33

Infected AI in *The Last of Us*

Mark Botta

33.1 Introduction

In *The Last of Us* a fungal infection has devastated human beings. The pandemic corrupts the mind and has left most of the population grotesquely disfigured and relentlessly aggressive. The survivors have been forced to struggle not only against those overcome by the fungus (known as the Infected) but also against predatory groups of survivors (known as Hunters). This chapter will focus on the AI behind the Infected. Subsequent chapters will discuss the AI for Hunters and buddy characters.

It was our goal to make the Infected feel fundamentally different than Hunters, despite the fact that they use the same AI system. This was done with a modular AI architecture that allows us to easily add, remove, or change decision-making logic. This allows us to create characters that interact with the world in highly varied ways while keeping the code as simple as possible. Simple code is more maintainable, which is crucial when rapidly iterating on new ideas. Developing the Infected required continuous experimentation to discover what worked best. The more quickly new ideas were implemented, the sooner the designers were able to provide feedback. Keeping that cycle of refinement short enabled us to make the Infected feel grounded, entertaining, and believable.

The best way to achieve these goals is to make our characters not stupid before making them smart. Characters give the illusion of intelligence when they are placed in well-thought-out setups, are responsive to the player, play convincing animations and sounds, and behave in interesting ways. Yet all of this is easily undermined when they mindlessly run into walls or do any of the endless variety of things that plague AI characters. Not only does eliminating these glitches provide a more polished experience, but it is amazing how much intelligence is attributed to characters that simply don't do stupid things.

33.2 The Infected

The Infected are humans who have succumbed to the parasitic *Cordyceps* fungus [Stark 13]. The infection ravages the body, leaving no trace of personality, compassion, or even self-preservation. The Infected are driven only by the instincts of the fungus.

Most survivors exist in prisonlike quarantine zones under martial law, but some choose to live outside of the quarantine zones, where they are free but constantly at risk of encountering the Infected or predatory gangs of Hunters.

We wanted the Infected to feel fundamentally different than the Hunters. Hunters work in groups, communicate with words and gestures, and protect each other. The player can see that their cold-blooded brutality is a means to survive. In contrast, the Infected seem chaotic and alien.

Table 33.1 summarizes the four types of Infected that represent the progression of the infection. Runners are the most common type. They are fast and often attack in uncoordinated groups. They can see and, like all Infected, their hearing is far more sensitive than that of the Hunters, making them just as effective in a dark room as they are on a sunlit street. Stalkers are similar to Runners but hide in dark areas and ambush prey. Clickers are visibly disfigured by the infection, with masses of fungus distorting their features and leaving them blind. They have developed a type of echolocation to compensate. They are slower than Runners but have a deadly frenzy attack and ignore melee attacks that don't use weapons. Bloaters are highly disfigured, blind, slow, and heavily armored. They grab and kill any character within reach, making melee attacks ineffective. They throw growths from their bodies that burst into disorienting clouds of irritating dust.

33.2.1 Senses

Our characters rely on their senses to reveal enemies and distractions in their environment. They track how and when they sense each entity and focus on the one that is the most threatening. All Infected, even Runners and Stalkers, rely primarily on hearing.

The sensory system does not reason about the actual audio heard in the game. Instead, logical sound events are generated specifically for this purpose. This gave the game designers more control, allowing them to specify which sounds are heard at what range

Table 33.1 Characteristics of Infected Character Types

Type	Runner	Stalker	Clicker	Bloater
Speed	Fast	Fast	Medium	Slow
Vision	Limited	Limited	Blind	Blind
Rarity	Common	Uncommon	Uncommon	Rare
Combat	Attack in groups	Ambush in dark areas	Melee frenzy, limited melee vulnerability	Armored, ranged attack, invulnerable to melee

by which character type. It also allowed sound designers to add or modify the game's audio without impacting the AI.

Wherever possible, we tried to correlate the logical sound events with actual audio, so that the player would be able to understand (and predict) the reactions of the Infected. There were exceptions, however. In particular, we wanted the AI to be able to sense nearby stationary targets, so we created a logical *breathing* sound that propagates over a very short range but has no audio.

Logical sounds are broadcast over a radius set by the designer, and any character within that radius—whether Infected or not—can hear it. Of course, not all characters hear equally well. We wanted the Infected to hear roughly six times better than the Hunters. We also wanted to be able to vary their hearing sensitivity based on their current behavior. For example, the Infected do not hear as well when they are unaware of the player. This makes it easier for the player to be stealthy, which slows the pace of the encounter, giving the player more opportunity to observe and plan. Thus, for a particular character type in a particular behavior, the radius of a logical sound is multiplied by a tunable value to give the effective radius within which that character will hear the sound.

Similar to actual audio, logical sounds are partially occluded by walls and obstacles. This was done to prevent the characters from hearing through walls as well as to reinforce the player's natural tendency to keep obstacles between them and the Infected. Each time a logic sound event is broadcast, rays are cast from each character within the sound radius to the source of the sound to determine the level of occlusion. The logical sound is broadcast to all characters in range that are not completely occluded.

In order to further differentiate the Infected from the Hunters, we wanted them to be more difficult to approach stealthily. It is a common video game trope to communicate stealthy movement to the player via crouch animations: when crouching, the player can approach another character from behind without being discovered. In *The Last of Us*, this was true of Hunters, but approaching the Infected with impunity while crouched made them feel considerably less dangerous. Our solution was to scale the broadcast radius of logical movement sounds with the speed of the player. This allows approaching the Infected more quickly from farther away but requires moving more slowly at melee range. To communicate this to the player, the Infected enter an agitated state when they start to hear noise, which gives the player a chance to respond before being discovered.

This raises an important point. Communicating the intent of the characters to the player is vital. We rejected some game features simply because they could not be communicated well. For example, Clickers make a barking sound that is an ornamental remnant of a more ambitious design. We originally wanted them to use a type of echolocation to build a local model of the environment (similar to how bats hunt and navigate). Each time that they barked, we would update their mental model of the local area by turning their vision on momentarily. This gave the character a sensory snapshot of threats in the world, but it did not communicate well and confused players that were *seen* by a blind character. We considered giving the bark a visual effect like a ripple of distortion washing over the environment but ultimately decided that this seemed too unrealistic. In the end, we abandoned this approach because it was complex and difficult to convey to the player.

33.2.2 Distractions

When Hunters see the beam of a flashlight or are hit by a brick, they can infer that somebody is nearby. The Infected lack this insight and react only to the stimulus itself. For example, they are drawn to the sound of a thrown object landing rather than deducing the existence of the character that threw it. This response empowered the player to manipulate the Infected and made bricks and bottles as valuable as any weapon.

This response was consistent with their instinctive behavior but it trivialized some encounters. This was particularly problematic when the player used a Molotov cocktail. The sound of the breaking bottle would attract all nearby Infected who would follow each other into the flames, become engulfed, and die. This was entertaining, but much too easy. We solved this by limiting the number of characters that could be affected by the flames. Molotov cocktails are expensive to craft and should be very effective, but not so effective that they take the challenge out of the game.

The Last of Us also allows players to create smoke bombs, which are used to break line of sight and mask the movement of the player. In principle, these should be ineffective against the Infected because they rely so heavily on their hearing, but again, that would take the fun out of the game. Instead, we had them occlude hearing as well as vision. Thus, after being attracted by the detonation of a smoke bomb, the Infected enter the cloud and become essentially blind and deaf. The player is rewarded with the opportunity to flee or to move among the distracted Infected strangling Runners and dispatching Clickers with shivs.

33.3 AI System

Our AI system makes a distinction between the high-level decision logic (*skills*) that decides what the character should do and the low-level capabilities (*behaviors*) that implement those decisions. This separation allows characters with different high-level skills to reuse the same low-level behaviors. For example, movement is encapsulated in the *move-to* behavior that is invoked by many different skills. Furthermore, in this behavior, the decision of where to go is independent of how the character decides to get there. One character may use the *move-to* behavior to move stealthily in the shadows, while another charges forward.

Skills decide what to do based on the motivations and capabilities of the character, as well as the current state of the environment. They answer questions like *Do I want to attack, hide, or flee?* and *What is the best place for me to be?* Once a decision is made, behaviors are invoked to implement it. For example, if movement is required, the skill may invoke the *move-to* behavior and then wait for it to succeed or fail.

The *move-to* behavior attempts to reach the destination using whatever capabilities are available to it. It answers questions like *Which route should I take?* and *Which animations should I play?* It generates paths, avoids obstacles, selects animations to traverse the environment, and ultimately reports the results to the parent skill.

33.3.1 Philosophy

As a general rule, characters don't need complex high-level decision-making logic in order to be believable and compelling and to give the illusion of intelligence. What they need is to appear grounded by reacting to and interacting with the world around them in believable ways. Of course, they also need to avoid doing stupid things. Characters in *The Last of Us* can see and hear threats and distractions, they can navigate by climbing over obstacles and jumping gaps, they can look around corners, they can respond to fire, and they can search for prey. By building a rich set of behaviors that allow characters to interact with the world and picking appropriate behaviors even in fairly simple ways, we create characters that draw the player into the drama of the moment, which sell the story and make the experience meaningful.

In order to be able to reuse skills as widely as possible, we needed to keep them flexible but well encapsulated and decoupled, so that each skill could be adjusted to fit its particular use without affecting other skills. This was very valuable in the late stages of development. Features changed quickly, and it was important to be able to insulate the rest of the code from unstable prototypes. Our modular approach also made it easy to completely remove any failed experiments without fear of leaving any remnants behind. This allowed us to experiment with major changes to the AI right up to the end of the project. For example, Stalkers were conceived of and implemented only a few months before the game shipped.

33.3.2 Data-Driven Design

One key to keeping the code general was to never refer to any of the character types in code, but instead to specify sets of characteristics that define each type of character. All of the Infected character types share a single C++ class and are differentiated only by the set of skills and the values of the tuning variables in their data files. For example, the code refers to the *vision type* of the character instead of testing if the character is a Runner or a Clicker. This may sound like a minor distinction, but it is central to our efforts to keep the code general. Rather than spreading the character definitions as conditional checks throughout the code, it centralizes them in tunable data. This gives designers control over character variations, rather than requiring them to request changes from the AI team, which would slow their iteration times considerably. Furthermore, when adding a new type of Infected or changing an existing type in a fundamental way, there is no need to hunt down all of the places in the code that refer to the character. There aren't any! Just add or remove skills, behaviors, and tuning variables, and everything is kept modular and flexible.

This approach requires constant vigilance. There were many occasions when we were tempted to add a check for a specific character type for *just one thing*. Unfortunately, that *one thing* tends to proliferate as code is extended and reused, or as features are added, or just for convenience when testing. The lesson here is to stay true to your design principles because that consistency will pay off in terms of stability and ease of implementing new features.

An additional benefit became clear when we added the difficulty levels to the game. Because of memory and performance restrictions, difficulty couldn't be tuned by adding more characters to encounters. Keeping all of the character-specific data tunable made it straightforward to configure the difficulty settings for each character type individually. For each character type, designers modified the thresholds at which the Infected would respond to stimuli. For example, at lower difficulty settings, Clickers respond to nearby stimuli by turning and barking, whereas at higher difficulty settings, they are less forgiving and will often chase the player at the slightest provocation.

33.3.3 Implementation

The AI for a specific type of character is implemented as a set of skills that invoke a hierarchy of behaviors. Both skills and behaviors are implemented as straightforward finite-state machines. Skills tend to be tailored to specific character types, while behaviors are more widely reused. For example, Hunters and Infected characters do not share any skills but they share most of their behaviors.

Each type of character maintains a prioritized list of skills. These are tested one at a time from highest to lowest priority to determine if they are valid to run. Testing stops when a valid skill is found. The last skill in the list must always be valid or the character could get into an uncontrolled state and become unresponsive, freeze, walk into a wall, or otherwise look stupid. Skills run until they are completed or are interrupted by a higher priority skill.

33.3.4 Debugging

Maintaining the separation between skills and behaviors has another benefit. We can debug a character by replacing the skills with logic that drives the high-level decisions from a game controller. Of course, nonplayer characters take input differently than a player character. For example, they move to specific locations as opposed to taking directional input from a thumb stick. Conversely, they have controls that the player character doesn't have. They can select from several demeanors (e.g., whether to appear aggressive or relaxed), they have numerous gestures for communication, they have to select look and aim points, and they have a rich set of behaviors to choose from. There are far more inputs available to nonplayer characters than there are buttons on a controller!

To support all of these inputs, an on-screen menu appears when a character is selected for debugging. From here, we can put the character into the desired state, select the desired behavior, and use a cursor to select a destination. We also have convenience options to teleport the character and repeat the last move command, making it much easier to refine animations and behaviors without needing to rely on waiting for the skills to use them.

33.4 Skills and Behaviors

Table 33.2 contains the skills used by each type of Infected, sorted by priority. Priorities specify which skills should be allowed to interrupt each other. For example, the *chase* skill is allowed to interrupt the *search* skill but can be interrupted by the *on-fire* skill. The *wander*

Table 33.2 Prioritized Skills for Infected Character Types

Runner	Stalker	Clicker	Bloater
On-fire	On-fire	On-fire	On-fire
Chase	Ambush	Chase	Throw
Search	Sleep	Search	Chase
Follow	Wander	Follow	Search
Sleep		Sleep	Follow
Wander		Wander	Sleep
			Wander

skill is the fallback for all types of Infected and is always valid. Conceptually, if the Infected aren't busy doing anything else, they'll be wandering around their environment.

Our goal was to share as much of the AI between the Infected characters as possible so that new character types could be added easily. When development started, it was unknown how many character variations there would be or how they would behave. We experimented with different types of game play and a variety of character types. Each variation went through several iterations before it showed promise and was kept for further refinement or until the potential was exhausted and it was cut. As a result, most of the skills are shared by all Infected character types. Only two skills are used by a single character type: the *ambush* skill used by the Stalker and the *throw* skill used by the Bloater. The Infected spend most of their time in the *wander* and *chase* skills. Similarly, most of these skills invoke behaviors that are also used by the Hunters. There are only two behaviors that are unique to the Infected: the *infected-canvass* behavior and the *infected-listen* behavior.

33.4.1 *Search* Skill

Hunters have a complex search skill that enables them to uncover all hidden locations in an encounter in a coordinated way. The Infected, on the other hand, are not particularly well organized or thorough, so an Infected-specific search skill was needed. The *search* skill becomes valid to run when an Infected loses track of the player during a chase. The search is not exhaustive; they eventually lose interest and resume wandering around. The intent of the skill is to keep the player moving as the character explores nearby hiding places.

Hiding places are polygons on the navigation mesh that are not visible from other parts of the navigation mesh. When the search begins, a graph of *search points* is generated that reveal hiding places. A search point is added at the center of the polygon containing the last known location of the target entity. A breadth-first traversal visits neighboring polygons until line of sight to the search point is broken. A new search point is added at the center of the parent polygon (i.e., the last polygon with line of sight to the previous search point). This process continues until the entire search area is visited or a fixed number of search points have been added. The result is a graph of points with visibility to each other and to all of the hiding places around them.

Points are selected on the search graph based on the predicted location of the player [Straatman 06]. This has the effect of propagating desirable search points over time, which the Infected explore in a natural-looking search pattern, and puts pressure on players who tend to stay put. When an Infected reaches a search point, it briefly investigates before moving on.

When investigating, we wanted the Infected to appear to move around frantically and cover as much of the immediate area as possible without looking planned or methodical. Also, since search points can be generated anywhere on the navigation mesh, the area to search could be an arbitrary shape and size and may contain obstacles like debris or furniture. The *infected-canvass* behavior was designed to meet these requirements. It works as follows (and is illustrated in Figure 33.1):

1. A logical grid is placed over the area covered by the *canvass radius* and centered on the Infected.
2. Obstacles and the area outside of the canvass radius are marked as *seen*, leaving an area of cells that need to be checked.

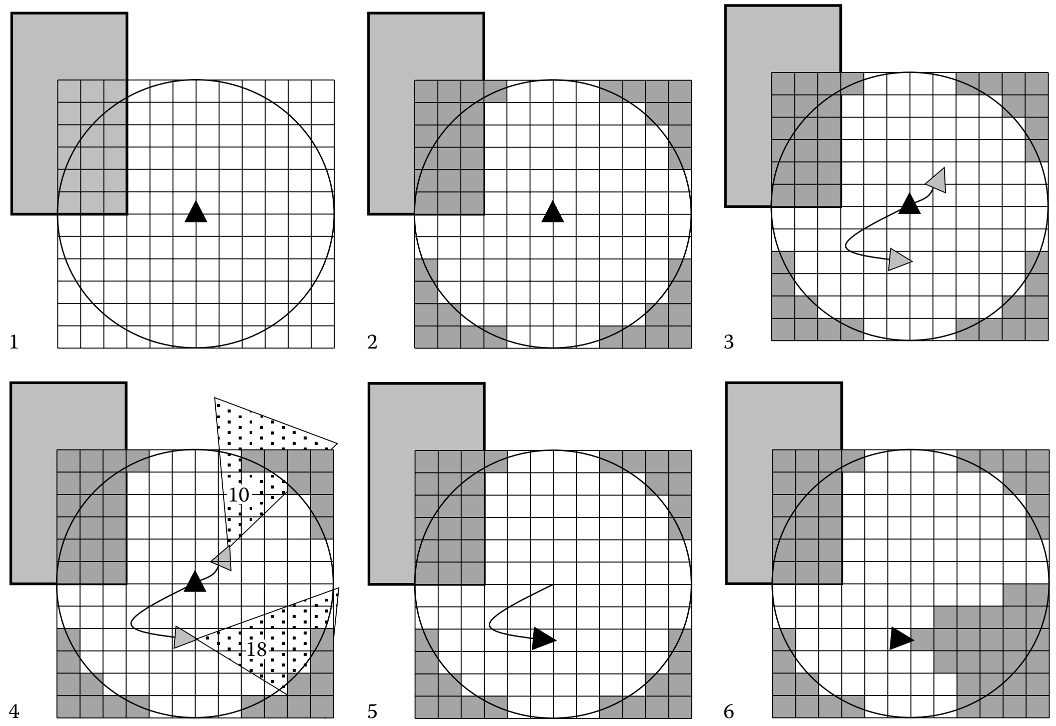

Figure 33.1

Selecting an animation in the *infected-canvass* behavior near an obstacle.

3. The invoking skill provides the behavior with a collection of animations that it can use. We determine the location and orientation that the character will have at the end of each of these animations.
4. The number of *unseen* cells in a sensory wedge in front of the character is counted for each animation. The wedge need not match the area covered by the actual senses. A larger wedge gives more coverage for a faster search and a smaller wedge gives a more thorough search.
5. Animations that result in more *unseen* cells are more desirable. Animations that were played recently are less desirable. The most desirable animation is played, moving the character.
6. When the animation has completed, the cells in the sensory wedge are marked *seen* and the process repeats at the new location from step 3.

The *Infected-canvass* behavior never explicitly selects the direction the character moves—it is based solely on the available animations. This made the code much simpler and animators were not restricted to producing animations in a fixed number of directions, which gave them the freedom to create a larger variety of expressive performances. Designers had control over the canvass radius and how long the character continues to canvass. For variety, the behavior includes sets of animations that move the character both near and far. In a small area, the character uses the near set of animations and turns quickly to look in all directions in an unpredictable (and seemingly random) order. In larger areas, the far set of animations are added to the mix, resulting in the character moving across the canvass area in short and long bursts. The process is not exhaustive, but it is convincingly frantic, it looks organic, and it is very flexible. The behavior was used by several other Infected skills in addition to the *search* skill.

33.4.2 *Chase* Skill

Most of the Infected have almost trivially simple combat tactics: when an entity is sensed, they get to it as fast as possible and attack it. This is done by the *chase* skill.

When an entity is first encountered, the Infected turns toward the stimulus and screams to alert the player that they have been discovered. The *chase* skill does this by invoking the *surprise* behavior that selects from a set of animations that orient the character in the appropriate direction.

Next, the *chase* skill invokes the *move-to* behavior, which provides an interface to the navigation system and is responsible for moving the character to a location or entity. As the entity moves through the environment, the *move-to* behavior continuously updates the path, leaving the parent skill to simply monitor for success or failure. This is one of the key advantages of the modular behavior system—it allows us to keep the high-level logic in the skills very simple.

During the chase, the character will occasionally pause to reorient, giving the player more opportunity to hide or prepare an attack. Instead of simply pausing movement, the *infected-canvass* behavior is used for a very short duration in a small area and with a special set of animations, which gives the Infected the appearance of frantically trying to reacquire the player during the chase.

33.4.3 *Follow* Skill

Unlike Hunters, the Infected do not communicate with one another. They share no information anywhere in the code. This is consistent with the idea that they are not intelligent and only respond instinctively to the stimuli in their environment. Consequently, if an Infected senses the player and starts to chase, any nearby Infected that do not directly sense the player are oblivious. Although this makes sense, in practice it made the Infected seem a bit too insensitive to their environment.

As an alternative, we added the *follow* skill, which allows one character to follow another that is chasing something. The following character does not receive any information about the entity being chased; it just has the compulsion to follow along. Conceptually, the Infected seek the opportunity to claim the unknown prey for themselves. As a result, when an Infected is alerted and starts chasing the player, it may pick up others along the way, but as the player expects, the alerted character will be the first to arrive.

33.4.4 *Ambush* Skill

The *ambush* skill is a combat skill used by Stalkers that replaces the *chase* skill used by all of the other Infected types. It was created late in the development of the game while experimenting with different character types. The character types we had were solid but we wanted more variety. We experimented with a light-sensitive character that the player could hold at bay with a flashlight. This showed promise but when the characters reacted quickly, they didn't feel dangerous, and when they reacted more slowly, the flashlight didn't feel effective. After a few iterations, we decided to take advantage of the dark in a different way. The idea was to have the character move from cover to cover, providing only glimpses to the player in dark environments. They would lay in wait until the player wandered too close, then attack and run back into cover. This simple concept proved very effective at heightening the sense of horror. In encounters with Stalkers, the player naturally became more defensive and approached each corner with caution. Keeping the Infected out of sight also served to hide the number of characters in the encounter, which made tracking down the last Stalker as tense as the first.

Ambush is the only Infected-specific skill that uses cover. Cover locations are selected using a system that evaluates points in the environment based on how suitable they are to ambush the player or to retreat after an attack. This system is shared with the Hunters and will be described in more detail in the chapter describing their AI.

33.4.5 *Throw* Skill

The Bloater is the only Infected character with a projectile attack. The fungus has swollen its body into a misshapen mass of armored plates and bulbous fungal growths. The Bloater rips these growths from various armored plates and throws them ahead of the moving player to produce a cloud of particles that briefly slow movement. The armor plates spawning these growths can be destroyed, and the *throw* skill plays animations that select from other plates until they are all destroyed and the Bloater dies. Until then, there is no limit to the number of growths a Bloater can throw, which prevents it from becoming less challenging as the encounter progresses.

33.4.6 *On-Fire* Skill

It may be odd to consider reacting to being engulfed in flame as an AI skill because, unlike other skills, it doesn't show intent and isn't a conscious decision. However, the goal here is not so much to show the intent of the character as it is to allow it to react to its situation and to make it more believable and entertaining.

The *on-fire* skill works by invoking the *infected-canvass* behavior briefly in a small area, with a custom set of animations. This causes the character to flail and dart around chaotically in an arbitrary environment without running into walls. It also provides emergent variation in the reaction because it combines several smaller reactions for the full effect. This is a great example of a case where we were able to reuse the same behavior with different contents (in this case, different animations) in order to achieve a very different, customized, and compelling result.

33.4.7 *Wander* Skill

The *wander* skill is the lowest priority skill for all of the Infected character types. It is the fallback that is used when nothing else is valid to run. When the Infected are not agitated, they will move throughout the environment either randomly or in predictable patterns. The challenge was to do this in such a way that the player is forced to make tactical decisions about when to move, distract, or attack.

Designers can specify whether the character wanders on a fixed route or randomly. Fixed routes require laying out a spline in the level editor and specifying environmental interactions at various points. These can be as simple as pausing and looking around or as complex as triggering a brief behavior (such as scavenging through debris) before returning to moving along the spline. Random wandering selects a random polygon on the navigation mesh as a destination. As the character moves, it keeps track of the polygons it visits. On arrival, it randomly selects a polygon that hasn't been visited and continues to wander. This has the effect of covering a large area in an unpredictable way.

Both types of wandering have their uses. A fixed route is predictable, which makes it useful for crafting stealth encounters where the player is encouraged to deduce the pattern and avoid it. Random movement is useful for covering a large area in unpredictable ways, offering the player a different challenge. In practice, fixed routes are used for the initial setup of an encounter, and random movement is used after leaving combat when the character may be far away from the original route.

33.4.8 *Sleep* Skill

Designers can configure the Infected to sleep when idle instead of using the *wander* skill. This is an idle state that greatly reduces the sensitivity of the senses, giving the player the opportunity to avoid, distract, or dispatch the character more easily. This skill was also useful for creating sentries at choke points throughout the game, since sleeping Infected stay where they are placed.

Sleeping characters react to disturbances in several ways. If the player moves too quickly or too near a sleeping Infected, it will stir briefly before settling back into slumber. This is feedback to inform the player that they should exercise more caution. If the player makes a loud noise from a great enough distance, the Infected will wake up and use the *infected-canvass* behavior to move about a small area searching for the disturbance before going back to sleep. If the player continues to disturb the character or is sufficiently loud, it will wake enraged and *chase* the disturbance. When the chase ends the Infected will wander through the environment instead of going back to sleep.

33.5 Conclusion

The Infected are a compelling part of a complex world. They are the result of design, art, sound, animation, and programming coming together to make characters that fit seamlessly into the environment, making the world more believable. They are entertaining characters and are worthy opponents that command respect but can be overcome with skill.

The capabilities provided by the behaviors allow the skills to focus on high-level decisions. This simplified the development process and enabled new ideas to be rapidly explored in order to refine the Infected into engaging opponents. Keeping the code simple and modular made it more maintainable and stable. This made it easier to isolate glitches that made the characters look stupid. Eliminating these problems went a long way toward making the Infected look smart.

Skills and behaviors are a small part of a complex AI character, but they provide the decision making needed to make the character respond convincingly to the player. Keeping it simple was the key. Using tuning values that were independent of the type of character allowed all of the Infected to share the same skills and behaviors but respond in ways that were unique and interesting.

References

[Stark 13] Stark, C. 2013. The creepy, real science behind "The Last of Us." http://mashable.com/2013/07/26/the-last-of-us/ (accessed September 10, 2014).

[Straatman 06] Straatman, R., A. Beij, and W. van der Sterren. 2006. Dynamic tactical position evaluation. In *AI Programming Wisdom 3*, ed. S. Rabin, pp. 389–403. Boston, MA: Charles River Media.