CS7GV5 Assignment 3:
Animation Principles and Physics - 'Piggyback'

Timon Sommer

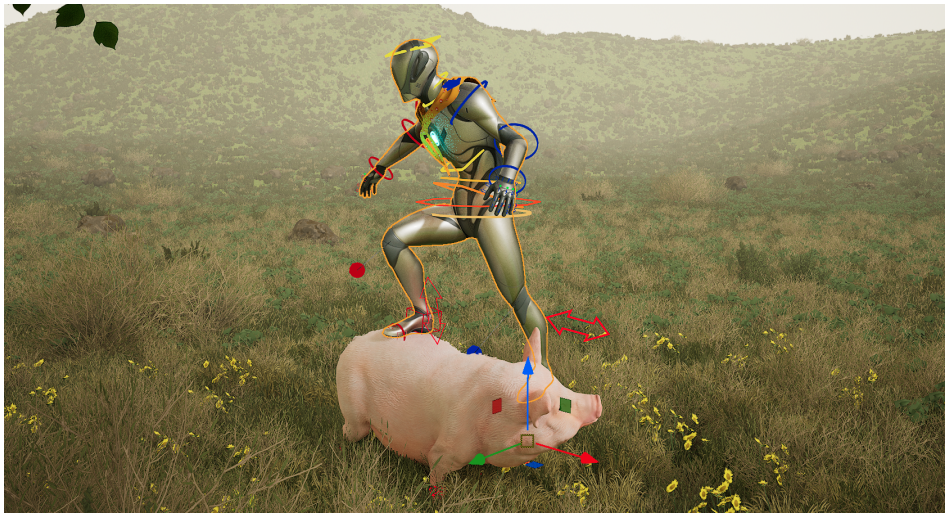
Student ID: 24334440
Video: `https://youtu.be/tKDJ09xxet4`

1 Principles of Animation

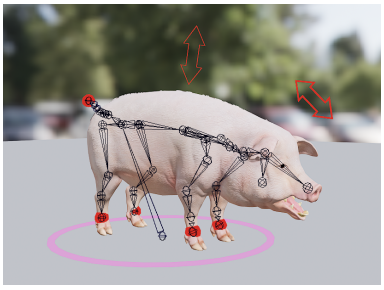
The animated short produced for this assignment involves a mannequin and a pig character. Trying to reach for an apple on a tree, the mannequin relies on the pig's help to finally obtain the desired fruit. To populate the scene, the following assets were used:

- Mannequin character:
  `https://www.fab.com/listings/ad542ee6-34d0-4828-a56c-13846cd649d0`
- Mannequin scarf:
  `https://www.fab.com/listings/93f93282-f61d-4f38-8bc2-83c822e4e2f5`
- Pig caracter:
  `https://www.fab.com/listings/2dd7964c-a601-4264-a53d-465dcae1644c`
- Tree and leaves:
  `https://www.fab.com/listings/a51702ae-6f16-4ae2-83f2-43a9eee40e7b`
- Apple:
  `https://www.fab.com/listings/02d7489f-cd8b-400c-9b1e-efd57cfc24c5`
- Ground foliage and rocks:
  `https://www.fab.com/listings/30e6c98c-6634-4b7e-b142-2c3cdfde6e1e`
- Grass texture:
  `https://www.fab.com/listings/69f0e5f2-0fce-43f5-b737-b3d18de2eb62`

All animations are created within the Unreal Engine 5 Sequencer tool by setting keyframes for the character rigs, props, cameras, and particle system triggers. The interpolation behaviour between two keyframes is then refined using the Sequence Curve Editor. As shown in Figure 1a, the mannequin character comes with a control rig that allows to move individual or groups of bones while setting keyframes in the Sequencer tool. Since the pig model does not include a control rig, a full-body control rig (see Figure 1b) was created to drive the bones of its skeletal mesh using IK targets for each limb, the spine, tail, head, and jawbone.



(a) Mannequin character control rig



(b) Pig character control rig

Figure 1: Control rigs for the characters used in the animation.

1.1 Squash and Stretch

As illustrated in Figure 2, this principle gives both characters a more cartoonish appearance when moving. For instance, the mannequin torso is scaled down slightly when kneeling on the ground and anticipating the jump. After having left the ground, the torso is stretched beyond its original height to emphasize the force of the jump. For the pig, the same logic applies, but here the spine bone is translated up and down instead of scaled, which still creates a squash and stretch impression due to the full-body IK system moving other bones along with the spine bone.

Figure 2: Squash and Stretch of both characters when jumping.

1.2 Anticipation

This effect can be demonstrated best by the sequence that leads up to the pig walking over to the mannequin, which is shown in Figure 3. Here, the character's body first slowly draws back and then thrusts forward more quickly to create the impression of taking a deep breath before starting to walk towards the mannequin. Hence, the interpolation between keyframes 2 and 3 follows a curve that is steeper than that between 1 and 2. In another instance of this principle, the mannequin's arms first move forward then backward before the character steps onto the pig's back in order to get closer to the apple. This is intended to show the mannequin's preparation to build up momentum for the big step that follows.

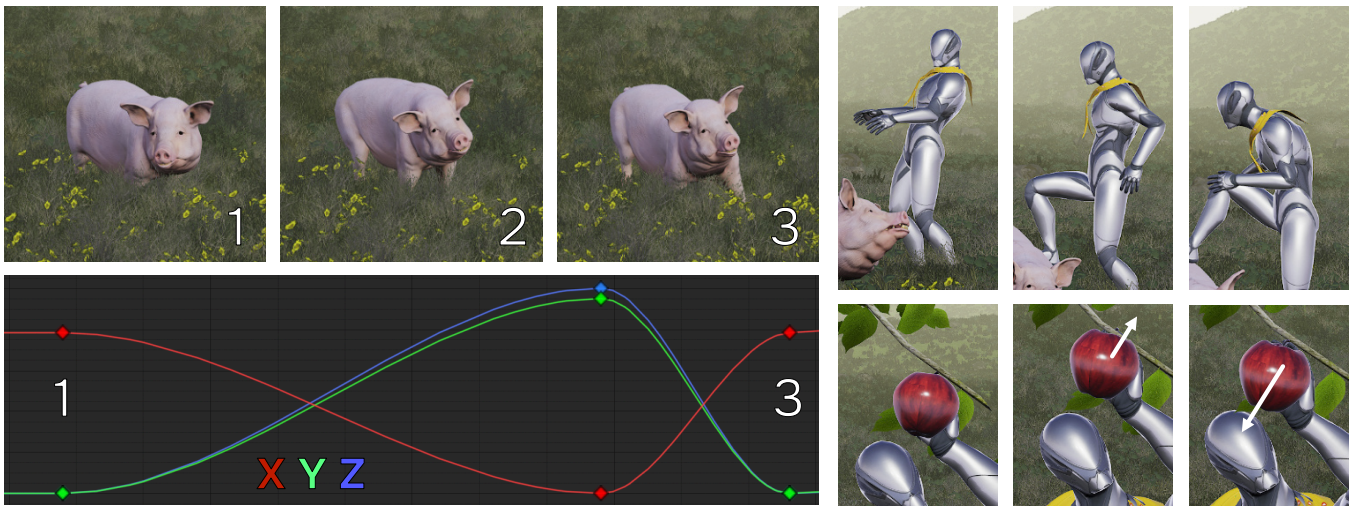


Figure 3: Anticipation before another action sequence begins.

1.3 Follow Through

Most of the effects in this category are achieved through cloth simulations, the implementation of which will be discussed in Section 2. This includes the pig's ears and the scarf worn by the mannequin, yet the pig's tail is driven using regular keyframe animation. The curves shown in Figure 4 illustrate the undulating motion of the pig's tail from side to side while the pig is walking.

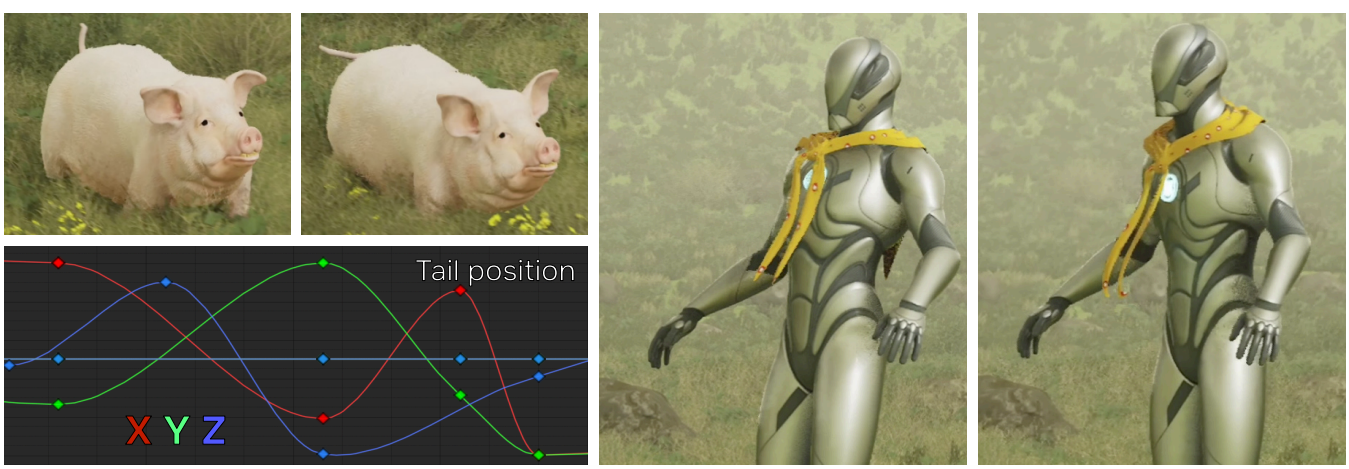


Figure 4: Follow Through using both bones and cloth simulation.

1.4 Ease In and Out

This principle is an integral part of all animated sequences in the short to create smoother transitions between poses and movements but also to mimic the natural tendency of characters to accelerate in the beginning of a movement and decelerate towards the end. In one instance where this effect is very pronounced, the pig's head is animated to shake from side to side in a way that it starts slowly, accelerates, and then slows down again before coming to a stop. This is illustrated in Figure 5, where both the amplitude and frequency of the roll curve of the head slowly taper off towards the end of the sequence.

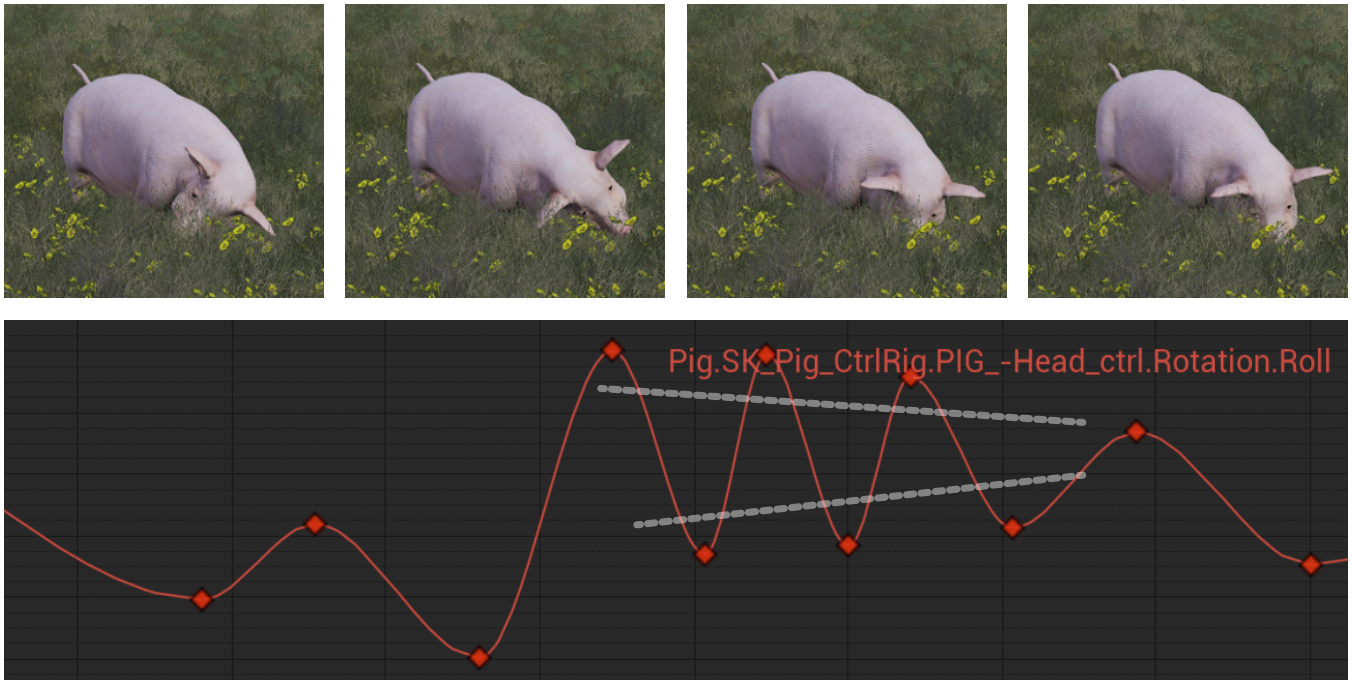


Figure 5: Ease In and Out of the pig shaking its head.

1.5 Exaggeration

Most of the examples shown so far also use exaggeration to further heighten the intensity of the action at hand. One example given in Figure 6 is the pig's arrival at the mannequin's location, with the pig's body pushing forward while its legs still rest in the final pose of the preceding walking cycle. This creates the impression of the pig's body being thrust forward with an intensity that is not physically accurate, yet this overshooting effect heightens the cartoonish nature of the animation. The same principle is used to emphasize the mannequin's surprise upon spotting the apple on the tree above, where the character performs a quick second take at the apple and thus stretches their head in that direction, leading to a short-lived but exaggerated deformation of neck and proportions.

Figure 6: Exaggerated actions, also including a particle system to emphasize the apple being plucked.

2 Physically-Based Animation

To make the animated sequences more engaging, two physically-based methods are implemented in Unreal Engine 5.

2.1 Cloth Simulation

2.1.1 Background

There are various approaches to simulating the behaviour of cloth-like materials in research, with the following three methods being the most seminal in research literature:

1. Finite Element Method (FEM) [1]:
   In this approach, the continuous description or mesh representing the cloth material is discretised into a finite set of surface positions for which we then can define rest pose relationships and constraints informed by real-world material properties such as stiffness. Since multiple constraints between different surface points need to be satisfied at the same time, they can be expressed as a system of differential equations. For each simulation step, a numerical approximation of a solution to such a system is performed in an attempt to satisfy all constraints as faithfully as possible. Consequently, this method is rather computationally expensive, which is why it is used predominantly in offline rendering and simulation scenarios that do not have to run in real-time.

2. Mass-Spring Model [2]:
   Operating directly on mesh representations of objects, this approach assigns mass values to vertices and models each edge as a spring with a given stiffness value. Using these two main properties, the stiffness of the cloth material at hand can be replicated by constraining the distance any simulated vertex may travel from its rest position. In each simulation step, the velocity of any given vertex is determined by means of Newton's laws of motion, a computation that also requires numerical approximation of integrals.

3. Position Based Dynamics (PBD) [3]:
   In contrast to the other methods introduced above, PBD foregoes physically accurate formulations of forces and velocities and instead updates each vertex position directly while satisfying various kinds of constraints using a solving mechanism similar to that of FEM. It is, however computationally less expensive and runs at interactive frame-rates, while it also provides a more straightforward implementation of position-based constraints such as cloth attachments.

Since PBD is used by the Chaos Cloth simulation system in UE5 [4], it is worth investigating this approach in more detail. For a given cloth mesh, different kinds of constraints inspired by real-world materials can be defined by expressing them through equations. For instance, a distance constraint between two adjacent vertex positions v_1 and v_2 can be written as

$$C_d(v_1, v_2) = |v_1 - v_2| - d \tag{1}$$

where d is an arbitrary distance value that v_1 and v_2 need to keep at all times throughout the simulation. This constraint is satisfied if $C_d = 0$ and considered not fully satisfied otherwise.

Next to simple distance constraints that are used to model cloth elasticity, PBD offers a variety of other constraint types such as for bending (constraining the angle between two faces) and for collision detection and depenetration. Given that a high-poly cloth mesh usually contains a great number of edges connecting two vertices, this creates a significant number of constraints that are to be satisfied simultaneously.

The corresponding equations form a system of equations for which PBD approximates a solution using numerical methods such as the iterative Gauss-Seidel or Jacobi solving approaches. Higher iteration counts lead to a more accurate approximation that adhere more strictly to the defined constraints. In the context of cloth simulation this means that stiffer materials require more iterations to create a believable result than more flexible materials would, since all constraints have to be fulfilled more faithfully, requiring a higher solver accuracy. The (approximated) solution to such a system of equations eventually provides new vertex positions that are then applied and then repeatedly updated within each simulation timestep.

2.1.2 Implementation

Figure 7: Selected shots that show the effect of the cloth simulation for both characters.

We use the Chaos Cloth [5] system built into Unreal Engine on two meshes to simulate cloth-like materials:

- Mannequin scarf
  The scarf worn by the mannequin is made of thin cloth designed to also serve as a tissue to wipe clean any apples this fruit-loving character might come across. To underline this aspect, the scarf is given an apple-themed texture, expressing the mannequin's affinity for apples. Attached to the mannequin's neck, its loose ends sway whenever the character moves and keep doing so even after the primary motion has stopped. This adheres to the Follow Through principle mentioned earlier and as such enhances the realism of the mannequin movement.

- Pig ears
  In order to make the pig character appear more enthusiastic and comical, the movement of its ears is animated using Cloth simulation. Serving mostly the principles of Follow Through and Exaggeration, this approach allows the ears to bend, sway, and flap around (see Figure 7) in a way that would be considerably more involved to achieve using keyframe animation. Without this, the ears have a stiff and lifeless appearance (see Figure 5), which would be in stark contrast with the pig's otherwise lively character.

The first step in the Chaos Cloth workflow is to create cloth simulation data for each skeletal mesh to be animated this way. While the cloth mesh for the pig ears is part of the pig mesh itself, the mannequin scarf is represented by a separate mesh that is attached to the mannequin skeletal mesh. After changing

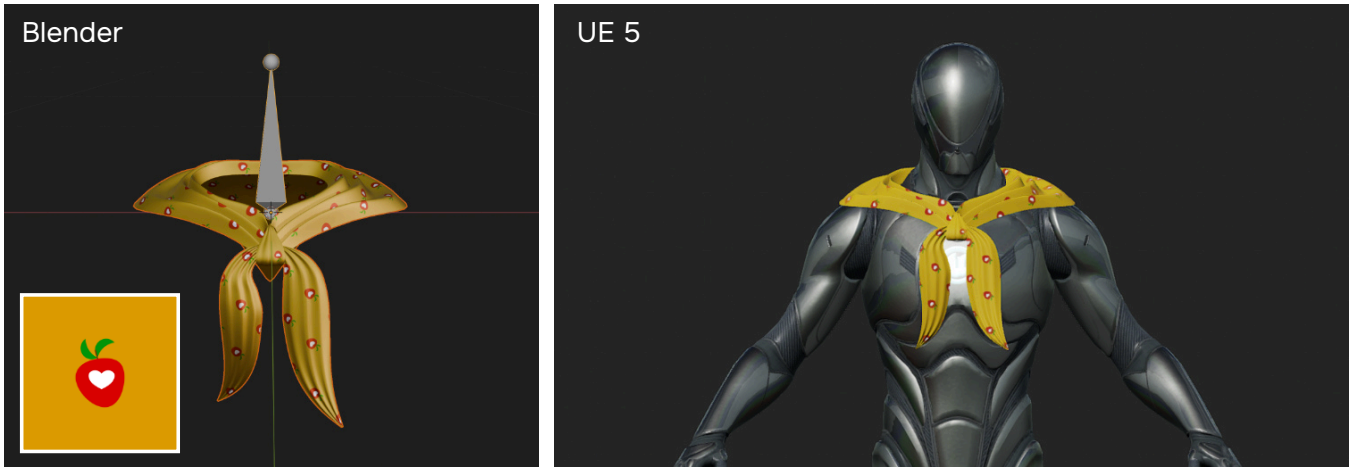


Figure 8: The scarf skeletal mesh in isolation and when attached to the mannequin.

the diffuse texture of the scarf and adding a single root bone as shown in Figure 8, the resulting mesh is imported into the engine and attached to neck bone of the mannequin skeleton by means of a bone socket. For this to work, both the mannequin and the scarf skeletal meshes need to be combined within a character blueprint. In conjunction with the mannequin control rig, the movements of the mannequin bones are then passed on to the scarf mesh.

Using a mesh painting tool, each vertex of the scarf mesh is now assigned a weight value between 0 and 100, which determines the maximum influence the cloth simulation has on the position of that vertex. As can be seen in Figure 9, the neck area uses vertex weights of 0, a way of ensuring that the scarf stays attached to the mannequin's neck at all times. Meanwhile, the ends of the scarf are assigned a higher weight, allowing the corresponding vertices to move away from their rest position more freely.

For the pig ears, we do not need to create a dedicated character blueprint, as they are part of the pig's skeletal mesh already. During weight painting, the ear vertices are assigned a non-zero weight, while all other vertices are set to 0 since only the ears should be affected by the cloth simulation. In order to mimic the elasticity of the cartilage tissue in ears, the stiffness value of the ear cloth is increased compared to the scarf, and bending constraints are enabled to prevent the ear mesh regions from deflating or collapsing. Additionally, the iteration count of the solver is set to 5, allowing these stricter constraints to be enforced more accurately. We also dampen the velocity of all cloth vertices to closely but not fully match the speed of the character animation, an intentional choice to further exaggerate the Follow Through effect.

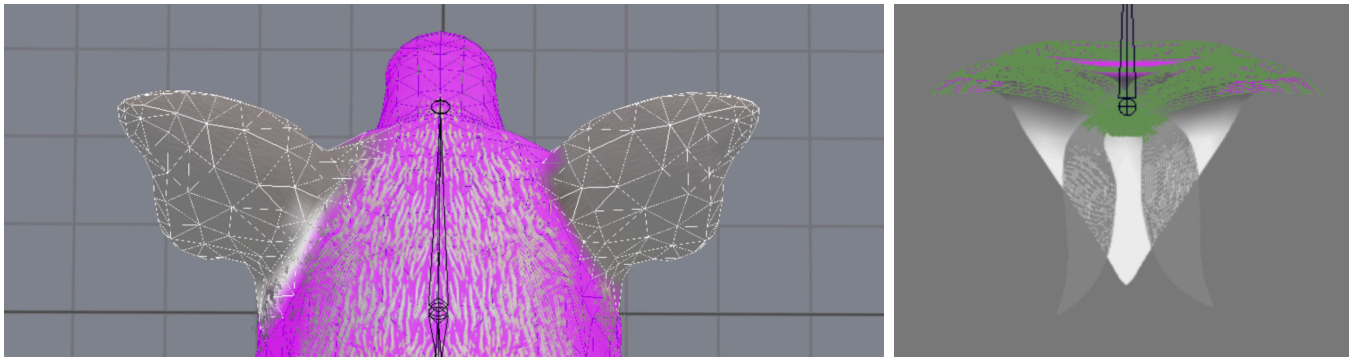


Figure 9: Chaos Cloth weights (pink = 0, white = 100) for the pig ears and the mannequin scarf.

2.2 Particle Simulation

2.2.1 Background

The notion of a particle system was first introduced in [6] which defines a particle as a single, disconnected point in space existing for a certain lifecycle duration and thus clearly separates it from other graphics concepts such as polygon meshes. The authors also outline the sequence of steps required to simulate and render a system of particles, which is still applicable to most particle system methods.

First, instances of any new particles are generated and individual properties such as lifetime, position, and size are initialised. To support a more diverse range of initial particle positions and resulting trajectories, different emitter shapes can be put in place that specify the range of values both the position and emission direction of new particles may fall into. For instance, for an explosion, a sphere emitter would be used, emitting particles equally in all directions.

Meanwhile, pre-existing particles that have reached the end of their lifecycle are removed from the system before the positions of all remaining particles are updated. These updates can be driven in various ways, i.e., they could rely on physics-based principles or more custom logic defined by the architect of the system to achieve certain particle dynamics. After all particles have been moved to their new locations, they are rendered to the screen using different approaches depending on the type of medium the system is designed to represent. Here, the approach presented in [6] renders particles and mesh-based objects separately and combines both results afterwards by means of compositing.

More modern approaches such as [7], [8] use a technique called billboard rendering that draws a 2D texture of variable size at the position of each particle. This enables the replication of larger particles that require more complex surface detail, such as rocks, flames or leaves. When designed to create the illusion of depth, billboard quadrilaterals have to remain perpendicular to the viewing direction at all times. However, in the case of objects that are flat in real-life (such as leaves), billboards are allowed to rotate and move independently of the viewing direction.

2.2.2 Implementation

In this project, the Unreal Engine built-in Niagara System [9] is used to simulate and render particles that represent leaves falling from the apple tree. We use two different emitter systems throughout the scene:

- Ambient leaf tumble
  The first system is used to create the illusion of leaves falling sporadically from the apple tree throughout the duration of the entire sequence. The leaves are spawned at the top of the tree and move slowly downwards, creating a natural effect of leaves tumbling down when the wind is the only force acting on them.

Figure 10: Niagara emitter for the sudden leaf burst and its lifecycle trigger in the sequencer.

- Sudden leaf burst
  We use this system to emit a small set of leaves bursting downward from the position on the tree branch where the apple was attached before being picked. As soon as the mannequin finishes plucking the apple, the system is triggered to emit leaves which then fall to the ground, exaggerating the forceful removal of the apple by the mannequin hand.

Both of these systems use the same leaf texture as the tree mesh to render as particle billboards. The lifetime of the tumbling system begins right at the start of the sequence and lasts until its end, emitting a single leaf approximately every second. It is based on the 'Blowing Particles' preset provided by Niagara which most importantly adds a wind force field that horizontally acts on the particles that, in combination with the regular downward gravity, results in an oblique particle trajectory from the branches to the ground. Since the leaves on the tree do not move, there is only little wind in the scene, which is why we reduce the influence of the wind on the leaves by reducing the range of aerodynamic drag values picked randomly both for translation and rotational movements. This makes the leaves reach the ground quicker while rotating more freely.

The sudden leaf burst system is built upon the 'Directional Burst' preset which uses a cone to randomly assign spawning directions that fall inside this cone to each particle. This cone extends 45° around the main emission direction as configured in the scene in Figure 10, almost perpendicular to the direction of the apple being removed from the branch, which is intended to create a recoil effect. Since we want the burst to appear as powerful as possible, any wind interference on the particles is ignored and only downward gravitational forces are applied. In order to slightly decelerate the particles after being emitted for a more controlled movement towards the ground, drag forces diminishing particle velocities are put in place as well.

References

[1] V. Jagota, A. P. S. Sethi, and K. Kumar, "Finite element method: An overview," *Walailak Journal of Science and Technology (WJST)*, vol. 10, no. 1, pp. 1–8, Jan. 28, 2013, ISSN: 2228-835X. [Online]. Available: https://wjst.wu.ac.th/index.php/wjst/article/view/499.

[2] X. Provot, "Deformation constraints in a mass-spring model to describe rigid cloth behaviour," in *Proceedings of Graphics Interface '95*, ser. GI '95, ISSN: 0713-5424 event-place: Quebec, Quebec, Canada, Toronto, Ontario, Canada: Canadian Human-Computer Communications Society, 1995, pp. 147–154, ISBN: 0-9695338-4-5. [Online]. Available: http://graphicsinterface.org/wp-content/uploads/gi1995-17.pdf.

[3] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2007.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320307000065.

[4] "Panel cloth constraint node reference | tutorial," Epic Games Developer. (Sep. 6, 2023), [Online]. Available: https://dev.epicgames.com/community/learning/tutorials/eKwY/unreal-engine-panel-cloth-constraint-node-reference (visited on 03/31/2025).

[5] "Clothing tool in unreal engine | unreal engine 5.5 documentation | epic developer community," Epic Games Developer. (), [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/clothing-tool-in-unreal-engine (visited on 03/30/2025).

[6] W. T. Reeves, "Particle systems—a technique for modeling a class of fuzzy objects," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 203–220.

[7] T. Umenhoffer, L. Szirmay-Kalos, and G. Szijártó, "Spherical billboards and their application to rendering explosions.," in *Graphics interface*, 2006, pp. 57–63.

[8] "Chapter 21. true impostors," NVIDIA Developer. (), [Online]. Available: https://developer.nvidia.com/gpugems/gpugems3/part-iv-image-effects/chapter-21-true-impostors (visited on 03/31/2025).

[9] "Overview of niagara effects for unreal engine | unreal engine 5.5 documentation | epic developer community," Epic Games Developer. (), [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/overview-of-niagara-effects-for-unreal-engine (visited on 03/30/2025).