5-2024

Animating Reality: Enhancing Immersion through Secondary Animation Rigging Techniques in Video Games

Karim Hudson
karimh@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Part of the Game Design Commons, and the Interactive Arts Commons

Recommended Citation
https://tigerprints.clemson.edu/all_theses/4274

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
ANIMATING REALITY: ENHANCING IMMERSION THROUGH SECONDARY ANIMATION RIGGING TECHNIQUES IN VIDEO GAMES

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

by
Karim Hudson
May 2024

Accepted by:
Rodney Da Costa, Committee Chair
Dr. Eric Patterson, Committee Co-Chair
Dr. Jerry Tessendorf
Abstract

This thesis explores enhancing realism in video games by strategically applying secondary animation rigging techniques to intensify player immersion. The research adopts a novel approach by exploring hybrid methodologies that seamlessly integrate animator control with simulation techniques to imbue characters with dynamic movements and lifelike qualities. Specifically, the study focuses on simulating intricate aspects such as hair, clothing, and accessories, recognizing their potential to elevate character animation beyond static representations. Through practical experimentation, the thesis examines the symbiotic relationship between animator-driven control and simulation-based dynamics. By harnessing the capabilities of modern rigging tools and simulation algorithms, animators are empowered to infuse characters with fluid, responsive movements that resonate authentically with players. Looking forward, this thesis lays the groundwork for further exploration in video game animation. By elucidating the principles and methodologies underpinning hybrid animation control and simulation, it seeks to inspire future research endeavors to push the boundaries of realism and immersion in interactive entertainment.
Acknowledgments

First off, I would like to thank my committee, specifically my committee chair, Rodney Costa, for teaching me all the basics about rigging and being an excellent resource in the development of this project. Thank you, Eric Patterson, for being a fantastic guide in this final chapter of my experience with the program. Lastly, thank you, Jerry Tessendorf, for being willing to hop onto the team mid-project and helping me achieve a final project objective. I want to thank my classmate David Miller, as he was the one who provided me with the model used in this project and helped me art direct the themes and personality behind the character. I also want to thank all my classmates and staff of the program who pushed me to express my creativity in a way that isn’t heavily relied on by other people to make up for any skills that I lack.
# Table of Contents

Title Page ................................................................. i
Abstract ................................................................. ii
Acknowledgments ....................................................... iii
List of Figures .......................................................... v
1 Introduction .......................................................... 1
2 Art of Immersion ...................................................... 3
3 Secondary Movement in Video Games ............................. 6
4 Rigging Basics for Video Games ..................................... 9
   4.1 Skeletal System .................................................. 10
   4.2 Control System .................................................. 11
   4.3 Deformation System ............................................. 12
5 Rigging Methods for Secondary Animations ..................... 13
   5.1 Offline Methods ............................................... 14
   5.2 Real-Time Methods ............................................. 15
6 The Character ......................................................... 18
   6.1 Her Rig ......................................................... 19
7 Experiment ............................................................ 21
   7.1 Rivets with nCloth ............................................. 22
   7.2 Ik Spline with nHair ........................................... 24
   7.3 Rigid Body Animation Nodes ................................. 26
   7.4 Chaos Cloth ................................................... 27
   7.5 Results ......................................................... 29
8 Conclusions and Future Discussion ............................... 31
Appendices .............................................................. 32
   A Scripts and Tools ............................................... 33
   B Unreal Engine Assets and Blueprints .......................... 36
Bibliography ........................................................... 39
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Woman with a Pearl Necklace by Johannes Vermeer[1]</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>The Reverse of a Framed Painting by Cornelius Norbertus Gysbrechts[2]</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>Outfit breakdown for Alloy, the protagonist of Horizon Franchise[3]</td>
<td>8</td>
</tr>
<tr>
<td>4.1</td>
<td>The Character Phase from Paragon by Epic Games Skeletal System.[4]</td>
<td>10</td>
</tr>
<tr>
<td>4.2</td>
<td>Correction Blend shape Example with Elbow.[5]</td>
<td>12</td>
</tr>
<tr>
<td>5.1</td>
<td>Standard rigging pipeline for video games.[6]</td>
<td>14</td>
</tr>
<tr>
<td>5.2</td>
<td>Character Blueprint Skeletal Mesh Hierarchy</td>
<td>16</td>
</tr>
<tr>
<td>6.1</td>
<td>Kahlia’s Turnaround</td>
<td>18</td>
</tr>
<tr>
<td>6.2</td>
<td>Kahlia’s Rig.</td>
<td>20</td>
</tr>
<tr>
<td>7.1</td>
<td>Rivet Constraint with nCloth Workflow Illustration</td>
<td>22</td>
</tr>
<tr>
<td>7.2</td>
<td>Interpolation Visual for Rivet with nCloth in Engine</td>
<td>23</td>
</tr>
<tr>
<td>7.3</td>
<td>IK-Spline with nHair Workflow Illustration</td>
<td>24</td>
</tr>
<tr>
<td>7.4</td>
<td>IK-Spline Collisions for Belt Dynamics</td>
<td>25</td>
</tr>
<tr>
<td>7.5</td>
<td>Clipping Visual for IK spline with nHair in Engine</td>
<td>25</td>
</tr>
<tr>
<td>7.6</td>
<td>Rubber Band Effect in Hair with Rigid Body Dynamics</td>
<td>27</td>
</tr>
<tr>
<td>7.7</td>
<td>Chaos Cloth Workflow Illustration</td>
<td>28</td>
</tr>
<tr>
<td>7.8</td>
<td>Hair Malfunctions with Chaos Cloth</td>
<td>29</td>
</tr>
<tr>
<td>7.9</td>
<td>Cape Stuck under Arm with Hybrid Method</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>UI and Code Snippet from Ik-Spline generator</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>UI and Code Snippet from Ik/Fk Match Tool</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>UI and Code Snippet from Controller Generator Tool</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Kahlia’s Master Animation Blueprint</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>Punch Event Graph. This image depicts the activation of the punch.</td>
<td>37</td>
</tr>
<tr>
<td>9</td>
<td>Kahlia’s Physics Assets</td>
<td>38</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Video games have evolved from simple pixelated experiences to immersive worlds where players can lose themselves for hours. Game developers continually seek new ways to enhance immersion as technology advances, drawing players deeper into the virtual worlds they create. One crucial aspect of immersion lies in the realism of character movement, which can significantly impact how players engage with and perceive the game environment.

My journey into the world of game development began as a spectator, watching “Let’s Plays” to experience games vicariously. While this provided me with a broad understanding of various game genres, I soon realized that my lack of direct interaction with games hindered my ability to accurately assess the feel and mechanics of gameplay. This was evident in my first game production project, I was drawn to every aspect of the development process, from animation to game mechanics. However, I soon encountered a challenge: peers’ feedback was heavily critical of the feel and responsiveness of the gameplay mechanics I was working on. Through this experience, I was told to play video games from various genres to grasp the experience I want to create for others.

This realization prompted me to delve into gaming firsthand, exploring titles outside my usual repertoire. I delved into third-person action-adventure games like Uncharted 4[7], Kena Bridge of Spirits[8], and Horizon Zero Dawn[9]. It was during my playthrough of Horizon Zero Dawn that a revelation occurred. Horizon Zero Dawn is an action-adventure game renowned for its immersive gameplay and captivating world. As I navigated its primal future setting, I found myself deeply immersed, treating each encounter with the game’s robotic adversaries as a life-or-death struggle. This heightened sense of immersion sparked a newfound passion for creating similar experiences for
Driven by this desire, I focused my efforts on mastering rigging techniques to enhance immersion in video games, mainly through implementing advanced secondary movement. Rigging, the process of creating digital skeletons and controls for character animation, plays a pivotal role in bringing characters to life in games. By refining secondary movement—subtle motions such as clothing swaying, hair flowing, or accessories reacting to the environment—developers can elevate the realism and believability of character interactions.

In this paper, I will share the techniques and methodologies I have used to enhance immersion by implementing realistic secondary movement in character rigging. Drawing from my experiences as a player and a developer, I will delve into various rigging methodologies and their ability to express reality artistically. Through rigorous analysis and experimentation, I seek to uncover innovative approaches to rigging that push the boundaries of immersion in gaming.

Ultimately, my goal one day is to contribute to advancing immersive gaming experiences, to transport players to worlds where every movement feels natural, every interaction meaningful, and every moment unforgettable. Leveraging rigging techniques is just the first step.
Chapter 2

Art of Immersion

In contemporary culture, immersion represents a complex interplay between classical artistic influences and modern technological advancements. Originating from the artistic practices of the thirteenth and fourteenth centuries, immersion has evolved into a sophisticated aesthetic approach that continues to shape visual styles in the present day.[10] While my initial understanding of immersion was simplistic – if it resembles reality, it must be real – immersion involves a deeper connection to the depicted world. For instance, Johannes Vermeer’s “Woman with a Pearl Necklace” (see Figure 2.1) creates a sense of reality through the masterful depiction of light, evoking sensations of warmth and familiarity. Similarly, paintings like Cornelis Norbertus Gijsbrechts’ “The Reverse of a Framed Painting” (see Figure 2.2) challenge perceptions of reality, blurring the boundaries between the material and the depicted.

However, immersion extends beyond visual representation; it encompasses an irresistible emotional connection transcending rational distancing. In contemporary society, highly developed illusions permeate our collective consciousness through mediums like computer games and adventure simulations. These forms of media, characterized by their persuasive appeal to affect, offer experiences that blend seamlessly with reality. Immersion, in this sense, allows individuals to lose themselves in these virtual worlds, akin to the intoxicating effects of a drug. This "art of immersion" disrupts traditional notions of mimesis, revealing the constructed nature of our perception and challenging the limits of human understanding.[10]

As technology advances, the novelty of media experiences diminishes over time. What once fascinated audiences, such as Lumière’s cinema or full-color images, now fails to capture attention
Figure 2.1: *Woman with a Pearl Necklace* by Johannes Vermeer[1]

Figure 2.2: *The Reverse of a Framed Painting* by Cornelius Norbertus Gysbrechts[2]
amidst the constant influx of new illusionistic technologies. Even the earliest computer simulations, which once intrigued viewers, now hold interest solely for historians. Successful immersion relies on captivating content, innovative narrative forms, and an emotional appeal accompanying technological advancements to sustain audience engagement. Rather than novelty alone, emotion becomes the cornerstone of immersive experiences, anchoring viewers in the virtual realm.[10]

Immersion, akin to empathy, operates through logic and memory, evoking sensations and emotions similar to real-life experiences. Various forms of artistic media, including animation, excel at depicting immersion through movement. Animation, particularly secondary movement, breathes life into characters and environments, enhancing the viewer’s immersion in the virtual world. Incorporating secondary animation makes characters move realistically, further blurring the line between reality and representation. In this way, immersion becomes a multifaceted experience that relies on emotional connections and innovative techniques to transport viewers into new realms of imagination.
Chapter 3

Secondary Movement in Video Games

In rigging, the primary objective is to equip animators with the essential tools to bring their artistic visions of character movement to life. Achieving this goal requires riggers to have a solid understanding of the principles of animation that guide animators and to recognize the unique challenges inherent in the specific media for which they are rigging characters. This section begins by elucidating the concept of secondary animation by delving into the 12 principles of animation. These principles, pioneered by Disney animators, revolutionized the animation industry by providing a framework for creating compelling and realistic character movements. We explore the origins and significance of these principles, outlining the philosophy that animators follow to achieve impactful and visually appealing animations.

Once we have established a comprehensive understanding of the foundational principles of animation, we can delve deeper into the concept of secondary animation. By examining the principles most closely associated with secondary animation, we can provide a concrete definition and illustrate its role in enhancing the depth and realism of character movements.

Finally, we will explore the significance of secondary animation within the realm of video games. In this digital medium, where player interaction drives character actions, secondary animation plays a crucial role in immersing players in the game world. We will analyze examples of how secondary animation is implemented in video games, highlighting its importance in creating
dynamic, engaging gameplay experiences.

The 12 Principles of Animation, introduced by Disney animators Frank Thomas and Ollie Johnston, have served as a cornerstone for creating captivating character movements in animation. These principles, outlined in "The Illusion of Life," have evolved from basic practices to fundamental rules that underpin the animation industry. They encompass concepts such as squash and stretch, anticipation, staging, straight-ahead action, pose-to-pose, follow-through and overlapping action, slow in and slow out, arcs, secondary action, timing, exaggeration, solid drawing, and appeal.[11]

Secondary movement in animation refers to additional motions that complement and enrich a character’s primary actions, contributing to a more dynamic and believable performance. It is closely associated with principles like follow-through and overlapping action, combined with secondary action.

Follow-through and overlapping action involve the continuation of movement beyond the primary action’s cessation and the synchronization of multiple movements within a character. This principle is evident in various aspects of character design, such as long ears, tails, and oversized coats, which continue to sway after the main body has halted its movement.[11]

The body’s movement is characterized by a sequential motion, with different parts moving in coordination. For instance, the hip initiates movement, followed by the upper body, creating a fluid and interconnected motion. Similarly, when swinging an arm, the movement begins at the shoulder, progresses through the elbow, and culminates at the wrist, resulting in a cohesive and realistic motion. Additionally, trailing elements like fat or loose flesh exhibit delayed movement compared to the rest of the body, contributing to a sense of naturalness and weight in the animation.

Secondary action incorporates additional movements that complement and enrich the primary action, enhancing the performance’s overall dynamism and believability. These supplementary motions, ranging from subtle gestures like wiping a tear to more pronounced movements like shaking one’s head or adjusting glasses, add depth and complexity to the character’s performance.[11]

By incorporating secondary actions, animators can create more nuanced and realistic animations that resonate with the audience on a deeper level. These additional movements help to flesh out the character’s personality and emotions, making the performance more engaging and immersive. Ultimately, secondary actions are crucial in bringing characters to life and enriching the storytelling experience.

In essence, secondary movement adds depth and realism to character animation by sim-
ulating the natural interactions between different elements within a scene. It enriches the visual storytelling and helps establish character and personality traits through subtle yet impactful movements.

In the realm of video games, secondary animation plays a crucial role in enhancing player immersion and overall enjoyment. Over the past two decades, advancements in gaming engines have led to stunning graphics and highly immersive environments. Part of this appeal stems from integrating secondary animation techniques into games.

Animating secondary movement in video games typically involves either hand animating a set amount of controls or utilizing simulation techniques. The evolution of simulations has dramatically increased the realism achievable in games, allowing for more dynamic character animations. Take, for example, the protagonist from The Horizon franchise, particularly her model in Horizon Zero Dawn\(^9\), adorned with armor and coats essential for survival in the game’s wilderness. Without dynamic movement in her equipment, the character’s animation could appear stiff and unnatural, potentially detracting from the player’s overall experience. (see Figure 3.1)

![Figure 3.1: Outfit breakdown for Alloy, the protagonist of Horizon Franchise\(^3\)](image)
Chapter 4

Rigging Basics for Video Games

Rigging is a critical component in 3D animation, playing a pivotal role in streamlining and enhancing the animation process. Animating a 3D digital character would be arduous and time-consuming without a well-crafted rig. However, creating a high-quality rig requires more than technical proficiency; it requires a comprehensive understanding of anatomy, kinematics, and the intricacies of artist-rig interaction. Additionally, riggers often employ a blend of technical expertise and artistic sensibility akin to that of a sculptor to achieve optimal results. A thorough comprehension of rigging entails familiarity with the three fundamental systems inherent in rigging a digital character: the skeletal, controller, and deformation systems. These systems work in tandem to imbue the digital character with lifelike movement and expression. The skeletal system lays the structural foundation, mimicking the underlying anatomy and facilitating movement by defining the character’s joint hierarchy. Meanwhile, the controller system provides animators with intuitive, straightforward controls to manipulate the character’s movements efficiently. Finally, the deformation system shapes the character’s mesh in response to the movement of the underlying skeletal structure, ensuring smooth and realistic deformations during animation. By understanding and mastering these central systems, riggers can create rigs that empower animators to bring digital characters to life with precision and fluidity.
4.1 Skeletal System

Establishing a skeletal system marks the foundational and paramount stage in the rigging process. Mastery of this step necessitates a profound understanding of anatomy, range of motion, and kinematics. Across various software platforms, this system relies on positioning points on a character, typically represented as joints or bones. It involves organizing these bones in a hierarchical structure, ensuring that manipulations of a parent bone propagate to its child’s bones, adhering to the inherent behavior of hierarchical structures.[13] Primarily, these joints emulate the skeletal framework of a creature, but they can also be extended to manipulate clothing, accessories, hair, and other elements. In some cases, additional joints may be required to facilitate deformation in specific mesh areas. Once the joints are accurately positioned, the skinning process ensues, wherein joints are assigned a percentage of influence over each vertex on a 3D model. A vertex with 100 percent skinning to a joint implies that the joint exerts complete control over the vertex’s position and behavior, effectively dictating its transformation information.[13] (see Figure 4.1)

![Figure 4.1: The Character Phase from Paragon by Epic Games Skeletal System.][4]

Furthermore, this influence can be distributed among multiple joints, allowing for smoother deformations as the vertex position interpolates between the transformations of the associated
joints. Extensive time and effort are invested in this phase to ensure that subsequent steps are not compromised if issues arise with the skeleton. This meticulous procedure holds particular significance in game engines, as the skeletal system is one of the few elements capable of seamless transfer between digital content creation (DCC) tools such as Maya and Unreal Engine.

4.2 Control System

Much like the cross-brace affixed to a puppet for puppeteering, rigging streamlines the manipulation of a skeletal system by employing a control rig. This user-friendly setup enables animators to swiftly animate the skeleton, facilitating the execution of simple movements with greater efficiency. Fundamental features of control rigs include inverse and forward kinematics switching and animation space switching controls.

An IK/FK switch in 3D rigging provides animators with the flexibility to choose between Inverse Kinematics (IK) and Forward Kinematics (FK) for specific movements. FK involves rotating joints from the root down to the end of the joint chain to create movement. This type of movement resembles how an action figure’s limbs are posed. For instance, raising an arm involves rotating the shoulder, bending the elbow, and twisting the wrist. On the other hand, IK creates movement controlled by translating an IK handle, which is typically attached to the chain’s last joint. Many animators prefer IK because of its ease of posing. However, IK interpolation often results in primarily linear movements, making it challenging to achieve smooth arcing motions, such as swinging an arm in an arc. Switching between IK and FK systems allows animators to overcome this limitation. Transitioning to an FK system enables them to create more natural and fluid movements, particularly when it comes to arcs and curves. This flexibility is precious in gaming, where achieving optimal results may require FK movement for specific animations.

Animation space switching allows animators to isolate the movement of a character’s body part, ensuring that it follows a specific body part or remains static within its world space. This functionality proves invaluable in scenarios such as placing hands on hips, where the hands should mirror the movement of the hip controller. Similarly, it is commonly utilized to isolate head movement, allowing animators to adjust the rest of the body without altering the head’s position.

Control systems can also drive helper joints, which support specific movements to maintain desired deformation. These helper joints are typically configured to be driven by node-based
algorithms, automating their movement based on mathematical calculations. As a result, when animators manipulate a controller, the joints respond accordingly, ensuring seamless integration between the control rig and the skeletal system.

4.3 Deformation System

When transferring information between different Digital Content Creation (DCC) tools for gaming engines, only skeletal and blend shape data can seamlessly be transferred due to limitations. Deformation systems for gaming are typically straightforward, primarily focusing on joint-based or blend shapes. Blend shapes are a way of reinforcing specific deformations to minimize the need for helper joints. Blend shapes, or morph targets or shape keys, are a collection of sculpted poses sharing the same topology as the original mesh. These poses morph into each other to create more defined body deformations. As seen in Figure 4.1, consider a character’s arm with simple topology around the elbow. As the arm bends at the elbow, the geometry may start to intersect or clip into itself. A blend shape is implemented to address this, sculpting the topology around the clipping area to simulate how the skin would realistically deform if driven by actual muscles. This process is commonly used in facial rigs, where blend shapes can control nearly every subtle movement within the face, enhancing expressiveness and realism in character animations. (see Figure 4.2)

![Figure 4.2: Correction Blend shape Example with Elbow][5]
Chapter 5

Rigging Methods for Secondary Animations

In gaming, animating characters typically involves applying handmade animations or motion capture data to the joint skeleton. However, when characters feature clothing, accessories, and hair, additional joints branching off the main skeleton are necessary to control these elements. Unfortunately, no method currently exists for capturing motion capture data specifically for these auxiliary joints. As a result, animators often turn to simulations to achieve realistic movement for clothing, accessories, and hair, which can then be transferred to the skeletal systems for seamless integration into the game engine.[15]

In this section, various methods within Maya and Unreal Engine are explored for simulating joints to enhance the realism of their movement. In a standard pipeline for game development, the rigging process unfolds in two distinct stages. (see Figure 5.1)

Firstly, the offline stage encompasses the creation of the initial skeletal and control rig within the animator’s preferred Digital Content Creation (DCC) software for animation. This rig is the foundation for animators to bring characters to life through intricate movements and gestures. Once the animation is finalized, the data is baked onto the skeletal mesh and exported to the game engine.

Subsequently, in the real-time stage within the Engine, the animation interacts with a dynamic rig setup utilizing additive animation techniques. This setup enables real-time simulation or augmentation of the baked animation information, adding secondary elements with lifelike dynamics.
By leveraging these techniques, animators can enrich the animation experience, infusing characters with fluid and natural movements that enhance visual fidelity and immersion.

![Diagram of standard rigging pipeline for video games.](image)

**Figure 5.1: Standard rigging pipeline for video games.**[6]

### 5.1 Offline Methods

In real-time productions, there are instances where there may be more cost-effective options than simulating dynamics in real-time due to limitations in control and direction. To address this challenge, technical animators or riggers employ methods to bring simulations from separate software and convert them for integration into the game engine. One such methodology is known as octopus skeletons.

The octopus skeleton process involves scattering joints heavily across the surface of an item being simulated while anchoring them to a singular point. These joints are then constrained in some manner to the simulated surface, ensuring that the simulation results align with the desired outcome. Subsequently, the positions of these joints are re-baked. With the joints now animated, they can drive the original mesh, and this data can be seamlessly transferred into the game engine. This section will explore two octopus skeleton methods for transferring simulation data onto joints, facilitating their integration into the character skeleton and the engine.[16]

The first method involves rigging dynamics through a simulation placed on a curve. This technique begins with an IK spline setup, a unique inverse kinematic system differing from traditional rotation-plane or single-chain setups. It utilizes a curve spanning the joint chain, and manipulation of the curve’s control vertexes (CVs) dictates the movement of the joints to match the curve’s shape.[17]
In Maya, you can make any curve dynamic by utilizing nHair. Maya’s nHair is primarily designed for creating natural movement and collisions for hair. Still, it also has another helpful application in creating dynamic curve effects such as ropes, chains, cables, wires, suspension bridges, sea creatures, and more. Creating this dynamic connection is straightforward: select the nHair menu and the Make Selected Curve Dynamic tool.[18]

Once created, the curve can be controlled by adjusting the properties and settings to achieve the desired outcome. After constructing this setup, the next step is to generate primitive collisions to represent specific parts of the character that these curves would collide into.

Another method for dynamically rigging cloth, hair, and accessories involves using rivet constraints and Maya’s nCloth. Rivet constraints enable the attachment of an object to specific points on a deforming mesh. There are two primary approaches to this constraint: one attaches itself to the location of one or several vertices, utilizing their normals to calculate rotation, while the other correlates the 3D world position with the UV space position of the geometry.[13]

Maya’s nCloth is a fast and stable dynamic cloth solution that uses a system of linked particles to simulate various dynamic polygon surfaces. It is commonly used for fabric clothing, inflated balloons, shattering surfaces, and deformable objects. According to Maya 2023 documentation, the basic setup for using nCloth for characters’ clothing involves making the clothing nCloth, making the character into a nCloth passive collision object, constraining the cloth to itself and the character, and adjusting the clothing parameters to achieve the desired appearance and behavior.[18]

This method, inspired by a SIGGRAPH paper by Blizzard Entertainment, leverages nCloth’s capabilities to drive joints affiliated with the character’s joint skeleton, ensuring seamless transfer of all animation data to the gaming engine. Additionally, skilled riggers can incorporate additional controls, allowing animators to refine and augment the simulation.

5.2 Real-Time Methods

Unreal Engine offers various options for generating dynamics within the Engine at runtime, with the setup of these dynamics revolving around physics assets. These assets comprise collision volumes and constraints, mimicking behaviors such as hinges, ball joints, or piston-like movements while holding physical properties assigned to the joints in the skeletal system.[13]

Upon import into the Engine, these physical assets are typically transferred to a singular
skeletal mesh, the Unreal equivalent of the skeletal system. Specific systems within the skeleton can be isolated and exported individually to streamline this process. This separation simplifies the setup by reducing the number of collision volumes and constraints needed. Assembly is then facilitated within the character’s actor blueprint (see Figure 5.2), allowing all the different physics systems to function seamlessly on the singular character.

Moreover, these physics assets can be dynamically controlled in real time through animation blueprints using a feature known as the rigid body animation node (RBAN). These RBANs are animation nodes attached to specific animation stages or overlaid on top of the animation blueprint’s animation system. This powerful tool controls whether dynamic secondary movement is added to a particular animation, enhancing realism and immersion.

Another approach to leverage Unreal Engine’s physics assets is by setting up collision volumes for Chaos Cloth, their in-house cloth simulation system. Cloth simulation plays a vital role in gaming production, adding a layer of realism to characters’ clothing and accessories. Different fabrics exhibit unique characteristics influencing their movement, and adjusting parameters affecting these movements is crucial.[16]

The basic setup for utilizing Chaos Cloth involves assigning the fabric to a skeletal mesh, with simulation controlled by material attributes. Painting tools can be used to determine the extent
of simulation versus joint-driven movement on the cloth surface. Additionally, various configurations in attributes allow adjustments to match desired outcomes. Collision setup by the physical asset enables interaction with the character, while collisions can also be configured to interact with the environment. Furthermore, wind effects in the world can influence cloth movement, adding further depth and realism to dynamic animations.[13]
Chapter 6

The Character

The character selected for this project, Kahlia, conceptualized by classmate David Miller, embodies the persona of a magical archaeologist adorned with a cursed golem-like gauntlet infused with mystical powers. Set in a fantasy post-apocalyptic world where magic and its artifacts are scarce commodities controlled by the privileged few, Kahlia’s design epitomizes the struggle for power and the allure of the arcane. From the meticulously crafted potions adorning her bandelier and belt to the hooded cape and skull mask fashioned from magical creature remains, every aspect of her design exudes adventure and fascination with the mystical. (see Figure 6.1)

Figure 6.1: Kahlia’s Turnaround

Kahlia’s character lends itself seamlessly to a 3D adventure action game, where her rig demands an extensive range of motion and meticulous control to ensure fidelity in the movement of her accessories. Incorporating realism into her design further enriches her character, elevating the overall visual appeal.
6.1 Her Rig

Inspired by Kahlia’s design, the project delves into the intricacies of setting up a control rig that empowers animators to animate every facet of her character, from her flowing hair to her dynamic belts and cape. This endeavor serves as the foundation for the creation of this paper, driven by the need to explore the optimal methods for rigging characters in a gaming environment.

Before integrating dynamic systems, Kahlia’s base rig was meticulously crafted to prioritize body mechanics and pantomimic exercise. A main body skeleton consisting of 67 joints that are primarily utilized for the basic full-body movements. Additional joints are allocated to support the twisting deformation of the wrist, bicep, and neck area. Thirty-nine joints serve as a placeholder for future implementation of a facial rig, supporting everything from eyelids, eyebrows, cheeks, and mouth. The remaining 106 joints are then allocated towards supporting the secondary animation systems, giving the cape, chains, belts, bandoliers, and hair a high level of control over its deformations. There are also some additional support joints for her props, such as her shoulder guard, which has some ancillary joints that are procedurally moving based on the rotation of the shoulder joint so that the shoulder guard doesn’t interpenetrate her shoulder. To help identify each secondary system joint chain, I also parented all the joints affiliated with that prop to a root joint similar to an octopus skeleton. Her control system features a simple Ik/Fk switching mechanism for her arms, legs, and spine, complemented by straightforward world-switching controls for her Ik hands and head. This setup facilitates streamlined retargeting of animations within Maya using Human IK, Maya’s built-in re-targeter, enabling swift animation production with minimal manual adjustments. For example, the Ik/Fk switching combined with a script can convert the retargeting animation data placed on the FK rig to the Ik system to correct foot and hand placement during specific animations.(see Figure 6.2 and Appendix A for more detail)
Figure 6.2: Kahlia’s Rig.
Chapter 7

Experiment

Experimentation is crucial to understanding the efficacy of dynamic rigging systems in video game character animation. In this section, I will outline the experimentation process conducted for this paper, detailing the construction and application of each dynamic rigging system and evaluating their respective pros and cons.

To evaluate the performance of each dynamic rigging system, a series of test animations were developed, culminating in a dynamic charging punch sequence. This sequence showcased Kahlia slowly winding up her arm, twisting her body for maximum power, and delivering a powerful haymaker punch. The focus was on the punch executed at maximum charge, as it effectively demonstrated the capabilities of the dynamic rigging systems in conveying force and impact.

All the secondary animation rigging methods mentioned earlier were employed to simulate the character’s cape, hair, bandolier, and belts to provide realistic secondary movement that adds to the immersion of the gameplay. These methods were implemented separately from each other to determine the limitations of each system. Due to their simplistic skeletal structure, her chains, goggles, and potions will be handled separately from this evaluation.

Each dynamic rigging system was meticulously constructed and integrated into Kahlia’s rig, allowing for a comparative analysis of their performance. Pros and cons were identified for each system, considering factors such as ease of manipulation, realism of movement, computational efficiency, and integration with existing animation pipelines.

Ultimately, the experimentation aimed to identify the method that yielded optimal performance regarding realism, flexibility, and ease of use. Additionally, opportunities for amalgamating
multiple approaches were explored to achieve a more desirable outcome, leveraging the strengths of each system to enhance Kahlia’s animations and immerse players in her dynamic world.

7.1 Rivets with nCloth

Using rivet constraints with nCloth to create secondary movement in video games offers a promising approach to achieving realistic cloth and hair animations. However, it comes with challenges and limitations that need to be addressed for effective integration into gaming engines.

My approach to constructing the system involves having three pieces of geometry, the first being the modeled original mesh. The second would be a proxy mesh, a plane matching the silhouette of the original mesh. The third is a duplicate of that proxy mesh, which will be the simulated mesh driven by the nCloth. Once those three pieces of geometry are in place, it’s time to make them communicate. The connection between the proxy mesh and the simulated mesh is connected via a blend shape, allowing the proxy mesh to match the deformations of the simulated mesh. Once established, rivet constrain the joints to the closest vertices of the proxy mesh. As long as the joints are already skinned to the original mesh, this setup allows the original mesh to be driven by the simulated mesh (see Figure 7.1), providing realistic movement. To add additional control, instead

Figure 7.1: Rivet Constraint with nCloth Workflow Illustration
of rivet constraining to the joint directly, the constraint can be connected to an offset transform group parented to a controller that drives the joint. This allows animators to animate over the simulation by manipulating that controller. This combination of simulation-driven movement and manual control offers flexibility in directing the animation’s realism.

However, this method has limitations, particularly when integrated into gaming engines. Loopable animations are challenging due to extreme joint movement during the simulation. The joint movement is so extreme that animation layering wasn’t enough to create smooth loops, which are animations ideal for making many animation states in gaming engines. Additionally, transitioning between animation states can be problematic. In Figure 7.2, these images show the animation trying to interpolate from the end of the punch animation back to the idle state in which the joints take an extreme path to return to their resting state. Addressing these limitations may require creating in-between animations or performing intense animation blending before porting it to the game engine to ensure seamless transitions.

![Figure 7.2: Interpolation Visual for Rivet with nCloth in Engine](image)

Despite its limitations, the rivet constraints with nCloth method offers a promising approach to achieving realistic secondary movement in video games, particularly for stylized aesthetics where complete control over cloth and hair movement is desired. With further refinement and integration techniques, this method has the potential to enhance character animations and immerse players in dynamic gaming experiences.
7.2 Ik Spline with nHair

Using IK spline with nHair presents another method for creating secondary movement in video games, offering user-friendly controls for artistically directing dynamic movement. However, like the previous method, it also comes with limitations and restrictions when imported into gaming engines.

The approach involves creating IK spline systems for the joints manipulating the geometry to be simulated. This is achieved by creating a curve with a curve vertex (CV) matching the location of each joint within the chain and generating an IK handle. The curve can then be connected to the nHair dynamic system, with the dynamic output curve serving as a blend shape of the original curve. Manipulating the CVs on the curve drives the joint movement, simulating the dynamics of the object. (see Figure 7.3) For specific articles of clothing, such as belts and bandoliers, dynamic constraints are maintained at both the start and end of the curve. In contrast, for objects like hair and cape, only the beginnings of each curve are constrained, allowing the ends to free fall. Adjustments to nHair attributes and collision settings may be necessary to fine-tune the simulation.

However, proper skeletal preparation is crucial, as readjustments to the rig and exported animations may be required. One significant limitation of this method is that the IK spline does not translate joints, making it challenging to mimic stretching without scaling each joint, resulting in irregular deformations. The overall output may resemble an umbrella flipped inside out or webbing between wet feet, making it more suitable for stylistic simulations rather than realism.

Regarding the punching animation, issues arise with the hair getting stuck in collision pock-
ets, the cape moving like inside-out umbrellas, and the belts and bandolier evaluating with little problems. The limitation of the IK spline not allowing joints to expand and contract may affect the usability of belts if the legs are split apart. Additionally, joints at the start of the chain on the bandolier seem to get forced out of alignment due to interpenetrating collisions. (see Figures 7.4 and 7.5)

Figure 7.4: IK-Spline Collisions for Belt Dynamics

Figure 7.5: Clipping Visual for IK spline with nHair in Engine

In conclusion, while IK spline with nHair is suitable for simulations like tails or long pony-
tails, it may not be ideal for achieving realism in clothing simulations. It should be reserved for specific applications where the limitations are acceptable and can complement the overall aesthetic of the character animation.

### 7.3 Rigid Body Animation Nodes

As discussed in the paper, integrating any secondary movement systems into Unreal Engine requires creating a physical asset and establishing the collisions for interaction with these systems. The rigid body animation method takes this further by treating each piece of clothing as a separate physics asset, aiding in the organization, optimization, and independent editing of rigid body nodes for each system.[19](See AppendixB for more details)

Setting up the rigid bodies involves creating collision body shapes for each joint associated with the clothing piece assigned to that physics asset. Then, determine whether the rigid body physics type is simulated or kinematic. Kinematic bodies follow the parent joint, while simulated bodies adhere to the dynamics of the simulation. After setup, constraints are assigned between collision bodies, with angular limits adjusted to control rotation and prevent unrealistic movement or clipping. Finally, fine-tuning physics settings for collisions and rigid bodies completes the process.

Overall, the performance of rigid bodies was satisfactory, with real-time evaluation resulting in decent secondary movement and follow-through during animations. However, there were some challenges, including difficulty adjusting settings for optimal physics outcomes and needing modifications to fix collision issues on some assets, particularly the cape. Due to the cape’s awkward joint positioning, initial collision locations may not facilitate realistic cloth-like movement.

In the case of the punch animation, the bandolier performed well, while issues with the belts arose from managing collisions and determining the appropriate physics type for joints. Hair chains, except for the massive ponytail, were evaluated adequately. Still, issues emerged due to the number of rigid bodies in a single chain, leading to jitting problems, as Unreal Engine’s documentation stated. Another reason the hair may not perform ideally is that when an animation is baked onto the joints with the rigid body animation node, it overlays the dynamics on top of that baked animation data. When creating isolated tests for both the rigid body system and the Chaos cloth system, all the joints affiliated with the geometry of interest had baked joint information that was completely static throughout the animations. Most likely because of the baked-in penetrations
happening because these joints are static. It’s causing the rigid body system to interact with its collisions very awkwardly, leading to constant slingshotting of the large ponytail when momentum is going forward. (see Figure 7.6) Despite these challenges, the rigid body method provides proper secondary motion, follow-through, and overlapping action within the gaming engine for various animations. Some refinement may be necessary, including implementing toggles to adjust how the skeleton responds to baked animation data or simulations on specific animations. Nonetheless, with further fine-tuning, the character’s dynamics can likely be achieved entirely with the rigid body setup.

7.4 Chaos Cloth

The other real-time method for creating secondary movement in game engines, specifically Unreal Engine, is its built-in cloth simulation, Chaos Cloth. Surprisingly, this was the best-performing system, with only a few minor issues.

The setup involves creating a simplified one-sided proxy mesh of the clothing to be simulated. Then, a shader is assigned to each proxy mesh to bring it into the engine, as Unreal uses shaders to determine which objects have cloth simulation. Once the cloth is imported as a separate skeletal mesh, the Chaos Cloth can be applied to the proxy mesh to drive the rendered cloth attached to the character’s skeleton. Cloth painting mode is then activated to paint the weight of influence that the
simulation has over the geometry. From there, configurations for the cloth can be adjusted based on trial and error. (see Figure 7.7)

![Chaos Cloth Workflow Illustration](image)

Figure 7.7: Chaos Cloth Workflow Illustration

The results of testing Chaos Cloth across all simulated meshes were astonishing. Regarding the punch animation, Chaos Cloth performed the best overall, with the cape and belts showing optimal performance. Surprisingly, the hair also fared decently, although it appeared loose and flimsy due to its freedom of motion. (see Figure 7.8) The bandolier, however, exhibited the lowest performance, not because the bandolier didn’t deform appropriately based on the simulation settings. It’s more so because of the bandolier’s interaction with the rest of the in-game model. Since the best way to put dynamics to her potions was using rigid body constraints, the bandolier, under the effects of chaos cloth, constantly clips with the potion’s rigid bodies because the joints that were attached to the bandolier’s rendered mesh, which drives the potions joints are not being animated because chaos cloth is only affecting the geometry itself. I am very uncertain that adjusting the collisions for the potions would do much to improve this situation.

In summary, Chaos Cloth would be my preferred method for creating secondary movement in Unreal Engine, particularly for cloth simulation. Once you understand how to configure the cloth using Unreal’s controls, achieving the desired result becomes straightforward.
7.5 Results

Now that we have evaluated each system independently, I have found a balanced hybrid approach to maximize the performance of secondary movement for my character. By combining the different methods, I have achieved the best of each world, addressing specific needs and optimizing the performance of her secondary movement.

The final breakdown involves using baked animations with rivet constraints and nCloth for non-looping animations, while the engine’s Chaos Cloth is implemented for the cape and belts. The hair and bandolier are controlled by rigid body systems, with guided animation underneath the hair to prevent bugs during extreme movements. The bandolier now moves joints correctly, preventing clipping issues with potions.

This combination has yielded the best results for the punch animation, although there’s still room for improvement. Additional techniques, such as wind effects or invisible colliders, could address minor issues like the cape caught under the arm while transitioning from the end of the punch to the idle. (see Figure 7.9) Alternatively, adjusting the base animation for smoother transitions may help resolve such issues. Overall, this hybrid approach provides a solid foundation for further refining the dynamics of the animation.
Figure 7.9: Cape Stuck under Arm with Hybrid Method
Chapter 8

Conclusions and Future Discussion

Throughout this paper, I have detailed the techniques and methodologies I employed to enhance immersion by integrating realistic secondary movement into my character’s rig. While my final results relied heavily on the capabilities of Unreal Engine’s dynamic systems, I recognize the potential for further development and refinement of the offline rig system to surpass the real-time system in terms of ease of control and level of detail. In future iterations of this project, I intend to explore these possibilities on a larger scale, dedicating more time to refining the offline rig’s controls for optimal performance.

These techniques have deepened my understanding of implementing secondary animation within gaming engines, bringing me closer to achieving immersive, dynamic gameplay through fluid animation. As I continue to learn and evolve within the industry, I am driven by the prospect of realizing this goal with great success and inspiring others to pursue their creative visions. Immersive gameplay has been instrumental in shaping my journey thus far and contributing to the advancement of immersion, no matter how small the contribution, holds profound significance for me as both an artist and a creative. My ultimate aspiration is to make the visions of fellow artists and creatives a tangible reality for others to experience and enjoy.
Appendices
Appendix A  Scripts and Tools

![Image of UI](image1.png)

a User Interface

![Image of Code Snippet](image2.png)

b Code Snippet

Figure 1: UI and Code Snippet fromIk-Spline generator

The script generated the Ik-Spline system of a joint chain in one of three ways: selecting each joint in the joint chain, selecting the base of a joint chain, or choosing the root of a series of joint chains.
This script was utilized to convert retargeted animation data from the FK controls onto the Ik controls to correct hand and foot placements for the character animations.

Figure 3: UI and Code Snippet from Ik/Fk Match Tool
This simple script was implemented to create controllers faster. Setting up controllers for basic relationships like the clavicles and head controllers.
Figure 7: Kahlia’s Master Animation Blueprint
This is the animation graph for Kahlia’s movement for this project. I utilize pose caches to create nested State machines that blend based on bool conditioning. For example, the climbing movement state machine is set up with a “Blend Pose by bool” node with the basic movement state machines, which blends between both state machines based on the “isClimbing” flag.
Figure 8: Punch Event Graph. This image depicts the activation of the punch. This is the blueprint logic for the punch mechanic. Press the E key, trigger the punching event, check if the player is on the ground, flag the “is charging” state, plus block movement input from the user controller. On release, check if the player is on the ground, trigger the knockback event and then, based on an int switch that is being incremented at three stages within the charging animation itself, apply a delay based on the duration of the punch, enable the character’s movement, and reset dynamics.
This is a collection of all the physics assets utilized in Unreal Engine regarding constructing constraints and collisions for the RBAN system and the Chaos Cloth system.
Bibliography


