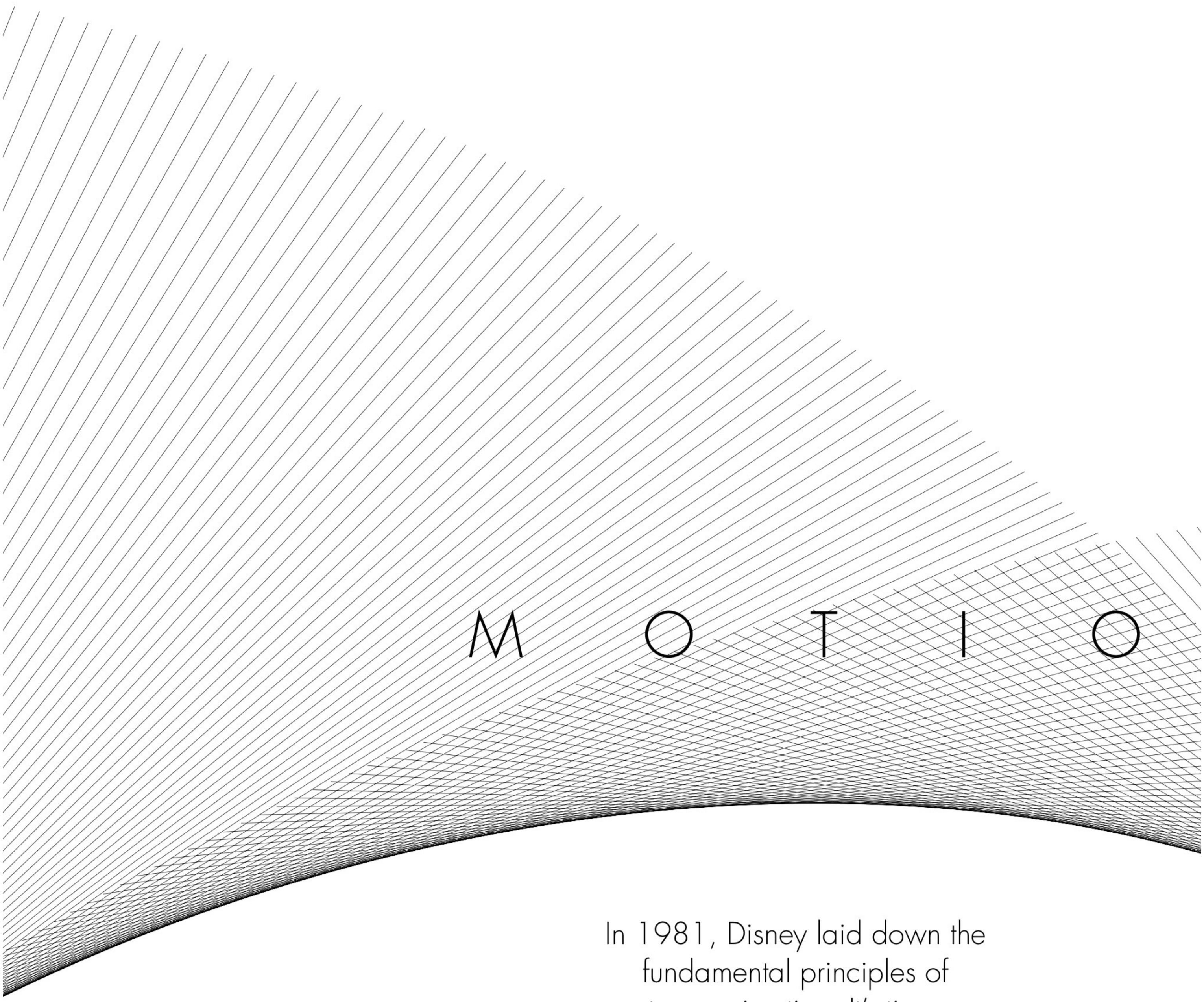

M O T I O

In 1981, Disney laid down the fundamental principles of cartoon animation. It's time we did the same for videogames

By Jonathan Cooper

N P L U S

▶

The 12 basic principles of animation, introduced by Disney's Ollie Johnston and Frank Thomas in the 1981 book The Illusion Of Life: Disney Animation, are a great foundation for any animator. Ultimately, however, they were written with the concept of linear entertainment like TV and film in mind, and the move to 3D kept all of these elements intact due to the purely aesthetic change in the medium. Since 3D animated cartoons and visual effects are still part of a linear medium, they will translate only to certain elements of videogame animation — often only if the game is cartoonish in style.

As such, it is time to propose an additional set of principles unique to game animation that don't replace, but instead complement, the originals. These are the five core tenets of our new nonlinear entertainment medium, that when taken into consideration form the basis of videogame characters that not only look good, but feel good under player control — something the original 12 principles didn't have to consider.

1: Feel

The single biggest element that separates videogame animation from traditional linear animation is interactivity. The very act of the player controlling and modifying their avatars, making second-to-second choices, ensures that the animator must relinquish complete authorship of the experience. As such, any uninterrupted animation that plays from start to finish is a period of time in which the player is essentially locked out of the decision-making process while they wait for the animation to finish (or reach the desired result, such as landing a punch). The time taken between a player's input and the desired reaction can make the difference between creating the illusion that player is embodying their avatar, or is just a passive viewer on the sidelines. That is why cutscenes have for years been the only element in videogames to consistently feature a 'skip' option — because they most reflect traditional non-interactive media, which is antithetical to the medium.

Response

Game animation must always consider the response time — that is, the window between the player's input and the game's response to it — as an intrinsic part of how the character or interaction will 'feel' to the player. While generally the desire is to have the response be as quick as possible, that is dependent on the context of the action. For example, heavier or stronger actions are expected to be slower, while enemy attacks must be slow enough to be seen by the player in time for them to respond.

It is the game animator's remit, often working in concert with a designer and/or programmer, to offer the correct level of response to provide the best 'feel', while also retaining a level of visual fidelity that satisfies all the intentions of the action and the character. It is important to not sacrifice the weight of the character or the force of an action for the desire to make everything as responsive as possible, so a careful balancing act and as many tricks as are available must be employed.

Ultimately, though, the best mantra is that 'the player wins'. The most fluid and beautiful animation will always be cut or scaled back if it interferes too much with gameplay, so it is important for the game animator to have a player's eye when creating response-critical movements — and, most importantly, to actually play the game.

Inertia and momentum

Inertia is a great way to not only provide a sense of feel to player characters, but also to make things fun. While some characters will be required to turn on a dime and immediately start running at full speed, driving a car around a track that could do the same would not only feel unrealistic but would also mean there is no joy to be had in approaching a corner at the correct speed and shaving a second off your best lap. The little moments when you are nudging an avatar because you understand their controls are where mastery of a game is to be found, and much of this is provided via inertia.

Judging death-defying jumps in a platform game is most fun when the character must be controlled in an analogue manner, where they take some time to reach full speed and continue slightly after the input is released. This is also a design and programming challenge, but the animator often controls the initial inertia boost and slowdown in stop/start animations.

Momentum, meanwhile, is often conveyed by how long it takes a character to change its current direction or heading. The general principle is that the faster a character is moving, the longer it takes to change direction — via larger turning circles at higher speeds, or longer plant-and-turn animations if the player wants to turn 180 degrees. Larger turning circles can be made to feel better by immediately showing the intent of the avatar, such as having the character lean into the turn or move their head towards it. But ultimately we are balancing within a very small window of time, lest we render our characters unresponsive.

Visual feedback

A key component of the 'feel' of any action the player and their avatar perform is the visual representation of that action. A simple punch can be made to feel stronger with a variety of techniques related to animation, beginning with the follow-through. A long, lingering held pose will do wonders for telling the player they just did something powerful. The animation on the attacked enemy is a key factor in informing the player just how much damage has been suffered, with exaggeration being a key component.

In addition, employing tricks such as camera-shake will help further sell the impact of landing the punch or gunshot, not to mention visual effects such as blood or flashes to further register the impact in the player's mind. Many fighting games employ a technique named 'hit-stop' that freezes the game for a single frame whenever a hit is registered. This further breaks the flow of clean arcs in the animations, and reinforces the frame on which the impact took place.

As many moves are performed quickly in order to be responsive, they might get lost on the player, especially during hectic fight scenes. Attacking actions can be reinforced by additional effects that draw the arc of the punch, kick or sword-swipe on top of the character in a similar fashion to the smears and multiples used in classic animation. When a sword-swipe takes only two frames to complete its arc, the player benefits mostly from the arcing effect it leaves behind.

Slower actions can be made to feel responsive simply by showing the player that at least part of their character is responding to their commands. For example, an avatar riding a horse can be seen to immediately turn the horse's head with the reins, even if the horse itself takes some time to respond to this and traces a wide circle as it turns. This visual feedback will feel entirely more responsive than the slowly turning horse alone would following the exact same wide turn.

A classic example of this is the difference between the early *Mario* and *Sonic The Hedgehog* games. Both rely heavily on inertia and momentum as core tenets of their gameplay, but whereas Mario will comically run on the spot as he ramps up to full speed or changes direction, Sonic simply runs slowly until he is fast enough to spin, and skids to a stop as he changes direction. Ultimately Mario feels better under player control because, while both characters perhaps handle similarly, at least Mario looks like he is trying hard to instantly match the player's desires.

Jonathan Cooper is an animator at Naughty Dog. This is an edited extract from his new book, Game Anim: Videogame Animation Explained, on sale February 3

2: Fluidity

Rather than being created using long, flowing animations, games are instead made of lots of shorter animations playing in sequence. As such they are often stopping, starting, overlapping and moving between each other. It is a videogame animator's job to be involved in how these animations flow together so as to maintain the same fluidity that they put into the animations themselves. A variety of techniques is used to achieve this, with the ultimate goal being to reduce any unsightly movement that can take a player out of the experience by highlighting where one animation starts and another ends.

Blending and transitions

In classic 2D games, a sprite's animation either played or it didn't. This binary approach carried on into 3D animation until developers realised that, due to characters essentially being animated by poses recorded as values, they could manipulate those values in a variety of ways. The first such improvement that arrived was the ability to blend across (essentially cross-fading animations during a transitory stage) every frame, taking an increasing percentage of a new animation's value and a decreasing percentage of the former as one ended and another began. While more ▶

THE MOST FLUID AND BEAUTIFUL ANIMATION WILL BE CUT OR SCALED BACK IF IT INTERFERES WITH calculation-intensive, this opened up opportunities for increasing the fluidity between individual animations and removing unsightly pops between them.

For a basic example of this, take an idle animation and a run. Having the idle immediately cancel and the run immediately play on player input will cause the character to break into a run at full speed, but the character will pop as they start and stop due to the repeated nature of the player's input. This action can be made more visually appealing by blending between the idle and run over several frames, causing the character to more gradually move between the different poses. Animators should have some degree of control over the length of blends between any two animations to make them as visually appealing as possible, though always with an eye on the gameplay response of the action.

The same situation above can be improved further – albeit with more work – by creating brief, bespoke animations between idle and run (starting) and back again (stopping), with blends between all of them. What if the player started running in the opposite direction to which they were facing? An animator could create a transition for each direction that turns the character as they begin running in order to completely control the character's weight-shift as they lean into the desired direction and push off with their feet. What if they aren't running, but only walking? Again, the animator could create multiple directional transitions for that speed also. As you can see, the number of animations can quickly spiral in number, so a balance must be made between budget and team-size, and the desired level of fluidity.

Seamless cycles

Even within a single animation it is essential to maintain fluidity of motion, and that includes when a cycling animation ends and restarts. A large percentage of game animations cycle back on themselves so it is important, once again, to ensure the player cannot detect when this transition occurs. As such, care must be taken to maintain momentum through actions so the end of the animation perfectly matches the start.

It is not simply enough to ensure the last frame of a cycle identically matches the first: care must also be taken to preserve momentum on each body part to make the join invisible. This can be achieved by modifying the curves before and after the last frame to ensure they create clean arcs and continue in the same direction. For motion-capture, where curves are mostly unworkable, there are techniques that can automatically provide a preservation of momentum as a cycle restarts.

Care should also be taken to maintain this momentum when creating an animation that transitions into a cycle, such as how the stopping animation should seamlessly match into the idle. For maximum fluidity, the best approach in this case is to copy the approved idle animation, and the stopping transition, into the same scene, to manually match the curves leading into the idle, exporting only the stopping transition from that scene.

Settling

This kind of approach should generally be employed whenever a pose must be hit at the end of an animation. It is rather unsightly to have a large movement, such as an attack animation, end abruptly in the combat idle pose, especially with all of the character's body parts arriving simultaneously. Offsetting individual elements such as the arms and root are key to a more visually pleasing 'settle'.

Notably, however, games often suffer from too quickly resuming the idle pose at the end of an animation in order to return control to the player, but this can be avoided by animating a long tail on the end of an animation and allowing the player to exit out of it before the end if they provide a new input. This ability to interrupt an animation before finishing allows the animator to use the desired number of frames required for a smooth and fluid settle.

Settling is generally achieved by first copying the desired end pose to the end of an animation but ensuring elements such as limbs, even divided into shoulder and forearms, arrive at their final positions at different times, with earlier elements hitting then overshooting their goal, creating overlapping animation. Settling the character's root – perhaps the single most important element as it moves everything not planted – is best achieved by having it arrive at the final pose with different axes at different times.

IT IS ESSENTIAL TO MAINTAIN FLUIDITY OF MOTION, INCLUDING WHEN AN ANIMATION ENDS AND RESTARTS

Perhaps it achieves its desired height first as it is still moving left-to-right, causing the root to hit then bounce past the final height and back again. Offsetting the head and limbs in the order of character root lessens the harshness of a character fully assuming the end-pose on a single frame — though care must be taken to not overdo overlap, which may result in limbs appearing weak and floppy.

3: Readability

After interactivity, the biggest differentiator between game and traditional animation, in 3D games at least, is that the former will more often than not be viewed from all angles. This bears similarity to the traditional principle 'staging,' but a game animator cannot cheat or animate to the camera, nor can they control the composition of a scene. So, actions must be created to be appealing from all angles; it is not enough to simply get it right from a front or side-view. Game animators must take care to always be rotating and approving their motion from all angles, much like a marble sculptor walking around their work.

Posing for game cameras

To aid the appeal and readability of any given action, it is best to avoid keeping a movement all in one axis. For example, a combo of three punches should not only move the whole character forward as they attack, but also slightly to the left and right. Similarly, the pose the character ends in after every punch should avoid body-parts aligning with any axis, such as arms and legs that appear to bend only when viewed from the side. Each pose should be dynamic, with lines of action drawn through the character that are not in line with any axis.

For the motions themselves, swiping actions always read better than stabbing motions as they cover an arc that will be seen by the player regardless of camera angle. Even without the aid of a trail effect, a swipe passes through multiple axes, and therefore camera angles — so even if the player is viewing from a less-than-ideal perspective, they should still have an idea of what happened, especially if the character dramatically changes their line of action during poses.

All this said, play to the game being made. If the camera is fixed to the side, as in a 1v1 fighting game, then actions should be created to be most readable from that angle. Similarly, if you are creating a run animation for a game mostly viewed from the rear, then ensure the cycle looks best from that angle before polishing for others.

Silhouettes

At the design and concept stage, the animator should get involved in helping guide how a character might look — and not just to avoid issues such as hard armour-like clothing at key versatile joints such as shoulders or waists. They should also help guide the design so as to provide the best silhouettes when posed. A character with an appealing silhouette makes the job of animating far easier when attempting to create appeal than one composed entirely of unimaginative blobs or shapeless tubes for limbs.

It is advisable to request 'proxy' versions of characters at early stages of development so they can be roughly animated and viewed in the context of the gameplay camera — which, due to wide fields of view, often warp the extremities of character as they reach the screen's edge. Generally, the most appealing characters look chunkier and thicker than they might in real life, due to them being warped and stretched once viewed through the game camera.

Collision and centre of mass

As with all animation, consideration must be given to the centre of mass (COM) of a character at any given frame, especially as multiple animations transition between one another so as to avoid unnatural movements when blending. The COM is generally found over the leg that is currently taking the full weight of the character's root when in motion, or between both feet if they are planted on the ground when static. Understanding this basic concept of balance will not only greatly aid posing, but will also avoid motions looking wrong to the player without them knowing exactly the issue.

This is especially true when considering the character's collision, or location, in the game world. This is the single point where a character will pivot when rotated, and, more importantly, where the character you're designing will be considered to exist in the game at any given time. You will always animate the character's position in the world when animating away from the 3D scene origin, though not so if cycles are exported in place. Importantly, animations are always considered to be exported relative to this prescribed location, so characters should end in poses that match others (such as idles), relative to this position. ▶

4: Context

In linear animation, the context of any given action is defined by the scene in which it plays and what has happened in the story up to that point. The same is impossible in game animation. Often the animator has no idea which action the player performed beforehand, nor the setting in which they are performing the action. More often than not the animation is to be used repeatedly throughout the game in a variety of settings, and even on a variety of different characters.

Distinction vs homogeneity

Due to the unknown setting of most game animations, the animator must look for opportunities to give *character* to the player and non-player characters whenever possible, and must also consider when they should avoid it.

If, for example, the animator knows that a particular run cycle is only to be used by a particular character, then they can imbue it with personality that matches the character description. If they can create a variety of run cycles for that character in different situations, so much the better. Are they strong and confident initially, but later suffer loss or failure and become despondent? Are they chasing after someone, or running away from a rolling boulder about to crush them? The level of distinction the animator should put into the animation depends on how much control they have over the context in which it will be seen.

However, if an animation is not designed for the player character but instead for multiple NPCs, then the level of distinction and notability should generally be dialled down so as to not stand out. Walks and runs must instead be created to look much more generic, unless the animation is shared by a group of NPCs – all the soldiers in a game might run differently from all the civilians, for example. Almost always, the player character is the most unique of all a game world's inhabitants, so this should be reflected in their animations.

Repetition

Similarly, within a cycling animation, if the action is expected to be repeated endlessly, such as an idle or run cycle, then care must be taken to avoid any individual step or arm-swing standing out against the rest lest it render the rhythm of repetition too apparent to the player – ie every fourth step has a noticeably larger bounce.

Stand-out personality can instead be added to on-off actions or within cycles via 'cycle-breakers' such as the character shifting their footing after standing still too long, performing a slight stumble to break up a tired run, or even by modifying the underlying animation with additive actions.

Placement

A key factor in setting the exaggeration of movement is the relative size on screen of the character as defined by the camera distance and field of view. While cameras have drawn closer and closer as the fidelity of characters has increased, players still need to see a lot of the environment on screen for awareness purposes, so many games may show characters that are quite small. Distant cameras require actions to be much larger than life so they can be read by the player.

The same is true of enemy actions that are far off in the distance, such as damage animations to tell the player they landed a shot. Conversely, only really close cameras such as those employed in cutscenes afford subtleties like facial expressions – here, overly theatrical gestures will generally look out of place. It is important as a game animator to be aware of the camera for any particular action you are animating. The wide field of view of the gameplay camera will even distort the character enough to affect the look of your animation, so as ever, the best way to evaluate the final look of your animation is in the game.

5: Elegance

Game animations rarely play on their own. They require underlying systems within which they are triggered, allowing them to flow in and out of one another at the player's input – often blending seamlessly, overlapping one another and combining multiple actions at once to ensure the player is unaware of the individual animations affording their avatar motion.

If not designing them outright, it is the game animator's duty to work with others to bring these systems and characters to life, and the efficiency of any system can have a dramatic impact on the production and the team's ability to make changes towards the end of a project. Just as a well-animated character displays efficiency of movement, a good, clean and efficient system to play them can work wonders for the end result.

Simplicity of design

The industrial designer Dieter Rams once stated that good design "involves as little design as possible", concentrating only on the essential aspects. A good game animation system should similarly involve no more design than is actually required, since bloated systems can quickly become unworkable as the project scales to the often-required hundreds or thousands of animations.

Every unique aspect of character-based gameplay will require a system to play back animations, from the navigation around the world to combat to jumping and climbing to conversation and dialogue and many more. Here the game animator must aid in creating systems to play back all the varied animation required to bring each element of character control to life, and often the desire to create many animations will come into conflict with the realities of production, such as project length and budget.

Thankfully there are many tricks that a team can employ to maximise their animation potential, such as re-using and sharing, layering and combining animations to create multiple combinations, or ingenious blending solutions to increase fluidity without having to account for every possible transition outcome. While the absolutely simplest solution is to do nothing more than play animations in sequence, this will rarely produce the best and most fluid visuals. The smartest approach is to manipulate animations at runtime in the game engine to get the most out of the animations the team has the time to create.

Bang for buck

Just as we look to share animations, being smart about choices at the design stage should create a workable method of combining animations throughout production. This should in turn prevent unique solutions being required for every new system. For example, a well-thought-out system for opening doors in a game could be expanded to interacting with and opening crates if made efficiently. When building any one system, anticipating uses beyond the current requirements should always be considered.

A good approach to system design will produce the maximum quality of motion for the minimum amount of work. It must be stressed that every new animation required not only involves its initial creation, but later modification over multiple iterations as well as debugging towards the end of the project. Every stage of videogame development is multiplied by every asset created, so avoiding adding 20 new animations for each object type is not only cost-effective but also allows more objects to be added to the game. All that said, sometimes the solution to a system is to just brute-force create lots of animations if your budget can allow.

Sharing and standardisation

As mentioned earlier, it is important to know when to keep animations generic and when to make unique ones. If the game requires the player character to interact with many objects in a game, then it would be wise to standardise the objects' sizes so you can use one animation to accommodate all objects of a particular size.

The same goes for world dimensions, where if a character can mantle over objects throughout the game then it makes sense to standardise the height of vaultable objects in the environment so the same animation will work anywhere – not least so the player can better read the level layout and know where they can and cannot vault.

That said, if your gameplay is primarily about picking up objects or vaulting over things, then it may be worth creating more unique animations to really highlight that area and spend less effort on animations elsewhere. This again feeds back into the idea of getting the most bang for your buck, and knowing what is important to your particular game.

All these decisions must come into play when designing systems for your game as very few teams can afford unique and bespoke animations for each and every situation. Nevertheless, beautiful game animation can come from even single-person teams that focus on one thing and do it very, very well. This is the crux of good design. ■

A GOOD APPROACH WILL PRODUCE THE MAXIMUM QUALITY OF MOTION FOR THE MINIMUM AMOUNT OF WORK