using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerFootsteps : MonoBehaviour

{

private AudioSource footstep\_Sound;

[SerializeField]

private AudioClip[] footstep\_Clip;

private CharacterController character\_Controller;

[HideInInspector]

public float volume\_Min,volume\_Max;

private float accumulated\_Distance;

[HideInInspector]

public float step\_Distance;

void Awake()

{

footstep\_Sound = GetComponent<AudioSource>();

character\_Controller = GetComponentInParent<CharacterController>();

}

// Update is called once per frame

void Update()

{

CheckToPlayFootstepSound();

}

void CheckToPlayFootstepSound()

{

//if we not on the ground

if (!character\_Controller.isGrounded)

return;

//accumulated distance iss the value howw far can we go

//e.g make a step or spint, or move while crouching

//until we play the footstep sound

if (character\_Controller.velocity.sqrMagnitude > 0)

{

accumulated\_Distance += Time.deltaTime;

if (accumulated\_Distance > step\_Distance)

{

footstep\_Sound.volume = Random.Range(volume\_Min, volume\_Max);

footstep\_Sound.clip = footstep\_Clip[Random.Range(0, footstep\_Clip.Length)];

footstep\_Sound.Play();

accumulated\_Distance = 0f;

// Random.Range(1, 5);

}

}

else

{

accumulated\_Distance = 0f;

}

}

}//class

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerFootsteps : MonoBehaviour

{

private AudioSource footstep\_Sound;

[SerializeField]

private AudioClip[] footstep\_Clip;

private CharacterController character\_Controller;

[HideInInspector]

public float volume\_Min,volume\_Max;

private float accumulated\_Distance;

[HideInInspector]

public float step\_Distance;

Phần code này là một script cho thành phần PlayerFootsteps trong một trò chơi Unity, dùng để xử lý âm thanh tiếng bước chân khi người chơi di chuyển.

 **footstep\_Sound**: Biến AudioSource này được sử dụng để phát âm thanh tiếng bước chân. Nó có thể được gắn vào cùng GameObject với script này hoặc một GameObject khác liên quan.

 **footstep\_Clip**: Mảng AudioClip này lưu trữ các đoạn âm thanh bước chân khác nhau. Điều này cho phép đa dạng hóa âm thanh được phát ra khi người chơi bước đi, làm cho trải nghiệm trở nên thực tế hơn.

 **character\_Controller**: Biến CharacterController này tham chiếu đến thành phần điều khiển nhân vật được gắn vào người chơi. Nó được sử dụng để kiểm tra trạng thái di chuyển của người chơi để quyết định khi nào phát âm thanh bước chân.

 **volume\_Min, volume\_Max**: Các biến float này đặt mức âm lượng tối thiểu và tối đa cho âm thanh bước chân. Bằng cách thay đổi âm lượng, tiếng bước chân có thể nghe động và tự nhiên hơn.

 **accumulated\_Distance**: Biến float này theo dõi khoảng cách mà người chơi đã di chuyển. Khi khoảng cách này vượt quá một ngưỡng nhất định (được định nghĩa bởi step\_Distance), âm thanh bước chân sẽ được phát và accumulated\_Distance được đặt lại.

 **step\_Distance**: Biến float này định nghĩa khoảng cách mà người chơi phải di chuyển trước khi một âm thanh bước chân mới được phát. Nó xác định tần suất của âm thanh bước chân dựa trên sự di chuyển của người chơi.

Kết luận:

các biến này hoạt động cùng nhau để quản lý thời điểm và cách thức phát ra âm thanh bước chân khi người chơi di chuyển qua môi trường trong trò chơi. Cơ chế dựa trên khoảng cách đảm bảo rằng tiếng bước chân được kích hoạt ở các khoảng cách phù hợp, trong khi sự đa dạng về âm lượng và âm thanh làm tăng tính thực tế cho trải nghiệm thính giác.

void Awake()

{

footstep\_Sound = GetComponent<AudioSource>();

character\_Controller = GetComponentInParent<CharacterController>();

}

 **void Awake()**: Đây là chữ ký của phương thức Awake. Phương thức Awake được gọi khi đối tượng script được tải, điều này xảy ra trước bất kỳ phương thức Start nào và trước khi trò chơi bắt đầu chạy.

 **footstep\_Sound = GetComponent<AudioSource>();**: Dòng này gán thành phần AudioSource được đính kèm vào cùng GameObject với script này cho biến footstep\_Sound. GetComponent<AudioSource>() sẽ tìm kiếm thành phần AudioSource trên cùng GameObject và trả về nó.

 **character\_Controller = GetComponentInParent<CharacterController>();**: Dòng này gán thành phần CharacterController từ GameObject cha của đối tượng hiện tại cho biến character\_Controller. GetComponentInParent<CharacterController>() sẽ tìm kiếm thành phần CharacterController trong các GameObject cha của GameObject hiện tại và trả về cái đầu tiên nó tìm thấy.

Kết luận:

phương thức Awake khởi tạo các biến footstep\_Sound và character\_Controller bằng cách tìm kiếm và gán các thành phần liên quan từ GameObject và cha của nó. Thiết lập này rất quan trọng để đảm bảo rằng script có quyền truy cập vào các thành phần cần thiết để quản lý hiệu quả âm thanh bước chân.

void CheckToPlayFootstepSound()

{

//if we not on the ground

if (!character\_Controller.isGrounded)

return;

//accumulated distance iss the value howw far can we go

//e.g make a step or spint, or move while crouching

//until we play the footstep sound

if (character\_Controller.velocity.sqrMagnitude > 0)

{

accumulated\_Distance += Time.deltaTime;

if (accumulated\_Distance > step\_Distance)

{

footstep\_Sound.volume = Random.Range(volume\_Min, volume\_Max);

footstep\_Sound.clip = footstep\_Clip[Random.Range(0, footstep\_Clip.Length)];

footstep\_Sound.Play();

accumulated\_Distance = 0f;

// Random.Range(1, 5);

}

}

else

{

accumulated\_Distance = 0f;

}

}

}//class

 **if (!character\_Controller.isGrounded) return;**: Dòng này kiểm tra xem người chơi có đang ở trên mặt đất hay không. Nếu người chơi đang ở trên không (ví dụ: nhảy hoặc rơi), phương thức sẽ kết thúc ngay lập tức, có nghĩa là không có âm thanh bước chân nào được phát.

 **if (character\_Controller.velocity.sqrMagnitude > 0)**: Dòng này kiểm tra xem người chơi có đang di chuyển hay không. velocity.sqrMagnitude là bình phương của độ lớn của vector vận tốc, điều này hiệu quả hơn so với việc tính toán độ lớn thực tế. Nếu người chơi đang di chuyển (tức là vận tốc của họ lớn hơn không), mã lệnh sau sẽ được thực thi.

 **accumulated\_Distance += Time.deltaTime;**: Dòng này thêm thời gian từ khung hình trước đó vào accumulated\_Distance. Time.deltaTime đại diện cho thời gian tính bằng giây mà khung hình trước đó hoàn thành, hiệu quả là tích lũy khoảng cách theo thời gian.

 **if (accumulated\_Distance > step\_Distance)**: Dòng này kiểm tra xem khoảng cách tích lũy có vượt quá khoảng cách bước chân được định trước (step\_Distance) hay không. Nếu có, âm thanh bước chân sẽ được phát.

 **footstep\_Sound.volume = Random.Range(volume\_Min, volume\_Max);**: Dòng này đặt âm lượng của âm thanh bước chân thành một giá trị ngẫu nhiên trong khoảng từ volume\_Min đến volume\_Max, tạo sự đa dạng cho âm thanh.

 **footstep\_Sound.clip = footstep\_Clip[Random.Range(0, footstep\_Clip.Length)];**: Dòng này chọn một đoạn âm thanh bước chân ngẫu nhiên từ mảng footstep\_Clip.

 **footstep\_Sound.Play();**: Dòng này phát đoạn âm thanh bước chân đã chọn.

 **accumulated\_Distance = 0f;**: Dòng này đặt lại khoảng cách tích lũy về 0 sau khi âm thanh bước chân đã được phát.

 **else { accumulated\_Distance = 0f; }**: Nếu người chơi không di chuyển (velocity.sqrMagnitude bằng không), khoảng cách tích lũy được đặt lại về 0.

Kết luận:

phương thức CheckToPlayFootstepSound đảm bảo rằng âm thanh bước chân chỉ được phát khi người chơi đang ở trên mặt đất và đang di chuyển. Nó tính toán khi nào cần phát âm thanh dựa trên khoảng cách đã di chuyển, thêm tính ngẫu nhiên cho âm lượng và đoạn âm thanh để tạo sự đa dạng, và đặt lại khoảng cách sau khi phát mỗi âm thanh bước chân.