35

Ellie: Buddy AI in *The Last of Us*

Max Dyckhoff

35.1 Introduction
35.2 Starting from Scratch
35.3 Ambient Following
35.4 Taking Cover
35.5 Combat Utility
35.6 Armed Combat
35.7 Finishing Touches
35.8 Conclusion

35.1 Introduction

In the last couple of chapters, you have read about the hunter and infected AI from Naughty Dog's third-person action adventure game *The Last of Us*. This chapter moves on to discuss the buddy AI.

The whole game focuses on relationships between characters and none more strongly than that of Ellie and Joel, the player's avatar. Ellie accompanies the player throughout most of the game, and we needed the player to love her companionship. We were acutely aware of the common pitfalls with AI companions: the risk of them turning into a tedious "escort" quest, generally getting underfoot, or turning into mindless drones with no agency in the world. We had to make sure that Ellie never annoyed or frustrated the player and that she remained a compelling entity throughout the game. Most importantly, we had to build a character that the player cared for.

Ellie is new to the world outside the military safe zone in which she grew up and new to the brutality and violence of the infected and humanity at large. She is seeing landscapes, cities, and actions as the player would, instead of through the world weary eyes of Joel. She is a young and fragile character in a dangerous world, and we wanted the AI to highlight this as well as support her growth and strength later in the game.

35.2 Starting from Scratch

The first decision we had to make was to start from scratch with just a few months of development remaining. Five months before we shipped, we had to present a press demo of the game, a combat encounter through a tight environment with a dozen or so infected. We wanted it to showcase how tense it would be fighting the infected and how having buddies with you would both aid you and increase the emotional impact of an encounter.

At that time, the two buddies in the encounter, Tess and Ellie, exhibited all the negative traits of buddy AI that we were trying to avoid: positioning themselves awkwardly, shooting at the wrong time, and generally getting underfoot. To avoid this, they had to be scripted for the entire duration, explained away using the video game trope of "staying back." Doing this completely removed them from the encounter and the player's attention; it was clear this was an unacceptable approach.

35.2.1 The Plan

The first thing we decided on was to keep buddies really close to the player. If a buddy is close to the player character, she is less likely to run into problems with enemy AI. Additionally, if she is far away from the player, she is easily forgotten, and the player will feel alone in the world, which is contrary to our storytelling goals for the game.

Once she reliably stayed close to the player, we started to give her utility in the form of both noncombat and combat behaviors. With these in place, she no longer felt like an escort quest. If she helps the player out of a tight situation, then the player feels thankful, grows to appreciate her presence, and has an improved opinion of the game in general.

Finally, we added touches to make her an interesting character, with a progression of combat vocalizations, a library of ambient animations for her to play, and a suite of noncombat conversation tracks. These systems are driven by the content creators, and with minimal effort, a character can become much more engaging.

35.2.2 Approach

One extremely important aspect of the development was how we thought about Ellie and the other buddies. We always treated her as an actual human being and spent time trying to get into her head, reasoning what she would do in a given situation and why. We tried to avoid the "gamification" of her AI as much as possible, restricting ourselves from cheating or doing things that would seem inhuman. We grew to genuinely care for Ellie and her well-being.

The decision to avoid cheating was extremely important. It's easy to cut corners by cheating, but even if the player never notices, it still moves you away from creating a living, breathing character, which was completely at odds with our goals for Ellie. There were exceptions, but we stuck to this policy as much as we possibly could.

35.3 Ambient Following

There are a lot of reasons for keeping a buddy character close to the player. Primarily, if she is sharing the same space as and behaving similarly to the player, then her actions can by definition be no more stupid than what the player is doing. If she has positioned herself correctly and is still seen by an enemy, then the player will have been seen too and consequently will attribute the failure to themselves rather than the buddy.

Additionally, a buddy that is close to the player has increased opportunity to trigger relevant dialogue. This dialogue can both serve as character exposition and provide gameplay utility. For example, rather than having the buddy run out from hiding next to the player when she is about to be seen by an enemy, she can vocalize the threat and allow the player to handle it. We can also vocalize exclamations of surprise or good fortune, augmenting the tension and reward of an encounter.

In the context of *The Last of Us*, when the player is separated from Ellie and no longer has her vocalizations of danger to rely on, the encounter becomes increasingly tense, and consequently, the relationship with the buddy is strengthened.

35.3.1 Follow Positions

To enable a buddy character to follow the player, or another character, we created a follow system that generated and evaluated positions near the leading character, and then moved the buddy there elegantly. The core of this was a follow region, a torus around the leader described by a set of parameters provided in data.

A number of candidate follow positions were generated inside this follow region and then evaluated for quality. The generation of the candidate follow positions was done by casting three sets of navmesh rays, as shown in Figure 35.1.

A first set of rays fan out from the leader position to the follow region in order to make sure that there is a clear line of movement from the buddy to the player. One candidate follow position is generated for each ray that reaches the follow region; see Figure 35.1a.

A second set of rays are then cast forward from each candidate position to make sure the position isn't facing a wall (Figure 35.1b). We tried allowing positions that would be close to a wall and just had the character face away from the wall, but in practice, a buddy moving right next to a wall feels unnatural.

Finally, rays are cast from the player's location to each "forward" location, to ensure that movement forward from this location wouldn't put an obstacle between the player and the buddy (Figure 35.1c). This ray may seem unnecessary, but in testing, we found that

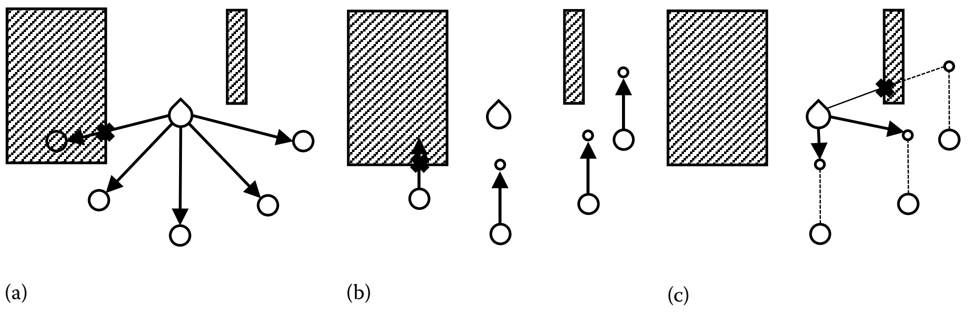

Figure 35.1

The generation of follow positions by casting a set of pathfinding raycasts. (a) Cast rays to generate candidate positions. (b) Cast rays to check forward direction. (c) Cast rays to check future position.

without it, Ellie would choose to stand on the other side of a doorway or fence. Adding this ray gave her a desire to stand in the same "room" as the player.

The resulting follow positions are then rated every frame, and an optimal position is chosen for the buddy to move to. These ratings are based on a number of criteria including

- Distance to the leader
- Staying on the same side of the leader
- Visibility of potential targets (either hiding or exposing)
- Not being in front of the player
- Distance to other buddies

35.3.2 Moving

Once we had chosen a position to move the buddy toward, we had to make sure she would perform the movement in a believable manner. We would watch her follow the player around a space, and if we saw a movement that seemed unnecessary or unnatural, we would pause the game and start investigating the cause.

From this, a lot of filters were added to the movement to make it appear more intelligent. First, we limited short-distance moves, only allowing them if they were absolutely necessary, for example, if her current location was exposed by an enemy. Then we prevented her from running past the leader character if possible, unless not doing so meant stopping in an undesirable location. It appeared very forced if she picked a follow position on the opposite side of the player and mechanically ran past the player. Allowing her to get "close enough" to her follow position, and not crowd the player, gave her follow behavior a very organic feel rather than the rigid behavior she had before.

Of course most of the time we are not generating follow positions around a stationary leader, as the player will be moving around the environment too. As the player moves, both the follow region and the generated follow positions move too. We regenerate follow positions each frame, but we include criteria in the position ratings to try and smooth out any noise.

In an ambient environment where the player is casually exploring a place, Ellie is allowed to saunter after the player without much urgency, while during combat, we like to keep her very close to the player. This meant creating a mapping from the desired movement distance to the buddy's movement speed to allow us to specify how urgently the buddy should move to her follow position. For short distances, a walk is sufficient; for extremely long distances, she must sprint; and in between, she may run.

Each movement mode (walk, run, and sprint) had a single fixed speed, which varied across buddies and was different from the player's movement speed. This meant that when following the player, she would approach some ideal distance from him and then oscillate between running and walking, which was clearly unnatural. To prevent this, we allowed the buddy to scale her movement animation speeds by as much as 25%, allowing her to stay in a given movement mode for longer. As she neared the threshold to start walking, she would scale back the speed of her run.

35.3.3 Dodging

We made the decision to have buddies not explicitly move out of the way of the player if they were there first. If the player and a buddy character are both stationary, we find that the player is not typically inclined to run straight at her, just as you would not run straight at a real-life friend standing near you. Instead of preemptively making her move, we assume the player is going to try and go around her and only at the last second make her play a canned animation dodging out of the way (along with a vocalization admonishing the player for encroaching on her personal space). This approach to dodging took what can frequently be a really frustrating and unrealistic behavior and turned it into a character element, making the player subconsciously respect Ellie and her choices.

35.3.4 Teleportation

One popular method to keep buddy characters near a player is teleportation, which can be effective but has a number of downsides. We decided very early on to never teleport the buddies for the purpose of keeping them near the player.

There were a number of reasons for this. The audio department didn't want the buddy's footsteps to suddenly appear closer to the player. We also didn't want to have situations where you know that Ellie is on one side of you and then suddenly she's on the other with no explanation for how she got there. You can write robust checks against this sort of issue, but if the player gets even a hint that teleportation is happening, it feels a little strange and can break the suspension of disbelief. We also just wanted to avoid it on principle, believing that if we aimed to make Ellie keep up with the player in all situations, then the end result would be more robust and believable. When we avoid cheating, we are forced to address the situations where we might cheat in a realistic way, and this creates small details in the character's behavior that make Ellie feel like a living, breathing character who is grounded in the world in which you are both participating.

Ultimately, the only time a buddy will teleport is when they need to immediately aid the player with a melee struggle, which is discussed later. We also considered allowing teleportation to bring Ellie closer to the player in cases where she was sufficiently far away, but in practice, this was rarely necessary.

35.4 Taking Cover

The follow system described earlier was largely used for ambient (i.e., noncombat) situations. Combat following, and particularly cover selection, requires something slightly different.

The Last of Us is a cover-based third-person game, so it was necessary to have the buddy characters take cover with the player efficiently. Our tools create "cover action packs" that represent places an AI character can take cover against a wall or similar obstacle, but ultimately this representation was too coarse for our needs. While enemy NPCs only require sparse information about where they can take cover, the player can take cover anywhere in the world. If we were going to keep the buddy characters nearby, then they needed to have the same ability.

35.4.1 Runtime Cover Generation

We used a hybrid approach to generate runtime cover, combining the system used by the NPCs with the dynamic system used by the player. We already have "cover edge" features generated by our tools, which are used to create the cover action packs that the enemy AI uses. These simply represent the intersection of a wall and floor collision plane, with a markup to indicate height and cornering.

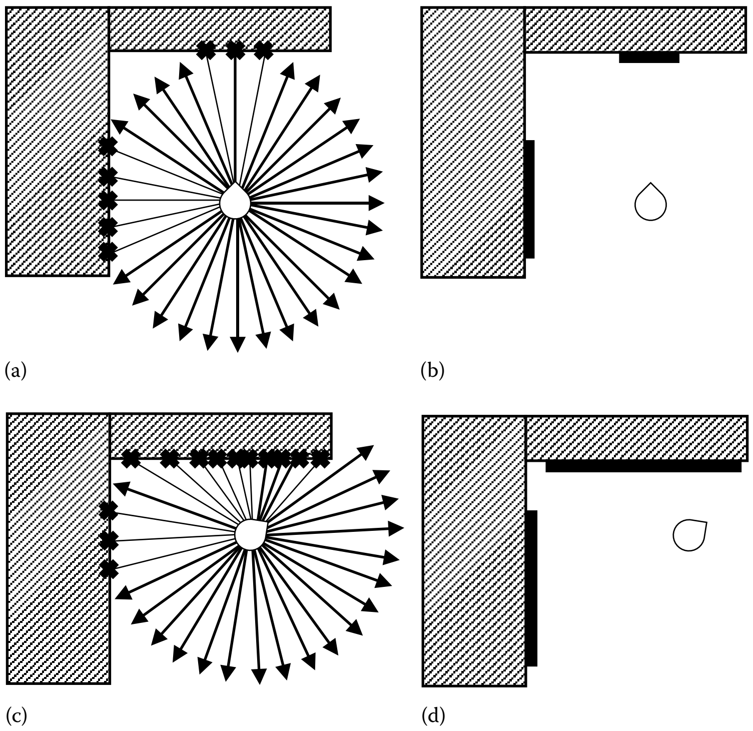

Figure 35.2

Generation of procedural cover edge features using collision raycasts. (a) Fire rays to find nearby collision. (b) Combine similar collision points to form cover edge features. (c) Fire rays again, when the leader moves. (d) Combine collision points with previous cover edge features to create larger edges.

To generate the runtime cover, first, we scan the environment around us and look for nearby cover edge features. We then augment this with a set of procedurally generated cover edge features. We cast a fan of raycasts from the leader's position against the collision geometry (see Figure 35.2a), compare the normal of the collision plane and location of each hit point, and then combine nearby points with a similar normal into procedural cover edge features (Figure 35.2b).

These two sets of cover edge features, one static and one procedural, are then combined and cached for future use. As the leader character moves, we perform the same tests (Figure 35.2c, d) and combine the results with the previous results. We only evict cover edge features from this cache when we start running out of storage. This combined set of cover edge features benefits from both the information derived by the tools and the runtime features, creating a more accurate set than either method alone. Unfortunately, the process was very computationally expensive, so doing it for all characters was unfeasible, and we had restrict it to just the buddy NPCs.

This set of cover edges is then rated to choose an ideal edge on which the buddy character should take cover. A large amount of time went into generating and tweaking the set of rating functions for cover edges, the most important being

- Visibility to enemies, a dot-product check
- Proximity to leader
- Predicted future visibility, derived by projecting enemy movement

35.4.2 Cover Share

Originally, the player was unable to share cover with a buddy; in fact, if the buddy saw the player approaching, she would pop out of cover, assuming that the player wanted to be in that location. This removed intentionality from the buddy's movement, much in the same way as dodging would, as discussed earlier. In order to solve this, we created an animation set in which Joel was able to enter cover on top of a buddy character, placing his hand on the wall next to her and shielding her with his body. The player was able to perform all the normal actions such as shooting or moving, and we didn't need to move the buddy at all.

One time, because of a bug, Ellie entered cover through Joel into this shared cover state. We realized it looked really believable, and with some small modifications to the animation set, we made it work all the time, allowing Ellie to take cover exactly where the player was.

We also added special dialogue lines, so the buddy character would comment on recent combat while sharing cover with the player. This cover share system really enhanced the bond with Ellie. It took our "stick close to the player" mantra to its logical conclusion, and it was virtually impossible as a player not to feel some compassion for Ellie when in cover share.

35.5 Combat Utility

Once we had the buddies following the player as robustly and smoothly as possible, it was time to move on to make them useful in combat. We always wanted Ellie and the other buddies to explicitly support the player rather than go off and do their own fighting. There were a number of ideals that drove our development of the buddy combat utility.

The game's difficulty was very carefully balanced, and we wanted to ensure the buddy performed consistently in a given encounter so that they wouldn't upset this balance. In particular, we didn't ever want a buddy to overpower an encounter and make the player feel useless. It was also important to make sure that the player saw the buddies doing things in combat. This is true of all AI characters but particularly so for buddies; they can be the most intelligent AI ever created, but if the player doesn't see what they are doing, then it is wasted effort.

There are also large sections of the game where Ellie is unarmed, but she still needs to interact in combat and support the player. This is where we started our development of combat behaviors.

35.5.1 Throwing

One of the most visceral combat actions Ellie performs is when she throws a brick at an approaching enemy. The logic to decide when to throw is very simple; we need to make sure that she isn't giving away the player's location but that she still throws the brick promptly enough to be effective. One of the primary triggers for the throw action is driven by hooking into the enemy perception and movement systems and predicting if they will be able to see the player in the near future.

These thrown objects stun the target for a few seconds, allowing the player to follow up in a number of ways: running away, melee attacking the enemy, grappling, etc. The throw action is put on a really long timer, to ensure it doesn't get overused. We always want it to be special and memorable when such an action occurs.

35.5.2 Grapples

Next, we added melee grapples, allowing an enemy to target and grab a buddy character. This was largely done through data by design, thanks to a verbose melee scripting system. We originally required the player to save Ellie every time she was grappled, but this quickly became tedious and clearly tipped us toward the "escort quest" design we were trying to avoid. Instead, we gave her a suite of animations allowing her to escape the grapple herself, after which she would reposition away from the temporarily stunned enemy, and we would disallow her from being targeted again for 15–30 s.

We kept the requirement for the player to save her in certain situations, but it was always done intentionally and infrequently. We would make sure she was visible and easily accessible to the player and that the player wasn't currently grappled himself. We wanted the player to want to protect Ellie, just not so often that it was irritating.

We also implemented the reverse situation; sometimes the player is grabbed by an enemy, and buddy characters are able to save you. People tend to feel really grateful if Ellie stabs a guy who has them in a headlock; this can take a desperate situation and turn the tables.

As mentioned earlier in this chapter, grapples are the only time we allow teleportation of a buddy. We decided that if the player is being grappled and we want Ellie to save them, then it's an acceptable time to teleport. During the grapple, the player has no camera control, so we can ensure it is facing in a direction that does not reveal the teleport.

35.5.3 Gifting

We already had one scripted sequence where one of the buddies, David, gave the player a box of ammo while fighting off infected at the start of the Winter chapter. Players reacted so well to this that we felt it should be something systematic that Ellie could do.

The game uses a complex drop system that works out what supplies should spawn in an area and what dead bodies should drop, and we just hooked straight into this to figure out what Ellie should give the player. We restricted it to ammo and health packs, meaning that the player would still need to scavenge for crafting items to apply as weapon upgrades. Because it was tied directly into the existing system, it didn't change the difficulty balancing that had already occurred, despite coming online very late in development.

We added another really long timer to her gifting, to prevent annoying the player. In the first playtest that we ran with gifting enabled, we noticed one player was stuck in a particular spot, and Ellie would give him ammo every minute. It not only highlighted that the level design needed some fixing, but it devalued the act of gifting. When properly tuned, it really helped to enhance the bond between her and the player; she would whisper and hand you something useful at exactly the right times.

35.6 Armed Combat

Having an extensive suite of unarmed combat behaviors, it was time to focus on making Ellie use a gun effectively and believably. For positioning, we used both the procedural cover edge features and the follow positions discussed earlier. Again we stuck with our mantra of staying near the player, and the settings for generation of follow positions were tuned to bring her closer to the player and allow her to move more responsively.

For the most part, we would try and mirror the player's decisions, crouching and taking cover or standing upright and out in the open in conjunction with the player's character. When in doubt, we would always prefer to select a cover location, as that was rarely a bad place to be.

Initially, we planned for the adult buddies like Tess and Bill to operate more independently in the environment, but in practice, it always felt like they were abandoning the player, and we didn't want that. Instead, they all behave like Ellie, although with more relaxed parameters for positioning and movement.

35.6.1 Shooting

The game heavily emphasizes stealth, so having a buddy give away your location is extremely infuriating and breaks all of the trust that the buddy has gained. Thus, knowing when (and when not) to shoot is absolutely key.

The first thing we did was reverse the logic for shooting; instead of being happy to shoot most of the time—like the enemy NPCs—Ellie wants to not shoot. This not only makes sense for AI reasons but also for real-world reasons; she is a young girl and hesitant to use a gun. We then built logic to give her permission to fire.

If the player is actively firing a weapon or engaging in noisy melee combat, then Ellie is allowed to fire upon her target. In addition, we model the intentions of enemy NPCs and allow her to fire if the player is in immediate danger. Finally, if the player was trying to sneak away from an encounter, we tried to recognize this by comparing the player's location with the enemy NPC's perception of where the player was. If these were significantly different, then we considered the player to be back in stealth, and shooting would no longer be allowed.

35.6.2 Balancing

After we enabled gun usage, suddenly Ellie was a killing machine. She did the same damage as the player and was pretty accurate. Fortunately, we had a lot of knobs to control her damage output: damage values, accuracy, firing rate, and so on.

We didn't want to change damage, because that makes an NPC feel broken. If it takes the player three shots to down an enemy, it should take the buddy the same number of shots (with some small margin of error). We also didn't want to change the fire rate or accuracy too much. That would make her face down an enemy with her gun pointing at it and shoot so infrequently or inefficiently that again she felt broken. We made minor changes to fire rate and accuracy to bring them to a believable limit on a per encounter basis, but the buddies were still far too effective.

In order to address this, we decided that if the player isn't going to see the hit, then it isn't necessary for a buddy to do any damage. This meant the player would hear a normal burst of gunshots, but the composition of the encounter didn't change. In practice, players rarely realized that the buddy wasn't doing any damage, so we were able to retain game balance without breaking the player's immersion in the game.

With that said, we didn't want Ellie to feel completely useless, so we tried to identify key moments when it would be good for her to hit and damage a target. In particular, if the player hasn't seen her shoot anyone recently, or if the player is in immediate danger from a charging enemy, low health, a grapple, or something similar, then this is a good time for Ellie's shots to start landing.

Despite all of these changes, we still found the need to lower her fire rate below "realistic" levels to balance damage, so we gave her a "furtive" idle animation that makes her look nervous when she wasn't shooting. This really helped pad out the gaps between shots, and the animation also helped to build her character.

35.6.3 Cheating

Throughout development, there was one prominent decision that we had to make: specifically, whether or not enemies could see Ellie and be alerted by her. For the majority of development, they were able to see her, and this drove us to make her as stealthy as possible, perfecting the AI as much as we could. In practice, it worked about 90%–95% of the time, and we were very proud of how close we got to perfect buddy stealth, but ultimately we had to make a decision one way or the other. Neither option was ideal: either buddies would occasionally give away the player's location, or enemies would never be able to see buddies unless they were actively in combat.

In the end, we decided that the only viable option was to make buddies invisible to enemy NPCs if the player was being stealthy. The result is that sometimes a buddy will run past a bad guy in clear view and not be seen, but as discussed, we had done a lot of work to make sure this wouldn't happen frequently. It breaks realism but considers the alternative. If Ellie was seen even once and gave away the player's location, then the bond between them would become fractured, and we wanted to avoid that at all costs.

35.7 Finishing Touches

Having a robust, well balanced, and ultimately fun buddy AI up and running, it was finally time to add the little touches that would really make Ellie and the other buddies come alive. It's important to recognize that much of what makes a character really deep and compelling has nothing to do with AI; it is having a large library of content to draw from.

35.7.1 Vocalizations

We realized very early on that there needed to be close integration between the combat AI and the dialogue system. This enabled us to easily have a buddy comment on specifics of the recent combat, such as the types of kill she had witnessed or if she had saved you during the fight.

Completely by coincidence, we noticed that Ellie's vocalizations would frequently mirror what the player was exclaiming. After a particularly gruesome battle, it wouldn't be uncommon for the player to utter profanities, followed shortly by Ellie playing an identical vocalization. This made an entirely unplanned connection between the player and Ellie.

35.7.2 Callouts

Next, we added callouts of unseen threats, either whispered or yelled depending on the severity of the situation. This system tries to model the player's view of the world and keep track of threats that they aren't aware of, and will then have a buddy comment on the threat at an appropriate time.

One very important thing we discovered when testing this new callout system is that it is absolutely imperative to ensure that whenever a buddy calls out an enemy, the enemy is there for the player to see. We were a little overzealous with our callouts originally, and Ellie would call out someone who she had only just caught a glimpse of or who may have ducked behind a wall again. When the player heard her warn about an enemy that they were unable to spot, it reflected badly on Ellie's intelligence, so we iterated hard to eliminate bad callouts. This is a classic example of a situation where the requirements for AI are actually more stringent than they would be for human intelligence.

When the system was working and tuned, it really helped give the player a better view of the world. We had a pretty minimal HUD—no minimap in the corner showing enemy locations—so any additional information you could get about the encounter was invaluable. Ellie was acting as an information source for the player, and it really improved the bond.

35.7.3 Ambience

Huge portions of the game have no combat, and we needed Ellie to be interesting and alive during these sections. There are lots of scripted dialogue interactions with the player, of which some are explicitly triggered through the controller and others are triggered dynamically after combat encounters.

We also created a suite of idle animations for her to play, things as simple as cleaning her knife, tying her shoelace, or straightening her hair. Fortunately, it was really easy for animators to get content like this into the game.

Finally, we added an "explore" system for buddies where designers would instrument the world with points of interest and cinematic interactions for her to use. The designers loved this and very quickly made the open spaces a lot livelier with buddies searching them, looking in cabinets, and so on. It's a really basic system, it only took a day or so to implement, but it felt great going into a space and having Ellie start wandering off looking at things. She shared in the player's wonderment at this abandoned world. We got a lot of good feedback about the sections where Ellie would explore, and designers loved having this tool to flesh out their spaces.

35.8 Conclusion

After all of this, buddy AI in *The Last of Us* was a complete success and really enhanced the game. Our hard work paid off and strengthened the bond between Ellie and the player, and the half dozen additional buddy characters that the player encounters throughout the game are unique and believable entities in the world.

Just 5 months of development from a very small team—just one core engineer with backup from the rest of the character engineering team—meant that we had to have laser focus on what we needed to accomplish. We learned that the key to good buddy AI development and AI development, in general, does not lie in the complexity of the systems driving it but in the nuances of the character performance.

Getting into the right mindset was absolutely key for this. It sounds clichéd, but from day one, we tried to approach Ellie's character, and all the other buddies, as living, breathing characters in a world we were creating. Trying to empathize with their situation and reason about how they would operate in this world was central to everything we did.

The core of Ellie's AI is a robust and believable following system. Even before Ellie was able to shoot or do anything in combat, she felt great just by moving intelligently, taking cover in good locations, and staying out of the player's way. After that was in place, we did everything possible to eliminate the traditional annoyances associated with buddy characters, coupled with giving her agency in the world through her combat abilities.

The majority of the development time was spent on iteration, rather than building new systems. We had an almost manic obsession with fine-tuning the details of follow positioning, firing parameters, and ambient dialogue triggers. Having the tools available to iterate on and debug AI is absolutely essential, and it enabled prototyping new behaviors and tweaking things as we were watching them.

Another key to success was to put the buddies' combat abilities behind really big timers. In a long game, you don't want to tire the player out with witty banter, overzealous gifting, or special moves. If Ellie saved you from a melee grapple just once during the game, you would remember that moment fondly and talk about it. If she gave you a health pack around every corner, you wouldn't think of her as helpful, but it would become just another pickup, devaluing the act and her presence.

Finally, making the decision that a buddy would never break stealth was an extremely important one to make. Ultimately, we are making a game here, and your buddy AI can be as intelligent or realistic as possible, but if they ruin what the player is trying to do just once, it's all in vain.