audience. There is no strict definition for what is considered ”low poly”, but an estimate suggests between 1 and 10k polygons would fall into this category. Generally, it is recognised by simple shapes, simplifying what an object would be expected to look like. High-poly meshes and high-resolution textures were discarded because of their computational expense, higher artistic skill ceiling if additions or editing were required, and the more serious tone communicated by the detail.

For applying physics to the trebuchet, an animation could have been created to create the perfectly customised movement of the trebuchet. This would have likely been a faster process to implement, as the model would not be relying on the physics settings in Unreal Engine being correct, and instead, more time could be spent setting up the ropes and sling, which would still need to be accomplished separately. However, part of the proposal for this project suggested that modern technology had advanced capabilities to compute complex physics simulations, so utilising the full suite of physics features and settings available within an advanced engine offers better insight into whether this is viable for future projects. This would also open up opportunities for users to experiment with customizing the trebuchet's movement. For example, a UI could be designed to let users adjust the hook angle, rope length, and/or pulling force applied to the front ropes, thereby allowing for greater control over the projectile's velocity. This level of customization goes beyond the simple adjustments possible for a fixed animation such as rotating the trebuchet or changing the size or weight of the projectile, making for a more interactive and engaging experience.

Most of the modelled components of the trebuchet (i.e. the non-rope parts) are static meshes that do not change their shape in any capacity (the base, the arm and axle, the hook) due to their real counterparts being made from solid materials (wood, metal of the hook), however, the sling is modelled after leather, which is a thick but bendy material. It could have also been made as a static mesh and just held the curved shape it starts in the whole time, however, this limits the realism of the material as it would typically open up to a more flat shape, so instead an articulated model needs to be created where it can change shape (from a curved U shape to an open _ shape). Given the possible bend in the green wood used for the arm, it could be argued that there needs to be some kind of articulation in this mesh as well, however, as a starting point for this project it was kept static as this would also function in real life, and the simulation is not at risk of breaking, so it would only slightly alter the physics by absorbing some of the forces applied, and it would add another layer of complexity to implement this.

The player's perspective in the simulation (and in the originally proposed game) was decided to be first-person rather than third-person, as there was not intended to be any emphasis on the character being played by the user, and first-person was chosen as it would be more immersive for the player and provide a better view of the trebuchet.

Other planned deliverables for this project include a paper discussing Unreal Engine 5's physics capabilities, and documentation of the fixes to some big issues faced in the creation process. Another way that was considered for displaying the fixes to issues was to create short video tutorials showing the problem and how it was rectified, however, with the provided time constraints and the primary goals of the project being focused on the trebuchet outcome itself, this ended up being beyond the scope of this project. The paper discussed could have also been written instead as a case study of the project as it was developed, documenting the way that the physics features were used in the project and how it helped or hindered progress. This was ruled out as research was needed before beginning the project to understand existing solutions and features available in UE5, which would not be covered in a paper structured this way.

The use of Unreal Engine 5 provides economic and technical benefits to the project in the long term. As Unreal Engine 5's payment model only charges for high-earning projects, there are no foreseeable economic requirements of the design, and if it were to be used in a monetized way in the future then it would most likely be economically beneficial for everyone involved. It would need to be hosted on a website such as itch.io or Steam to be monetized effectively, which may incur some costs, whether or not it makes a profit, but that would be for a future project based on this to consider. The technical benefits are that the engine is owned by a very large and successful company, Epic Games, and so will continue to receive support for the foreseeable future, meaning that if future projects wish to update this it will be simple to do so, although the packaged version of the current project should continue to work no matter what happens to the original engine as it creates all of the binaries required to run the application.

IV. Implementation

The trebuchet was made from an existing mesh asset provided by Synty Studios, imported into Blender by exporting it from Unreal Engine as an .FBX file, and edited to fit the axe style of a traction trebuchet as discussed previously. The original mesh (see Fig. 5) was also one solid piece, so here it was also separated into the different components that would move in the simulation - the base, the arm and axle, and the hook. The original mesh had static ”ropes” and a bucket for the projectile. Naturally, the ropes were not going to function as intended, so they were removed, and the bucket was also not the intended representation to align with the suggested leather sling from Hjesvold et al. so this was removed as well. The hook also needed to be altered from the initial design of a complete ring to a sharp pointed spike, and the arm and axle were moved up to above the crossing beams to take advantage of the structural support those beams provide, and further supports were added and removed in appropriate places. The shapes of the wooden pieces were also rounded in Blender to look more akin to round tree trunks would be used without too much shaping in traditional traction trebuchet building. All of the meshes were exported from Blender as .FBX files to be imported back into Unreal Engine 5. The use of the low-poly models provided by Synty Studios has allowed the trebuchet to be more computationally efficient without compromising on the aesthetic quality or historical accuracy of the trebuchet structure.

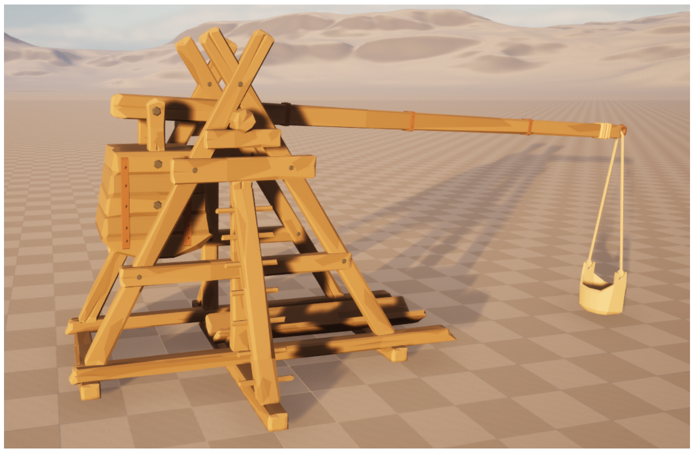
Fig. 5. A screenshot of the original model provided by Synty Studios.

In this design process a few new model faces were created and no longer had the UV mapping that Synty Studios had set on the original model. Their material was one large material with many different types of material on it (e.g. wood, stone, signage, colours) to minimise storage space and computational power for an asset pack with low-fidelity artwork, meaning that a whole variety of conflicting colours displayed on the faces without proper UV mapping. This required some investigation but by going back into Blender and altering the UV maps of the affected faces, it was able to be brought into Unreal looking smooth (see Fig. 9). This means that a second material exclusively for these parts didn’t need to be created, saving some storage space and computational power to optimize the application and reduce the carbon footprint of this project.

Upon importing these meshes into Unreal Engine, they originally behaved very strangely. This is because Unreal Engine attempted to create simple collision shapes to represent each part, but when running the simulation, they were already colliding from where the arm was placed on the base. Each mesh needed to have collision shapes created manually, as the automatic tools were not advanced enough to handle the many open spaces in each mesh (see Fig. 6).

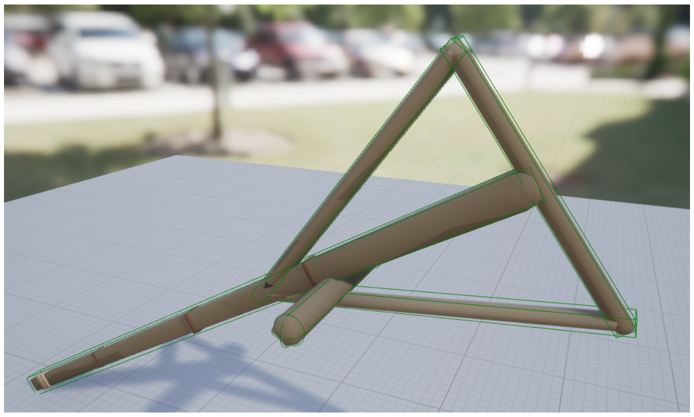
Fig. 6. A screenshot of the collision boxes on the Arm and Axle mesh.

Another model was created at this point to replace the original sling, this time with a low poly U-shaped sling with segments attached to bones in Blender to allow the mesh to be altered in the desired fashion (opening out to a flat sheet). Unfortunately, this mesh hasn’t been implemented correctly in Unreal Engine just yet, and still maintains its U shape throughout the motion of the arm, though some buggy movements have shown it can ”open up” in Unreal Engine, so it is presumed to be some settings that haven’t been fixed yet. When brought into Unreal Engine, an ”armature” from Blender is split into three assets - the Skeletal Mesh, the Skeleton, and the Physics Asset, each of which has different physics settings/components that need fixing, like the collision components on the Physics Asset, which needed to be created manually as the defaults were not useful for the model.

The main hurdle in implementing the design was utilising the cable component. It behaves unexpectedly frequently, and the settings can be unintuitive. The collisions on the rope are still in development, so do not work consistently, and the component is designed to only connect to one socket, with the other end being set based on the origin of the component, so it has to be a child of the component it attaches to to move with it. For the rope that connects to a specific place on the arm, a Scene object was created as a child of the arm and placed in the desired spot, and the rope was then made to be a child of that object so that it moved with that part of the trebuchet.

The cable itself is essentially visual only, as it does not constrain the two objects it is connected to. It also cannot

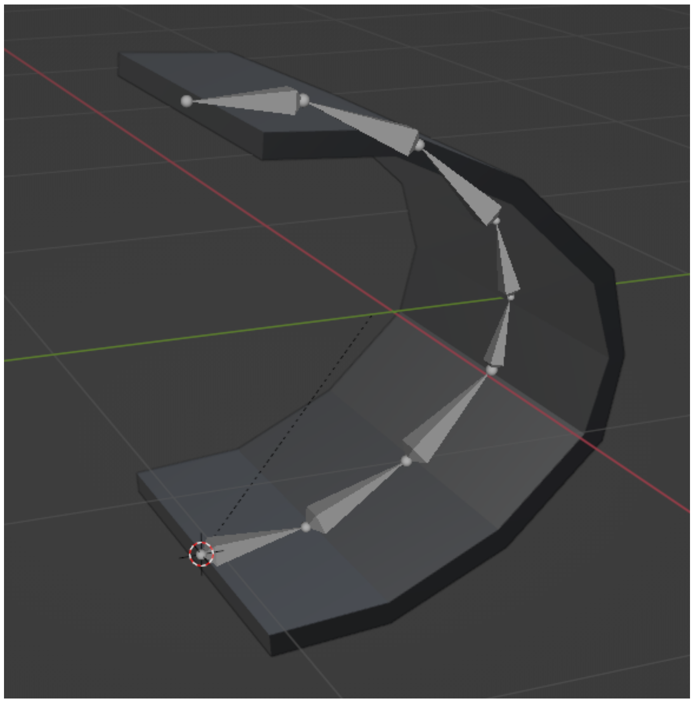
Fig. 7. A screenshot of the the sling in Blender.

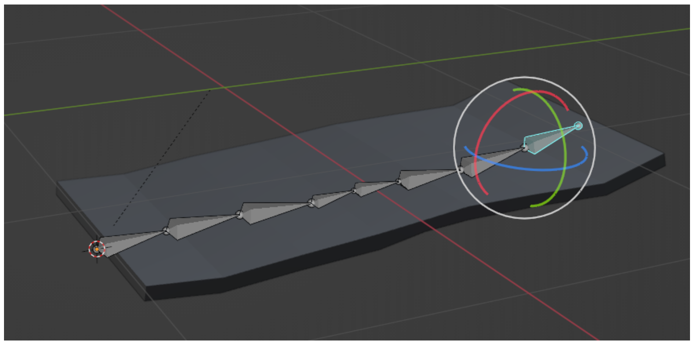
Fig. 8. A screenshot of the sling in Blender's Pose Mode flattened out.

create a loop around connecting to itself, so another mesh was created to hook over the spike and create the effect of coming free and flying backwards with the sudden change in force. This rope loop was a simple adaptation of the torus default mesh provided by Blender. The cable component also could not have specific UV mapping applied to it as it is an Unreal component rather than a mesh, so another material needed to be created to make these ropes look appropriate with the style of the rest of the trebuchet.

For connecting the sling's movement appropriately to the arm and the rope, physics constraints were added to the trebuchet blueprint. A physics constraint was created for each rope connecting the sling, so that when the trebuchet fires, the rope loop coming off of the hook allows the sling to open up and let the projectile go. Another physics constraint keeps the arm movement restricted to forward rotation, and one was also required to attach the hook to the arm properly (as they are separate components, both with physics applied to them, they need to be constrained together or they act independently). Finally, 3 more ropes were created to represent the pulling ropes at the front of the trebuchet. As these ropes cannot be used in the simulation to apply a force to the trebuchet, minimal settings were used here to cut back on computational power to retain performance and reduce environmental impacts.

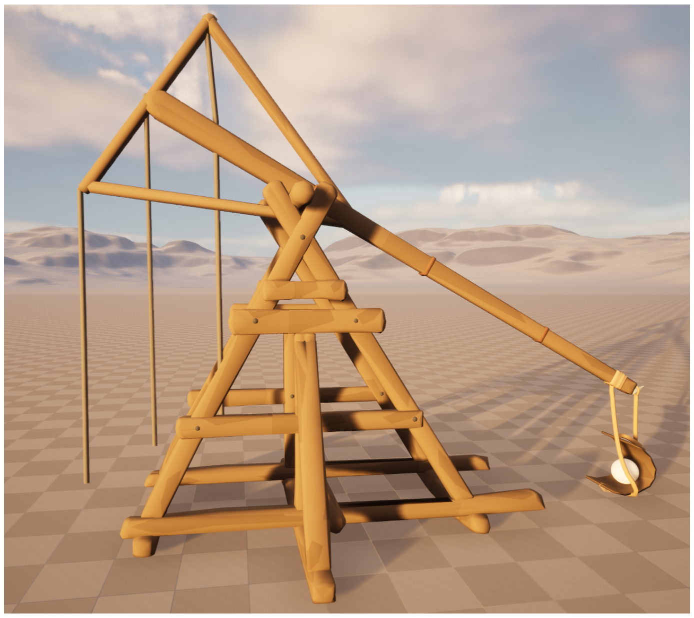
Fig. 9. A screenshot of the final model based on the model provided by Synty Studios.

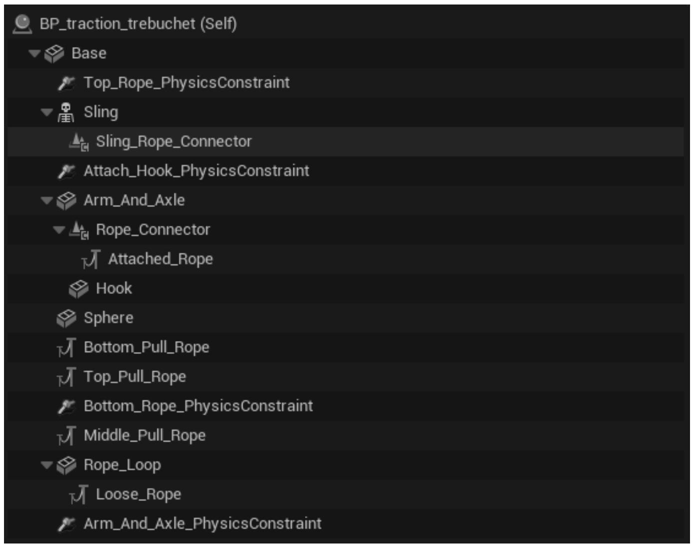
```
BP_traction_trebuchet (Self)
  Base
    Top_Rope_PhysicsConstraint
    Sling
      Sling_Rope_Connector
    Attach_Hook_PhysicsConstraint
    Arm_And_Axle
      Rope_Connector
        Attached_Rope
      Hook
    Sphere
    Bottom_Pull_Rope
    Top_Pull_Rope
    Bottom_Rope_PhysicsConstraint
    Middle_Pull_Rope
    Rope_Loop
      Loose_Rope
    Arm_And_Axle_PhysicsConstraint
```

Fig. 10. A screenshot of the components used to make up the trebuchet blueprint.

V. Evaluation

As the project has changed in scope significantly from its original plan, the means of evaluating it have also changed somewhat. With the game as the main outcome, the project was going to be evaluated primarily with user testing. Instead, the following metrics are used.

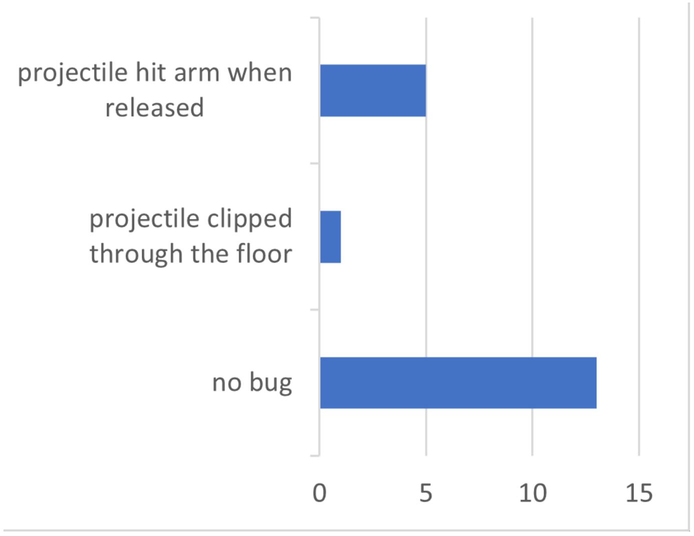

Fig. 11. A graph depicting the number of bugs encountered when firing the trebuchet.

A. *Percentage of Misfires/Bugs*

To get an idea of how consistently the trebuchet fires correctly, 20 trials were run with each type of bug recorded as it came up. Two main bugs are occurring as of the moment - the projectile sometimes clips through the ground instead of bouncing, and it sometimes misfires and hits the arm when it releases. Including the bug of the projectile clipping through the ground (which is otherwise a successful throw), 7 out of 20 throws had some misfire or bug, or 35% (See Fig. 11). This is unreasonably high for something to release, but the majority of throws are still successful.

B. *Throwing Distance*

Based on the units provided by Herrlich et al., the 20 trials used in the previous section were also measured to gain an average of how far the projectile flew in metres on average [10]. This was measured by pausing the simulation when the projectile hit the ground and viewing the movement of the projectile in the Y direction (corresponding with the direction the trebuchet is facing). This was instead of waiting for the projectile to stop, as the projectile currently has very low friction it rolls for a long time, thus making the results less accurate. It threw the trebuchet a mean average of 89.61 metres with a median of 79, which is within the possible range indicated by Hjesvold et al., however, the maximum throw was 173.7m which is beyond the maximum of 120m suggested [2]. This also includes the misfired shots from the previous section, excluding those that put the mean average up to 102.7 and the median to 86.6. These are reasonable values, excluding the exceptionally far throws, but there is a large variation between the maximum (173.7m) and the minimum (53.98) even after excluding the misfires, so it suggests that further testing may be necessary. It is not yet clear why there is so much randomness to these throws as the conditions should be the same each time.

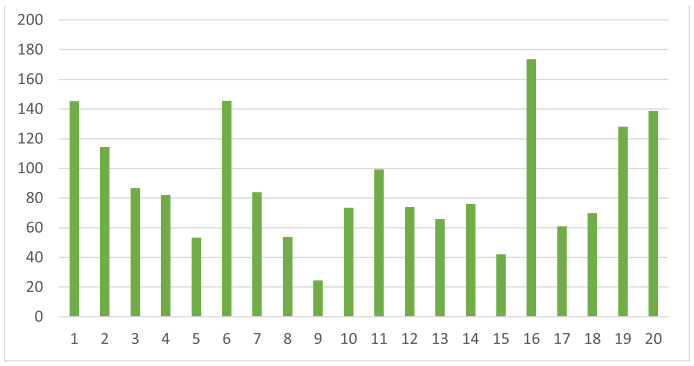

Fig. 12. A screenshot of the components used to make up the trebuchet blueprint.

C. *Frame Rate*

When simulating the Unreal Editor, it very comfortably maintains approximately 120 FPS on my PC, with a slight dip sometimes when the projectile hits the ground. This will likely vary for users on other devices, however, the consistency suggests that there is no one part of the simulation that is particularly computationally expensive.

VI. Conclusions and Future Work

As of the present moment, the project is functional but inadequately polished, and lacking a lot of the original functionality proposed. However, the physics capabilities have been explored deeply here, and there is plenty of room for future work, both on refining the project and building from it. Naturally, there is the possibility of integrating this tool into a game with the originally proposed gameplay mechanics to enhance the understanding and entertainment value of the trebuchet.

Alternatively, a physical rig could be created with a pulley that could measure the velocity of someone pulling on a rope in real life and convert that to how the trebuchet would react, to create a unique Human-Computer Interaction experience and offer a fun challenge to heighten the realism of interacting with the trebuchet.

It would also be possible to amplify the educational aspect of this project, and instead create an interface around the viewport that allows you to view statistics such as distance travelled and current velocity, and provide functionality to alter the angle of the hook, projectile type/weight, and rope length. This project in particular would require further fine-tuning of the physics as it would be working in a wider variety of scenarios.

Acknowledgments

Thanks to Simon McCallum for supervising this project, Synty Studios for providing the original mesh that the traction trebuchet was created from, and my partner Corban for supporting me throughout the year.

References

[1] P. Chevedden, “The Trebuchet,” Jan. 2013. [Online]. Available: https://www.medievalists.net/2013/01/the-trebuchet/

[2] S. Hjesvold and S. McCallum, “Traction Trebuchet,” *EXARC Journal*, no. EXARC Journal Issue 2018/3, Aug. 2018. [Online]. Available: https://exarc.net/issue-2018-3/at/traction-trebuchet

[3] P. Purton, “After “Rome”,” in *A History of the Early Medieval Siege, c.450-1200*, ned - new edition ed. Boydell & Brewer, 2009, pp. 1–36. [Online]. Available: https://www.jstor.org/stable/10.7722/j.ctt14brvp6.9

[4] N. L. Webster, “High poly to low poly workflows for real-time rendering,” *Journal of Visual Communication in Medicine*, vol. 40, no. 1, pp. 40–47, Jan. 2017, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17453054.2017.1313682. [Online]. Available: https://doi.org/10.1080/17453054.2017.1313682

[5] “Traction Trebuchet,” May 2023. [Online]. Available: https://empireearth.fandom.com/wiki/Traction_Trebuchet

[6] “Traction Trebuchet,” May 2023. [Online]. Available: https://ageofempires.fandom.com/wiki/Traction_Trebuchet

[7] “Picture Bible - Medieval & Renaissance Manuscripts Online - The Morgan Library & Museum.” [Online]. Available: https://ica.themorgan.org/manuscript/thumbs/158530

[8] M. Laaksonen, “Unreal Engine 5 Compared to Unreal Engine 4,” Nov. 2022. [Online]. Available: https://www.linkedin.com/pulse/unreal-engine-5-compared-4-mika-mike-laaksonen

[9] M. Lentine, “Chaos Scene Queries and Rigid Body Engine in UE5,” May 2022. [Online]. Available: https://www.unrealengine.com/en-US/tech-blog/chaos-scene-queries-and-rigid-body-engine-in-ue5

[10] M. Herrlich, R. Meyer, R. Malaka, and H. Heck, “Development of a Virtual Electric Wheelchair – Simulation and Assessment of Physical Fidelity Using the Unreal Engine 3,” in *Entertainment Computing - ICEC 2010*, ser. Lecture Notes in Computer Science, H. S. Yang, R. Malaka, J. Hoshino, and J. H. Han, Eds. Berlin, Heidelberg: Springer, 2010, pp. 286–293.

[11] “Cable Components.” [Online]. Available: https://docs.unrealengine.com/5.0/en-US/cable-components-in-unreal-engine/

[12] “VICODynamics: Rope/Cloth/Soft-Body Simulation Plugin in Code Plugins - UE Marketplace.” [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/vico-dynamics-plugin/questions

[13] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, “GitHub Copilot AI pair programmer: Asset or Liability?” *Journal of Systems and Software*, vol. 203, p. 111734, Sep. 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121223001292