Introduction

Horizon Zero Dawn (HZD) is a third-person, open-world role-playing game set in a post-apocalyptic world where players take on the role of Aloy, a brave and skilled hunter who seeks to uncover mysteries behind an event that wiped out old civilisation and unleased deadly robots.

AI plays a central role in gameplay. HZD features several AI systems, notably the Navigation Mesh system for pathfinding, Hierarchical task network planner, and Behaviour trees to control NPCs. Additionally, HZD also uses the ML algorithm (inverse kinematics) to create realistic animations for creatures.

Gorilla games had minimal experience with open-world games and creatures. Their previous work, mainly the Killzone series, was a cover-based corridor shooter where the characters were only humanoids who would occasionally move from cover to cover. As a result, gorilla games faced considerable challenges with modifying their previous work to HZD creatures and with animation fluidity (foot sliding) and pathfinding/ dynamic avoidance.

AI Systems

Machines:

Twenty-eight unique types of machines can be found in the wild, each with unique passive and aggressive behaviours.

1. Acquisition and Scavenger machines:

   These machines are worker-class machines found typically in open grasslands working. They are relatively docile and prefer to flee combat, but they can engage in combat if they are alone.

2. Recon Machines:

   Recons scan the area for threats and alert the herd so that, depending upon their behaviours, they can flee or engage in combat.

3. Transport Machines:

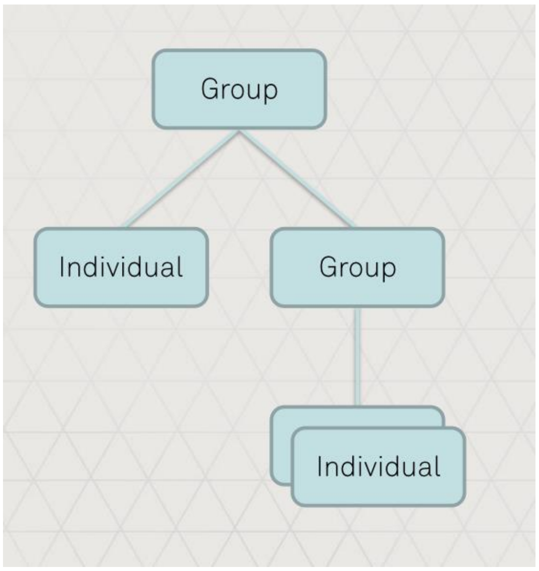
   These machines are slower and typically defended by small and more nimble combat machines. They gather resources from scavenger machines to transport them.

4. Combat Machines:

   These machines primarily attack the player and nearby threats, such as NPCs, during quests.

   Many ai systems dictate the machine’s behaviour, how they behave in the herd, and how the different herds are placed in the open world.

Hierarchy:



Horizon zero dawn adopts a hierarchy where a machine can be considered an agent on its own or as a child of a group agent (herd). Group agents are used for storing information about a particular herd. A group agent is just used to coordinate the behaviours within its children and does not physically exist in the game world.

Each agent has its own sensors to detect nearby threats, hear noises, and spot the player. But the group agent also has a blackboard system that stores in-game information, such as local patrols, nearby disturbances, enemy locations, etc., that machines may want to use. This allows agents in the group to share information with each other. This will enable scenarios such as when a recon robot spots Aloy and alerts other units that the acquisition robots might flee, and nearby combat units engage the player in combat.

Hierarchical task network planner:

A hierarchical task network (HTN) planner is the system used by HZD. HTN generates plans that comprise action macros, and each macro contains actions in a particular sequence. For example,

Check -Disturbance (macro)	Start Petrol (macro)
Heard-Disturbance Move-To-Location Investigate-disturbance	Get-Valid-Patrol Move-To-Location Idle-Wait Move-To-Location Idle-Wait

Guerrilla games used this system in the Killzone series, and HZD uses a more mature version of HTN planning.

Individual agents can ask the system plans to resolve a particular task, such as attacking the player, moving to a new area, or fleeing. On the other hand, group agents run plans that dictate what goals each individual agent should achieve/work on. This also allows agents of the group to actively share information with each other within a hierarchy and potentially form new subgroups or disband existing ones.

Each individual agent has its own behaviours, but the group agent controls what role each agent should play within the herd to support the herd archive its collective goal.

The Collective:

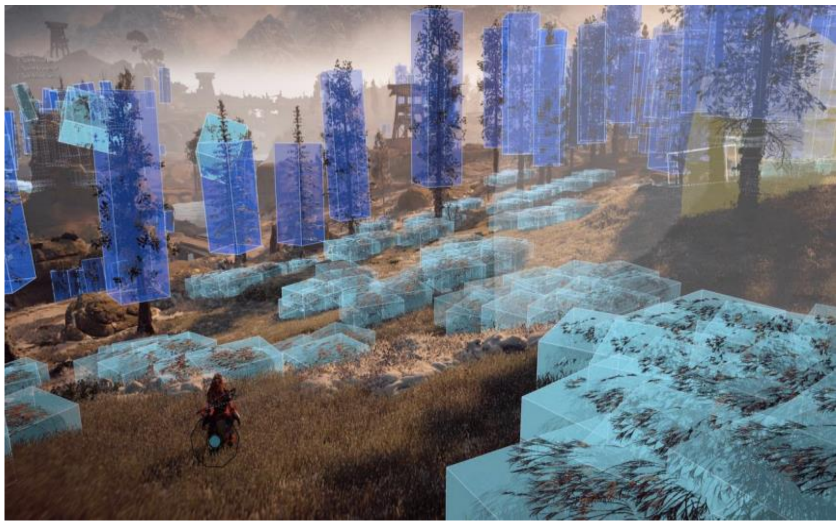
The collective is the supergroup. All group agents and individual agents are part of the collective. It manages the spawn of all machines in the world. It also keeps track of if the individuals are part of a group and handles move movement of individuals between the groups. It manages the machine system such that the game's performance and immersion do not suffer. If a machine has survived its previous group being destroyed or fled, it can request the collective to join a new group so the collective can recycle stray machines to form/ join new groups. To find if the machine fits another group, the collective uses information added to the machine, its passport. It has its level and type stored in it, and the collective checks if the machine fits the requirements of another group and only moves the machine to a new group if it satisfies the group's needs.

All herds are placed into the world in a relaxed state. A herd is usually a combination of recon, acquisition, and combat units where acquisition robots are in the centre doing their thing. In contrast, combat (to defend in case of attack) and recon units patrol the area. When the herd is created, several groups will be created, and each of the groups and individuals requests their roles and plans within the hierarchy. These roles assign the machine's behaviours and objectives. Roles can shift over time, and game events force changes in herds' behaviour and structure. There is a limit on how many agents of each type can fill in the spots in the herd so to not to disturb the balance of the herd. Machines' roles dictate their goals. For example, with a relaxed heard acquisition, machines use an HTN planner to get a plan to find a local area to mine while combat and recon units will generate petrol paths. These paths are automatically generated based on local terrain. They are generated in a way that they avoid uneven terrain and areas such as vegetation where the player might be hiding, for example, long grass. But patrol paths pass near stealth areas so that the player has opportunities to perform stealth attacks on machines.

When the herd is attacked, the machine's behaviour will change only a few machines will choose to run away while others attack the player. Even scavenging machines engage in combat if they are attacked on their own. When the heard is attacked and alerted that the information about the attack is passed to all the individuals in the heard at this point, the hierarchy in the heard will change. Individuals request a new role and priority for execution for the corresponding behaviour as well as the HTN planner reassigns which groups exist and assigns new knowledge shared and roles. For example, a new group of acquisition machines will run away together, and new groups of recon and combat machines attacking each character attacking them (for some missions, there are NPCs with you to take part in combat, not just the player)

The AI combat system attempts to structure combat so it is not random or downright broken. The different combat groups formed will be balanced and have different priorities and roles in attacking the player. For example, larger machines will be spread out with groups with smaller supporting machines. Groups select machines to attack the player not only if the HTN planner considers it can execute a specific attacking behaviour but also on whether the attack will be engaging or not. This avoids situations like all machines attacking the player at once, resulting certainly in player death/ unwinnable fight for the player. This factor uses an action selection utility function that calculates how "interesting" that machine's attack will be based on the current machine's state, whether the player has visibility of that machine/player's awareness of that machine. The distance between the player and the machine and the amount of damage received and given by that machine. This becomes a precondition for the HTN planner; more importantly, when the HTN planner provides the behaviour to attack the player to one machine, the other machines are given behaviours to stalk the player, which allows players to exploit weaknesses in the groups.

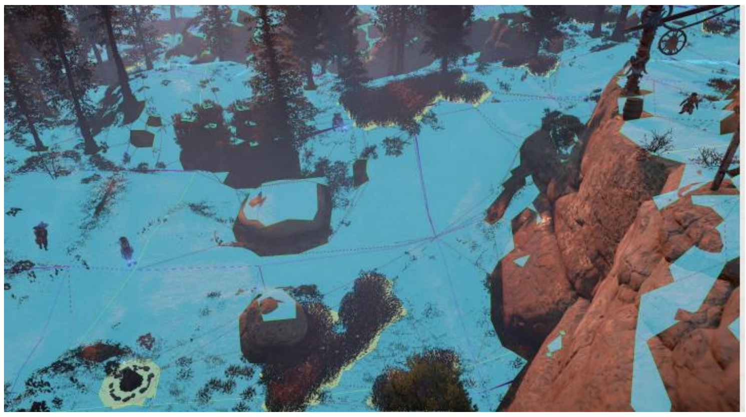
Sensors:



Each machine has its own set of sensors, for example, visual sensors (watches' eyes), radar and proximity sensors (long legs), audio sensors, colliders and also smell sensors (rarely used). These sensors are calibrated for each machine in a different way. This is done so that, for example, to make it easy to sneak attack a watcher but make it harder to sneak attack a stalker. Sensors in HZD are a lot more nuanced in HZD. This is done through information packets that are attached to objects themselves that can activate or stimulate one or more of the machine's sensors. The machine can receive information from these packets, like what it detected etc. this allows machines to differentiate between and detect different objects and tell what those objects are. But this system also ensures that the player in stealth (hiding in long grass or behind obstacles) cannot be spotted as those packets do not send information to the machine's sensors. Each character responds differently to data received by the sensors. Some characters may downright ignore some data, which allows for breathing in unique personalities in those characters.

Navigation:

Ground-based navigation.



HZD uses runtime dynamic navigation meshes for the movement/navigation of characters/creatures. HZD uses up to 6 nav meshes. One mesh for each small, medium, and large machine, one nav mesh that can generate over water for machines that can swim, one for human-type characters and one for the player mounts. These meshes differ by their generation settings.

These meshes are only generated when the character requests the nav mesh be built in the area and where they are the centre or surrounds the area they are limited to. This nav mesh is created in the background and continuously updated when changes are detected in that area or nav mesh objects. These objects are obstacles or stealth areas etc. These obstacles are limited to the character's own behaviour as one object (large stones) that is perceived as an obstacle by one character (watcher goes around) may not be considered by another character (angry thunder jaw who charges through the stone and breaks it into smaller rocks).

Navigation behaviours are also dependent on a machine’s current state for example, long grass is avoided when the machine is in a relaxed state but when the machine is in investigating state/behaviour it can walk through long grass.

Flaying navigation:

Flying machines need to know how to fly through the air in safe places where they can land and attack players. For this ,the flying machines need to know information about obstacles such as hills, trees, etc.. For this, HZD has a separate navigation system in the air.

This air navigation uses a technique called Hierarchical Path Planning Over Mipmaps. Mipmaps provide a collection of the same images gradually lowering in resolution. This approach is used because the machine flying does not need to know about the area far from it, but it needs to know the layout of the land around its current location in case it has to land. Path planning system for aerial paths uses MIP mapping for the height of local map geometry using four levels, becoming more complicated as they move down in levels. Level 3 is an abstract rough map, whereas level 0 is the accurate height map. These maps are built at runtime.

When a machine needs to flay to a position the path planner plans a path on the highest level of MIPMAP using A* algorithm, this is the simplest path. A* here has a limit on its iterations. Then it takes the section of the path and goes down in levels and calculates the path in lower levels which makes the path more accurate and helps to navigate through uneven geometry. Then the path is smoothed.

The machine’s velocity is tied to its animations, so when a machine is hovering over a player (for combat) it has the path stored in its memory, but it’s playing the howe animation, so it doesn’t move.

For landing the system communicates between the ground nav mesh and the air navigation system and chooses a safe place to land, typically a higher area once on the ground, it uses one of the 6 ground nav meshes for navigation depending upon its size.

This approach has one limitation the machines can’t fly under bridges.

Dynamic Obstacle Avoidance:

HZD uses velocity Obstacles for dynamic obstacle avoidance. To reduce the cost of this, HZD only avoids five most relevant obstacles at a time. These are chosen on the basis of the lowest time till collision. For building the velocity obstacles, shapes of both characters are combined by Malikowski addition into one polygon, which is used as input to create velocity obstacle. Then a set of velocities that fall outside these velocity obstacles are generated and scored using a scoring function that considers numerous factors. After that, the best target velocity is chosen and is used as input to the path’s moving solution if the character needs to avoid and after avoidance, the character returns to its old smooth path. This method is not suitable for humanoid characters as it looks robotic.

Path Following / Bezier path smoothing:

To make the path following smoother and to reduce issues like foot sliding HZD used Bezier path smoothing which this moves the path tangent continuously. HZD uses cubic Beziers which consist of

4 control points, First and last points dictate the start and end of the curve respectively the and other two control tangency along the curve (tangent handles). A curve is placed in each path segment, tangent handles are aligned with the previous and next tangent handles. This gives a smooth continuous path.in game tangent placement was placed on avoidance detection or character velocity, path polyline and optional modifies that change length of the first tangent depending upon NPC's size.

To make sure the path is safe to traverse the tangent handles are clamped to the nav mesh and check if the line between the handles is also unobstructed. This in turn created a safe hull on nave mesh in which the resulting trajectory was also safe.

Some issues arise from this. One of them is smoothened path that deviated largely from the original point path and as it deviated a lot it became more likely that it would fall outside of the navigation safe space. If the line between to tangent handles is not clear, then the character will be forced to stop and make a quick turn or start again. Another issue is that large characters could not make smooth turns and end up with foot sliding.

To fix this Gorilla games used a new solution using Euler Spirals and a string-pulling algorithm, but this did not make it into Horizon Zero Dawn as it was not finished before the launch of HZD.

Locomotion sampling/motion table:

AI in HZD needed information about animation and its locomotion capabilities.

Devs ran Animations through the offline content conversion process. Then they activated and sampled all relevant animation blend trees. Then animation data such as Root bone transformations, animation duration and metadata were captured. This data is then stored in a table called a motion table.

Creating a motion table works as rotate a given animation blend tree and create a tree-like data structure where each layer represents a used animation variable and each branch following the layer is a singular value used by a blend node in a animation blend tree. Then the motion table can be queried at runtime. This system makes the character ai more aware what animations it has available at a movement.

Stop Prediction:

HZD uses motion tables to estimate where to stop the character. The motion table is used to check how much the character will overshoot or undershoot and if the nav mesh allows it. If yes, then the stop animation is played without alterations. This makes stopping more natural.

Attack system:

Motion table is also used a lot in the melee system to look up different properties related to attack animations.

Using different conditions machines decide which attack to use most importantly if the attack is going to hit this is done using a trigger volume placed around the character where it will do damage when the target is inside this volume the attack is triggered. This is done using motion table values for the velocity of the attack and the time it will take to reach that point this avoids scenarios where the attack overshoots the target and lands behind the target then the attack winds up to allow adjusting of the character, and then the attack is performed.

Conclusion

The AI systems are adapted to be used with characters of different sizes and shapes. The system using different nav meshes for different character sizes allows other characters to show their own movement patterns which in turn makes the game more immersive.

The system mostly satisfies the goals set by the developers while transitioning from the old Killzone navigation system to the new HZD one except one that is the foot sliding issue on large characters while turning.

This system performs far better than other systems used in open-world games. Other games don't have different paths for different sizes of characters which makes them feel unnatural.

Although there are some improvements to be made such as more natural turns, variety in combat animation and the problem with characters performing quick turns while following an S-shaped path.

References:

[1] Julian Berteling, Beyond Killzone: Creating New AI Systems for Horizon Zero Dawn, GDC, 2018.

[2] Arjen Beij, The AI of Horizon Zero Dawn, Game AI North, 2017. Arjen Beij, The AI of Horizon Zero Dawn, Game AI North, 2017.

[3] Wouter Josemans, 2017. "Putting the AI back into Air: Navigating the Air Space of Horizon Zero Dawn", Game AI North 2017

[4] Tommy Thompson, Behind The AI of Horizon Zero Dawn (Part 1 and part 2), Game Developer, 2019.