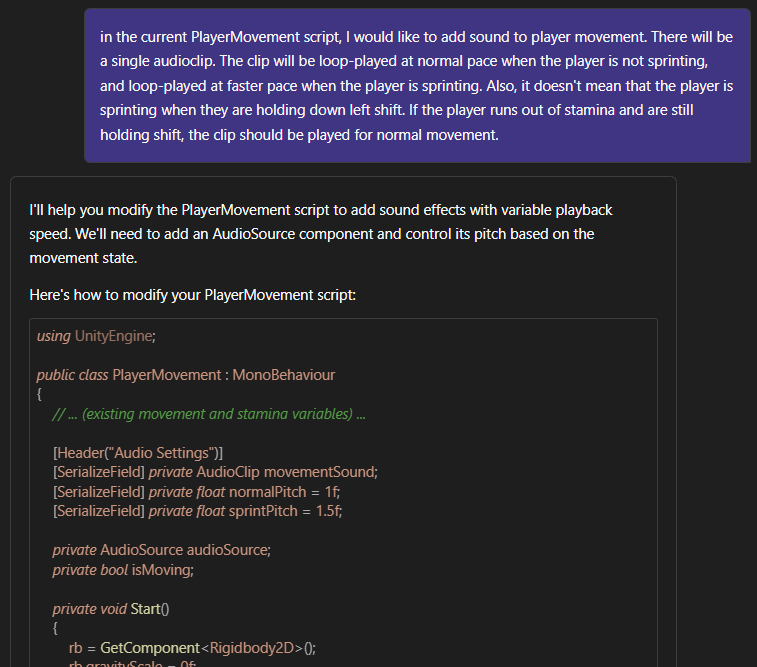
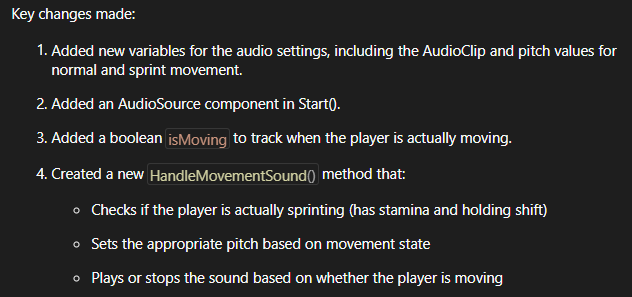
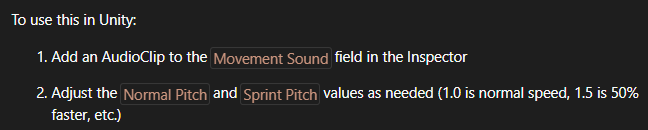
# PLAYER MOVEMENT SOUND UPDATE

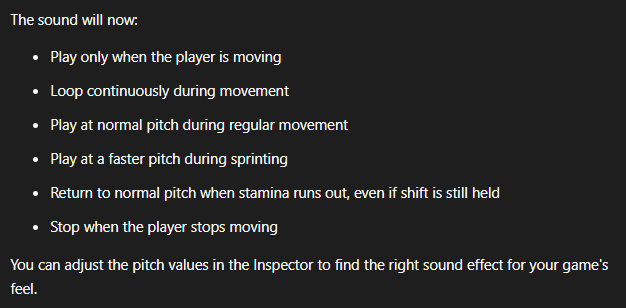


Prompt:

in the current PlayerMovement script, I would like to add sound to player movement. There will be a single audioclip. The clip will be loop-played at normal pace when the player is not sprinting, and loop-played at faster pace when the player is sprinting. Also, it doesn't mean that the player is sprinting when they are holding down left shift. If the player runs out of stamina and are still holding shift, the clip should be played for normal movement.

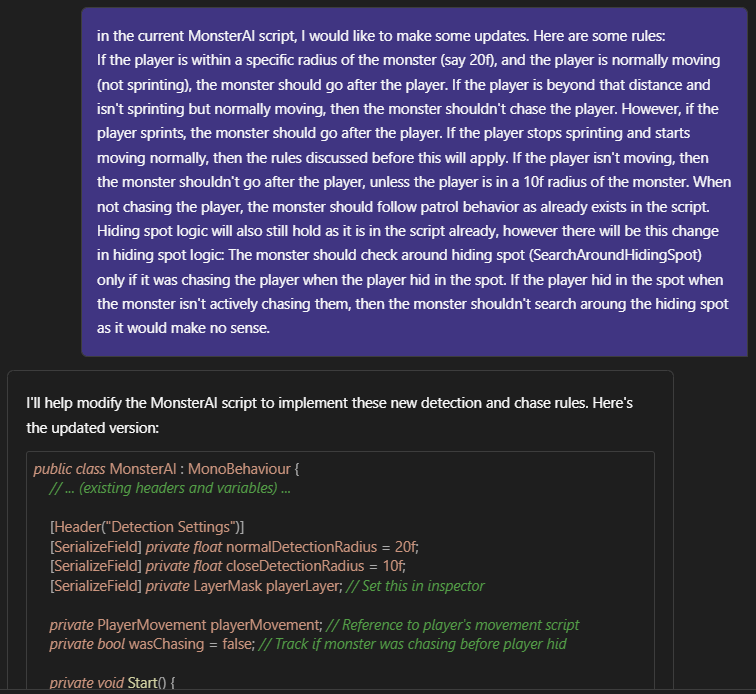


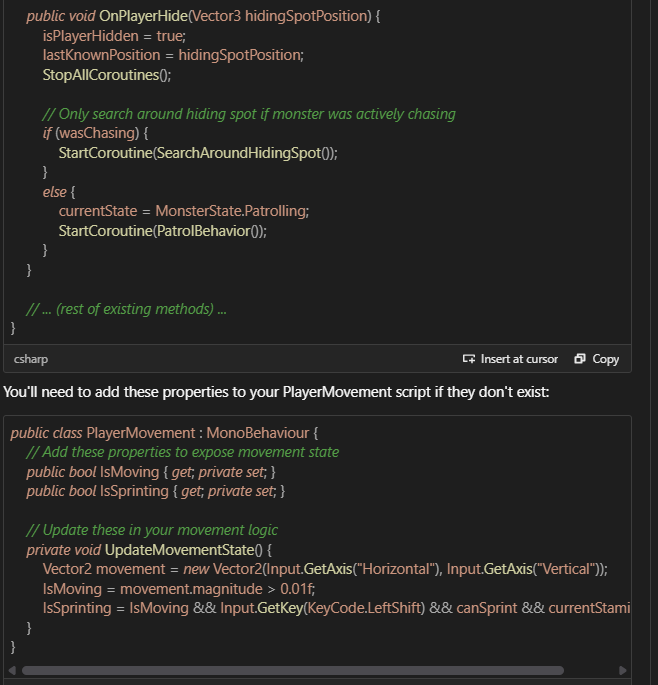


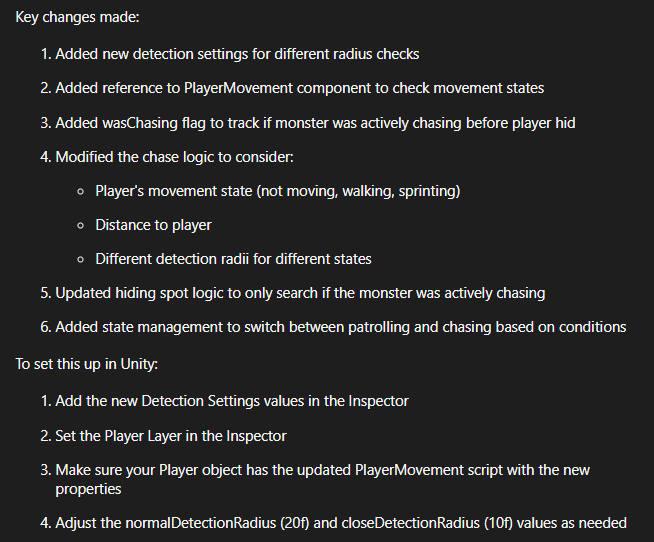


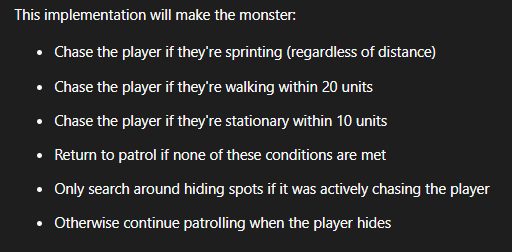
I added another audio source for this because there the existing one was in use

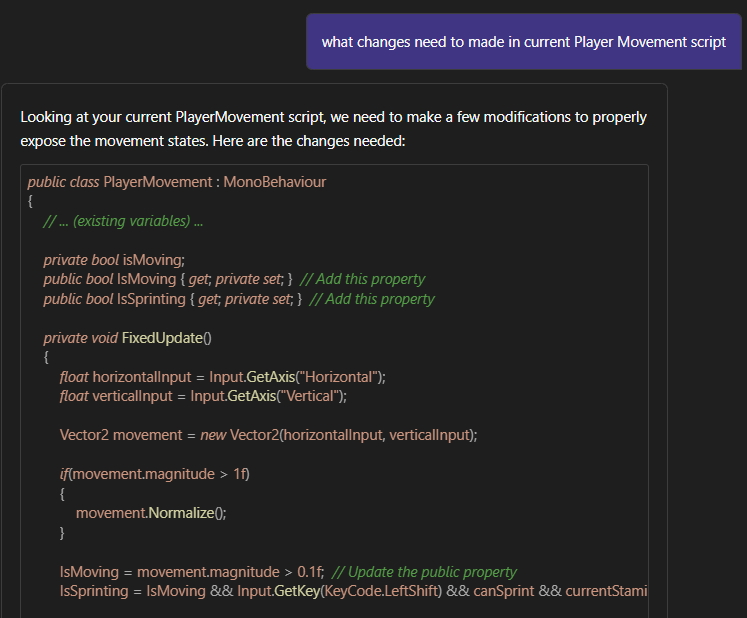
# MONSTER CHASE LOGIC UPDATE

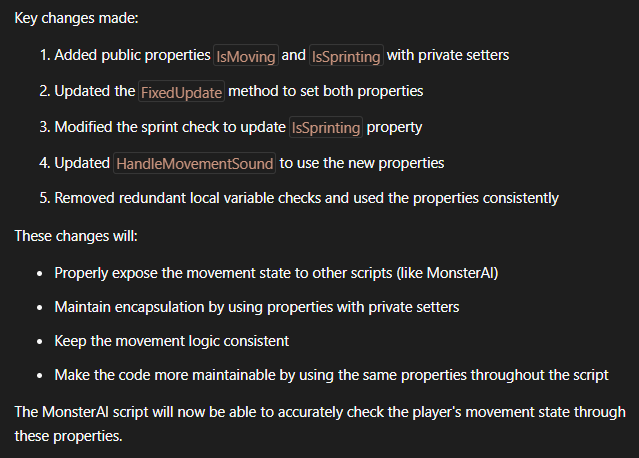


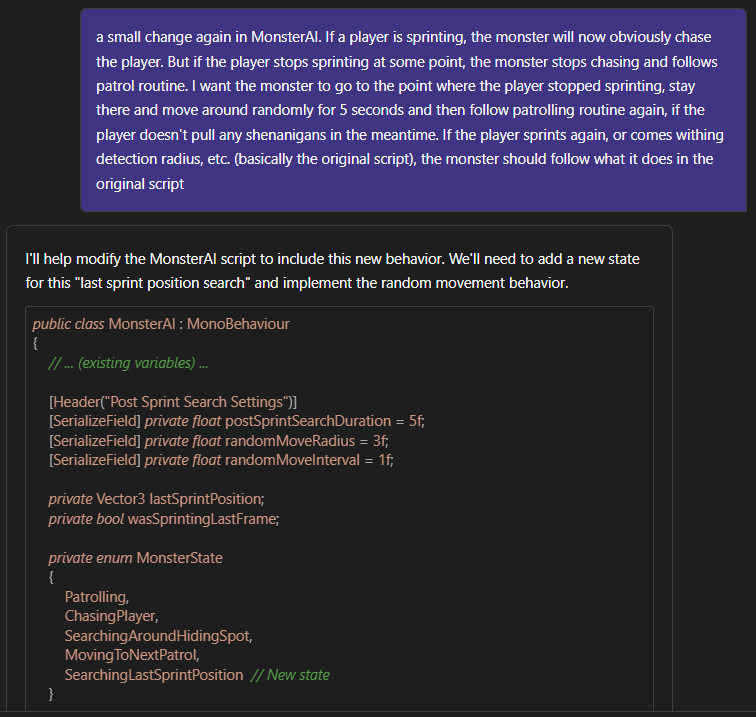












public class MonsterAI : MonoBehaviour

{

// ... (existing variables) ...

[Header("Post Sprint Search Settings")]

[SerializeField] private float postSprintSearchDuration = 5f;

[SerializeField] private float randomMoveRadius = 3f;

[SerializeField] private float randomMoveInterval = 1f;

private Vector3 lastSprintPosition;

private bool wasSprintingLastFrame;

private enum MonsterState

{

Patrolling,

ChasingPlayer,

SearchingAroundHidingSpot,

MovingToNextPatrol,

SearchingLastSprintPosition // New state

}

private void CheckPlayerState()

{

if (isPlayerHidden) return;

float distanceToPlayer = Vector2.Distance(transform.position, playerTransform.position);

bool shouldChase = false;

bool isPlayerMoving = playerMovement.IsMoving;

bool isPlayerSprinting = playerMovement.IsSprinting;

// Check if player just stopped sprinting

if (wasSprintingLastFrame && !isPlayerSprinting)

{

lastSprintPosition = playerTransform.position;

StartCoroutine(SearchLastSprintPosition());

wasSprintingLastFrame = false;

return;

}

// Update sprinting state

wasSprintingLastFrame = isPlayerSprinting;

if (!isPlayerMoving)

{

shouldChase = distanceToPlayer <= closeDetectionRadius;

}

else if (isPlayerSprinting)

{

shouldChase = true;

}

else

{

shouldChase = distanceToPlayer <= normalDetectionRadius;

}

if(shouldChase && currentState != MonsterState.ChasingPlayer)

{

wasChasing = true;

currentState = MonsterState.ChasingPlayer;

StopAllCoroutines();

UpdatePath();

}

else if (!shouldChase && currentState == MonsterState.ChasingPlayer)

{

wasChasing = false;

if (!wasSprintingLastFrame) // Only start patrolling if we're not handling sprint stop

{

currentState = MonsterState.Patrolling;

StartCoroutine(PatrolBehavior());

}

}

if(currentState == MonsterState.ChasingPlayer)

{

pathUpdateTimer -= Time.deltaTime;

if(pathUpdateTimer <= 0)

{

UpdatePath();

pathUpdateTimer = nextPathUpdateTime;

}

}

}

private IEnumerator SearchLastSprintPosition()

{

currentState = MonsterState.SearchingLastSprintPosition;

// First, move to the last sprint position

agent.SetDestination(lastSprintPosition);

// Wait until we reach the position or get close enough

while (Vector3.Distance(transform.position, lastSprintPosition) > agent.stoppingDistance + 0.1f)

{

// Check if we should interrupt the search

if (ShouldInterruptSearch())

{

yield break;

}

yield return null;

}

float searchStartTime = Time.time;

float nextMoveTime = 0f;

// Random movement for specified duration

while (Time.time - searchStartTime < postSprintSearchDuration)

{

// Check if we should interrupt the search

if (ShouldInterruptSearch())

{

yield break;

}

// Generate new random position periodically

if (Time.time >= nextMoveTime)

{

Vector3 randomPosition = GetRandomPositionAroundPoint(lastSprintPosition, randomMoveRadius);

agent.SetDestination(randomPosition);

nextMoveTime = Time.time + randomMoveInterval;

}

yield return null;

}

// After search duration, return to patrol if no other conditions are met

if (currentState == MonsterState.SearchingLastSprintPosition)

{

currentState = MonsterState.Patrolling;

StartCoroutine(PatrolBehavior());

}

}

private bool ShouldInterruptSearch()

{

if (isPlayerHidden) return false;

float distanceToPlayer = Vector2.Distance(transform.position, playerTransform.position);

// Interrupt if player is sprinting

if (playerMovement.IsSprinting) return true;

// Interrupt if player is moving and within normal detection radius

if (playerMovement.IsMoving && distanceToPlayer <= normalDetectionRadius) return true;

// Interrupt if player is within close detection radius

if (distanceToPlayer <= closeDetectionRadius) return true;

return false;

}

private Vector3 GetRandomPositionAroundPoint(Vector3 center, float radius)

{

for (int i = 0; i < 30; i++) // Multiple attempts to find valid position

{

Vector2 randomCircle = Random.insideUnitCircle \* radius;

Vector3 randomPoint = center + new Vector3(randomCircle.x, randomCircle.y, 0);

NavMeshHit hit;

if (NavMesh.SamplePosition(randomPoint, out hit, 1.0f, NavMesh.AllAreas))

{

return hit.position;

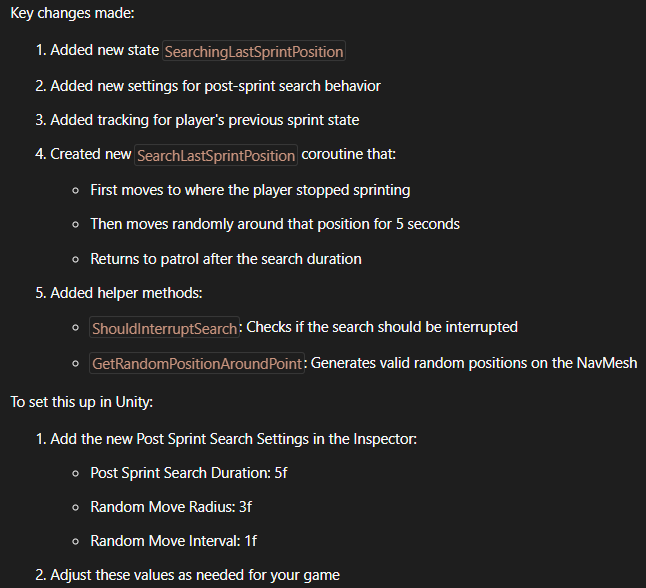
}

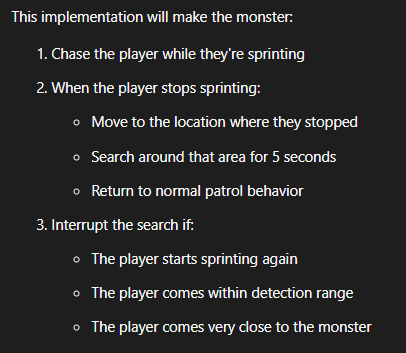
}

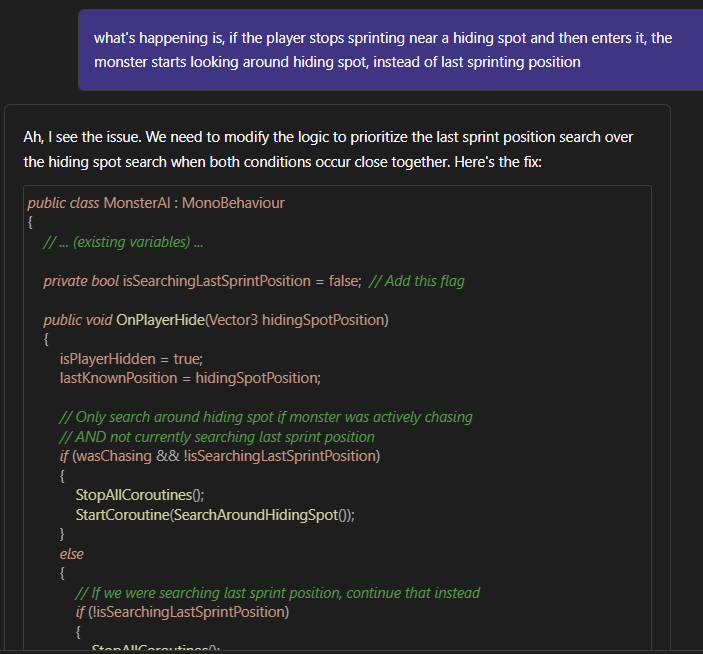
return center; // Return center if no valid position found

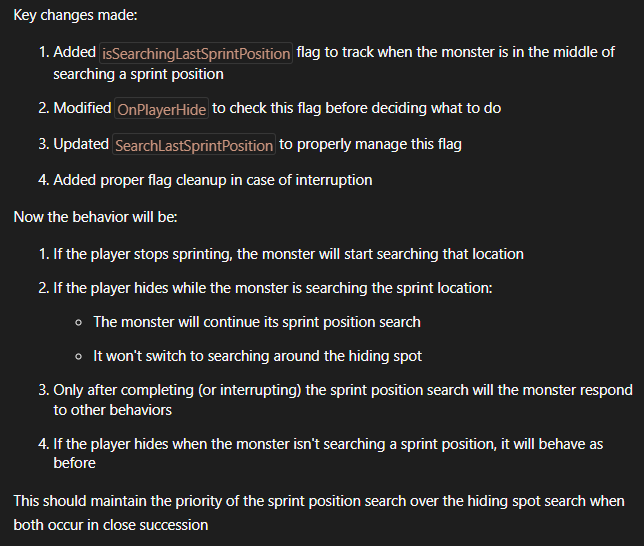
}

}









Prompts:

in the current MonsterAI script, I would like to make some updates. Here are some rules:  
If the player is within a specific radius of the monster (say 20f), and the player is normally moving (not sprinting), the monster should go after the player. If the player is beyond that distance and isn't sprinting but normally moving, then the monster shouldn't chase the player. However, if the player sprints, the monster should go after the player. If the player stops sprinting and starts moving normally, then the rules discussed before this will apply. If the player isn't moving, then the monster shouldn't go after the player, unless the player is in a 10f radius of the monster. When not chasing the player, the monster should follow patrol behavior as already exists in the script. Hiding spot logic will also still hold as it is in the script already, however there will be this change in hiding spot logic: The monster should check around hiding spot (SearchAroundHidingSpot) only if it was chasing the player when the player hid in the spot. If the player hid in the spot when the monster isn't actively chasing them, then the monster shouldn't search aroung the hiding spot as it would make no sense.

what changes need to made in current Player Movement script

a small change again in MonsterAI. If a player is sprinting, the monster will now obviously chase the player. But if the player stops sprinting at some point, the monster stops chasing and follows patrol routine. I want the monster to go to the point where the player stopped sprinting, stay there and move around randomly for 5 seconds and then follow patrolling routine again, if the player doesn't pull any shenanigans in the meantime. If the player sprints again, or comes withing detection radius, etc. (basically the original script), the monster should follow what it does in the original script

what's happening is, if the player stops sprinting near a hiding spot and then enters it, the monster starts looking around hiding spot, instead of last sprinting position