Animations Driving the Game Simulation

Jared Goronkin
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
jgoronkin@gmail.com

Abstract – This paper analyzes animation driven game simulations as present in Dark Souls, where gameplay parameters are modified directly through animation data via metachannels. Unlike traditional models that decouple animation and logic through state machines or frame data, this approach allows animations to drive the gameplay simulation, consolidating the design pipeline and enhancing the artist's expressive capabilities. Using tools like DSAnimStudio to inspect .TAE files, the paper explores the affordances and constraints of this design pipeline compared to other traditional models for animation integration.

1. Introduction

A key feature of many Action RPGs is that they are comprised of a collage of disparate mechanics. Dark Souls is a game that pushes back against this quality. In Dark Souls, artistic and mechanical unity is prioritized above all else. There is a glue, a singular subtle mechanic, that strings almost everything else together. Not only does it impact the end-user experience, but it's a simple and elegant piece of technology that has profound implications in terms of the game development process as well. Although there is no singular term for it, this mechanic will be referred to here as an animation driven game simulation. While Dark Souls did give rise to an entirely new genre, and there have been many other games that have implemented the same feature, it is still rare to see it implemented so thoroughly and successfully.

2. Defining Animation Driven Game Simulation

This mechanic is animation. Specifically, an animation system crafted in such a way that the animations themselves are the driving force behind the game simulation. Animations are almost always designed to communicate some state to the player; they communicate the difference between jumping, and performing a jump attack. (Swink p. 272) The game runs the logic for handling an attack, and while that action is being performed characters and the environment are animated accordingly. Unlike most games, in Dark Souls animations are prescriptive. When animations are driving the game simulation, a barrier of communication between the designer and the player is being torn down. The player is experiencing the game simulation firsthand.

3. Animation Integrated in Other Games

In order to understand the affordances and constraints imposed by such a mechanic, it's worthwhile to first have a general understanding of the standard ways of implementing animation in games. In most games, animations are handled separately from the actions an entity can take. Two common approaches involve animation state machines and frame data.

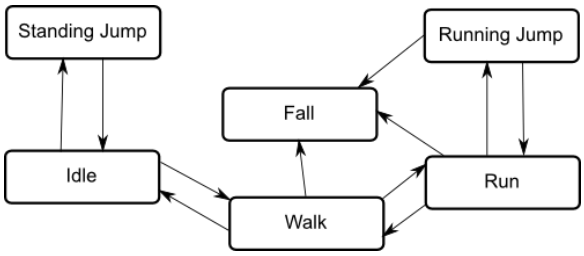

Figure 1: Animation State Machine Example (Unity Technologies)

```
set_attack_value(AT_JAB, AG_NUM_WINDOWS, 3);
//startup
set_window_value(AT_JAB, 1, AG_WINDOW_LENGTH, 4);
set_window_value(AT_JAB, 1, AG_WINDOW_ANIM_FRAME_START, 0);
set_window_value(AT_JAB, 1, AG_WINDOW_ANIM_FRAMES, 2);
//active
set_window_value(AT_JAB, 2, AG_WINDOW_LENGTH, 3);
set_window_value(AT_JAB, 2, AG_WINDOW_ANIM_FRAME_START, 1);
set_window_value(AT_JAB, 2, AG_WINDOW_ANIM_FRAMES, 1);
//recovery
set_window_value(AT_JAB, 3, AG_WINDOW_LENGTH, 10);
set_window_value(AT_JAB, 3, AG_WINDOW_ANIM_FRAME_START, 3);
set_window_value(AT_JAB, 3, AG_WINDOW_ANIM_FRAMES, 3);
set_window_value(AT_JAB, 3, AG_WINDOW_HAS_WHIFFLAG, 1);
```

Figure 2: Rivals of Aether Attack Windows (Mawral, 2020)

3.1 Animation State Machines

The standard process involves utilizing an animation state machine. Game designers work with the software engineers to craft the underlying gameplay simulation. This includes implementing movement, attacks, and all of the other interactions. This could be a variety of subsystems, or a monolith, often including multiple state machines. Parameters (such as velocity, movement state, etc.) within these systems are exposed. At the same time, animators create the necessary animations for the characters and environment. The animation state machine is used to tie the animations to the gameplay. Various animation states are defined to communicate to the player what gameplay state an entity is in. Transitions are created between these various states utilizing the exposed parameters of the gameplay systems. This often ends up in the creation of an animation state machine that attempts to mimic the entity's state. There are some notable repercussions to this practice. For example this opens up the possibility of a state machine to entity state desync, often manifesting in a character stuck in a pose, such as falling or T-pose . Despite the challenges, there are benefits to this method, namely isolating animations, the animation state machine, and the game logic from each other. Each of these components fit nicely into a microservices architecture affording the benefits and drawbacks of such a structure.

3.2 Frame Data

Defining actions via frame data is an alternative to an animation state machine, which is often employed in hack and slash, and fighting games. Frame data is a general term for a data structure that holds all of the unique parameters of an action. It often takes the form of a table or a timeline, not unlike keyframe animations, where each parameter has a specified period and start point defined in frames. When an action is taken the parameters are processed; this results in a wide variety of effects, including the character playing out an animation. Often the animation is stretched to the specified duration. This method brings the underlying gameplay systems closer to the animations. It’s perfect for fighting games where precise balancing is essential. Animations still do not impact the game simulation, but they do tend to be more representative of the underlying process. Animations, frame data, and gameplay systems are all still separated, although there is some increased coupling between frame data and gameplay systems.

Dark Souls Attack.gif

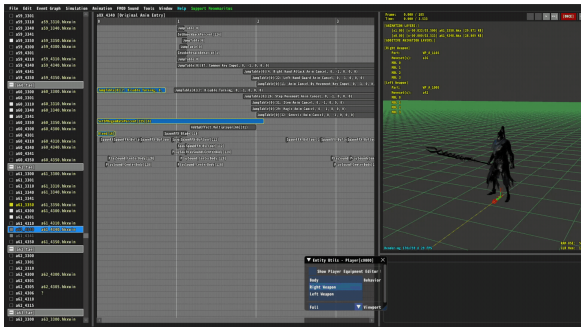

Figure 3: DS Anim Studio Attack (Meowmaritus 2023)

3.3 Animation Driven Simulations

This is the technique employed by Dark Souls. Core to animation driven game simulations is a technique which is itself widely used. Instead of storing game-specific information within the frame data, this data can be directly encoded into metachannels within the animation file. Jason Gregory (2018, p 746) states “It is quite common to define a special channel that contains event triggers at various time indices... Whenever the animation’s local time index passes one of these triggers, an event is sent to the game engine, which can respond as it sees fit.” The key difference here is that it now falls on the animators to define when and how these events trigger. “One common use of event triggers is to denote at which points during the animation certain sound or particle effects should be played. For example, when the left or right foot touches the ground, a footstep sound and a ‘cloud of dust’ particle effect could be initiated.”(Gregory, 2018, p 746) Dark Souls takes this technique and applies it to weapon effects, animation cancels and transitions, along with many other parameters. This puts gameplay implementation directly into the hands of the animators.

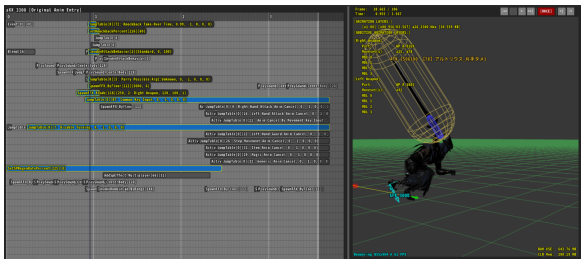

Figure 4: DS Anim Studio HitBox(Meowmaritus 2023)

DSAnimStudio by Meowmaritus (2023) is described as a time act editor. This modding tool is used to both visualize and edit .ANIBND files in Dark Souls. A single .ANIBND file contains all of the .TAE files for a character. .TAE is an animation file used by From Software (the company behind Dark Souls) that combines animations and events. Characters include variants, for example the player and all NPCs are contained within the same .ANIBND file. The .TAE files contain all of the animations and animation events. There are many different types of events, each event containing multiple arguments to further specify the type of event that will occur. These include the common features of playing sound effects such as footsteps, as well as activating gameplay events such as attack behaviors, parry windows, and i-frames. (?ServerName?, 2023)

Game development is an art, just because multiple processes could theoretically produce the same output doesn’t mean that the processes are the same. Almost all programming languages are Turing complete and could theoretically be used for any application. The same applies here as well. What is important are the practical implications of such a mechanic. The affordances and constraints of the animation driven simulation allows the designers intentions to shine through.

3.4 Design goals of the mechanic

The core of Dark Souls combat is positioning and timing. Miyazaki, the director and producer of Dark

Souls, has stated “Each weapon will have characteristics that are vastly different from other weapons in the game.” (Dark Souls Q&A, 2011) This is achieved through letting the animations drive the game simulation. Something as minor as changing a weapon's swing speed ends up having an impact on the game. By fine tuning what events activate and when, each weapon is given a unique identity. Balance isn’t the goal for dark souls, as Miyazaki explains, “We want to give players many options, even if that means they use the sword that 'fits best in the hand.' We want you to become emotionally and physically attached to the weapon you're using.” (Dark Souls Q&A, 2011)

3.5 Affordances and Constraints

When animations are driving the game simulation animators become responsible for implementing gameplay. They must work closely with designers and to some extent become designers themselves. Consolidating roles means fewer people need to be involved to make gameplay changes. Faster iteration times opens up the possibility for increased content and polish. This ability to engage in rapid prototyping of weapons is further enhanced by the modularity of the actions. Any weapon can have its actions swapped out with any other allowing for experimentation by designers, this has been put to practice by the modding community with tools like DSAnimStudio.

The rapid iteration, along with the precision with which events can be orchestrated, allows the designers of Dark Souls to emphasize the basic qualities of combat within a 3D space. Timing is essential to combat, not fast reaction, but observation and planning. The player must watch the enemies’ actions and take advantage of openings, all while keeping in mind the ranges of attacks, and how actions move the character. The fun of weapons in Dark Souls is building an intuitive sense for how they work rather than a logical understanding. Miyazaki stated “Some players will say, ‘is this difficult to use? Leave it to me, I'll master it!’ The mind games and strategising - that's the most fun aspect of the game to me.” (Dark Souls Q&A, 2011)

Actions have many events that fire off throughout their duration. These events are so numerous and varied that there is no way for a player to consciously identify every detail of what is occurring. The effect can however be felt and intuited. The actions are designed consistently. Despite the combat bearing little resemblance to reality, it feels realistic because the game world is following a very specific set of rules that makes sense to the player. Actions don’t function the way they do for the sake of it. Each action exists to contribute to the game world.

4. Conclusion

Dark Souls combat, like other elements of the game, is defined by its artistic consistency. By driving the game simulation via animation, the various disparate mechanics of the RPG are harmonized. The mechanics of the game melt into the background of a greater immersive experience.

References:

Dark Souls Q&A: Variety is the Spice of Death—Souls Lore. (2011, February 14). Retrieved September 16, 2023, from http://soulslore.wikidot.com/das1-variety-is-the-spice-of-death

Gregory, J. (2019). *Game Engine Architecture* (Third Edition). CRC Press.

Mawral. (2020, December 13). *RoA Workshop Guide: Explaining Attack Windows*. Ko-Fi. https://ko-fi.com/post/RoA-Workshop-Guide-Explaining-Attack-Windows-M4M72XOMP

Meowmaritus. (2023). *Meowmaritus/DSAnimStudio* [C#]. https://github.com/Meowmaritus/DSAnimStudio (Original work published 2018)

?ServerName? (n.d.). *TAE - Animation Events—Souls Modding*. Retrieved September 18, 2023, from http://soulsmodding.wikidot.com/format:tae

Swink, S. (2009). *Game feel: A game designer's guide to virtual sensation*. CRC Press.

Unity Technologies (n.d.). *Unity - Manual: State Machine Basics*. Retrieved September 17, 2023, from https://docs.unity3d.com/Manual/StateMachineBasics.html