using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerSprintAndCrouch2 : MonoBehaviour

{

private PlayerMovement playerMovement;

public float sprint\_Speed = 10f;

public float move\_Speed = 4f;

public float crouch\_Speed = 2f;

private Transform look\_Root;

private float stand\_Height = 1.6f;

private float crouch\_Height = 1f;

private bool is\_Crouching;

private PlayerFootsteps player\_Footsteps;

private float sprint\_Volume = 1f;

private float crouch\_Volume = 0.1f;

private float walk\_Volume\_Min = 0.2f, walk\_Volume\_Max = 0.6f;

private float walk\_Step\_Distance = 0.4f;

private float sprint\_Step\_Distance = 0.25f;

private float crouch\_Step\_Distance = 0.5f;

private float sprint\_Value = 100f;

public float sprint\_Treshold = 10f;

private PlayerStats player\_Stats;

void Awake()

{

playerMovement = GetComponent<PlayerMovement>();

look\_Root = transform.GetChild(0);//chuyen lookroot thanh child cua player

player\_Footsteps = GetComponentInChildren<PlayerFootsteps>();

player\_Stats = GetComponent<PlayerStats>();

}

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

sprint();

crouch();

}

void sprint()

{

//if we have stamina we run

if (sprint\_Value > 0f)

{

if (Input.GetKeyDown(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = sprint\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

// sprint\_Value -=Time.deltaTime\*sprint\_Treshold;

}

}

if (Input.GetKeyUp(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

}

if(Input.GetKey(KeyCode.LeftShift) && !is\_Crouching)

{

sprint\_Value -=sprint\_Treshold\*Time.deltaTime;

if(sprint\_Value <= 0f)

{

sprint\_Value = 0f;

//reset the speed and sound

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min = walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

}

player\_Stats.Display\_StaminaStats(sprint\_Value);

}

else

{

if(sprint\_Value != 100f)

{

sprint\_Value += (sprint\_Treshold / 2f) \* Time.deltaTime;

player\_Stats.Display\_StaminaStats(sprint\_Value);

if (sprint\_Value > 100f)

{

sprint\_Value = 100f;

}

}

}

}

void crouch()

{

if (Input.GetKeyDown(KeyCode.LeftControl))

{

if (is\_Crouching)

{

look\_Root.localPosition = new Vector3(0f, stand\_Height, 0f);

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min =walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

is\_Crouching = false;

}

else

{

look\_Root.localPosition = new Vector3(0f, crouch\_Height, 0f);

playerMovement.speed= crouch\_Speed;

player\_Footsteps.step\_Distance = crouch\_Step\_Distance;

player\_Footsteps.volume\_Min = crouch\_Volume;

player\_Footsteps.volume\_Max = crouch\_Volume;

is\_Crouching = true;

}

}

}

}

public class PlayerSprintAndCrouch2 : MonoBehaviour

{

private PlayerMovement playerMovement;

public float sprint\_Speed = 10f;

public float move\_Speed = 4f;

public float crouch\_Speed = 2f;

private Transform look\_Root;

private float stand\_Height = 1.6f;

private float crouch\_Height = 1f;

private bool is\_Crouching;

private PlayerFootsteps player\_Footsteps;

private float sprint\_Volume = 1f;

private float crouch\_Volume = 0.1f;

private float walk\_Volume\_Min = 0.2f, walk\_Volume\_Max = 0.6f;

private float walk\_Step\_Distance = 0.4f;

private float sprint\_Step\_Distance = 0.25f;

private float crouch\_Step\_Distance = 0.5f;

private float sprint\_Value = 100f;

public float sprint\_Treshold = 10f;

private PlayerStats player\_Stats;

=>Gọi các biến sau để chuẩn bị cho các hàm sau

void Awake()

{

playerMovement = GetComponent<PlayerMovement>();

look\_Root = transform.GetChild(0);//chuyen lookroot thanh child cua player

player\_Footsteps = GetComponentInChildren<PlayerFootsteps>();

player\_Stats = GetComponent<PlayerStats>();

}

playerMovement = GetComponent<PlayerMovement>();=> dùng để kế thừa PlayerMovement.cs

look\_Root = transform.GetChild(0);=> dùng để chuyển hàm lookroot thành child của Player

player\_Footsteps = GetComponentInChildren<PlayerFootsteps>();

player\_Stats = GetComponent<PlayerStats>();🡺 dùng để nhận thông tin từ hàm PlayerStat.cs

void sprint()

{

//if we have stamina we run

if (sprint\_Value > 0f)

{

if (Input.GetKeyDown(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = sprint\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

// sprint\_Value -=Time.deltaTime\*sprint\_Treshold;

}

}

if (Input.GetKeyUp(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

}

if(Input.GetKey(KeyCode.LeftShift) && !is\_Crouching)

{

sprint\_Value -=sprint\_Treshold\*Time.deltaTime;

if(sprint\_Value <= 0f)

{

sprint\_Value = 0f;

//reset the speed and sound

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min = walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

}

player\_Stats.Display\_StaminaStats(sprint\_Value);

}

else

{

if(sprint\_Value != 100f)

{

sprint\_Value += (sprint\_Treshold / 2f) \* Time.deltaTime;

player\_Stats.Display\_StaminaStats(sprint\_Value);

if (sprint\_Value > 100f)

{

sprint\_Value = 100f;

}

}

}

}

Hàm sprint() trong đoạn mã này được sử dụng để quản lý hành vi chạy nước rút (sprint) của nhân vật trong trò chơi. Cụ thể, hàm này kiểm tra và điều chỉnh tốc độ di chuyển, âm thanh bước chân, và lượng stamina (thể lực) của nhân vật khi nhấn phím chạy nước rút (LeftShift)

Phân tích hàm sprint():

if (sprint\_Value > 0f)

Kiểm tra nếu nhân vật còn stamina (thể lực) để chạy nước rút.

if (Input.GetKeyDown(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = sprint\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

// sprint\_Value -=Time.deltaTime\*sprint\_Treshold;

}

}

 Khi người chơi nhấn phím LeftShift và nhân vật không đang cúi người (is\_Crouching), thiết lập các giá trị sau:

* Tốc độ di chuyển (playerMovement.speed) được đặt thành tốc độ chạy nước rút (sprint\_Speed).
* Khoảng cách bước chân (player\_Footsteps.step\_Distance) và âm lượng bước chân (player\_Footsteps.volume\_Min, player\_Footsteps.volume\_Max) được đặt thành các giá trị tương ứng khi chạy nước rút.

if (Input.GetKeyUp(KeyCode.LeftShift) && !is\_Crouching)

{

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = sprint\_Step\_Distance;

player\_Footsteps.volume\_Min = sprint\_Volume;

player\_Footsteps.volume\_Max = sprint\_Volume;

}

Khi người chơi nhả phím LeftShift và nhân vật không đang cúi người, thiết lập các giá trị sau:

* Tốc độ di chuyển trở về tốc độ bình thường (move\_Speed).
* Khoảng cách bước chân và âm lượng bước chân trở về các giá trị tương ứng khi đi bộ.

if(Input.GetKey(KeyCode.LeftShift) && !is\_Crouching)

{

sprint\_Value -=sprint\_Treshold\*Time.deltaTime;

if(sprint\_Value <= 0f)

{

sprint\_Value = 0f;

//reset the speed and sound

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min = walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

}

player\_Stats.Display\_StaminaStats(sprint\_Value);

}

Khi người chơi giữ phím LeftShift và nhân vật không đang cúi người:

* Giảm stamina (sprint\_Value) theo thời gian (sprint\_Treshold \* Time.deltaTime).
* Nếu stamina cạn kiệt (sprint\_Value <= 0f), thiết lập lại tốc độ di chuyển và âm thanh bước chân như khi đi bộ.
* Hiển thị chỉ số stamina hiện tại lên màn hình (gọi hàm player\_Stats.Display\_StaminaStats(sprint\_Value)).

else

{

if(sprint\_Value != 100f)

{

sprint\_Value += (sprint\_Treshold / 2f) \* Time.deltaTime;

player\_Stats.Display\_StaminaStats(sprint\_Value);

if (sprint\_Value > 100f)

{

sprint\_Value = 100f;

}

}

}

Khi người chơi không giữ phím LeftShift hoặc nhân vật đang cúi người:

* Hồi phục stamina dần dần (sprint\_Value += (sprint\_Treshold / 2f) \* Time.deltaTime).
* Hiển thị chỉ số stamina hiện tại lên màn hình.
* Đảm bảo stamina không vượt quá giới hạn tối đa (100f).

Kết luận:

Hàm sprint() kiểm soát hành vi chạy nước rút của nhân vật bằng cách điều chỉnh tốc độ di chuyển, âm thanh bước chân, và quản lý lượng stamina. Khi người chơi nhấn hoặc giữ phím LeftShift, nhân vật sẽ chạy nhanh hơn, nhưng đồng thời cũng tiêu tốn stamina. Khi stamina cạn kiệt hoặc người chơi nhả phím, nhân vật sẽ trở lại tốc độ đi bộ bình thường và stamina sẽ dần hồi phục khi nhân vật không chạy.

void crouch()

{

if (Input.GetKeyDown(KeyCode.LeftControl))

{

if (is\_Crouching)

{

look\_Root.localPosition = new Vector3(0f, stand\_Height, 0f);

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min =walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

is\_Crouching = false;

}

else

{

look\_Root.localPosition = new Vector3(0f, crouch\_Height, 0f);

playerMovement.speed= crouch\_Speed;

player\_Footsteps.step\_Distance = crouch\_Step\_Distance;

player\_Footsteps.volume\_Min = crouch\_Volume;

player\_Footsteps.volume\_Max = crouch\_Volume;

is\_Crouching = true;

}

}

}

}

Kiểm tra khi phím LeftControl được nhấn:

if (Input.GetKeyDown(KeyCode.LeftControl))

Khi người chơi nhấn phím LeftControl, hàm sẽ kiểm tra trạng thái hiện tại của nhân vật (đang cúi hay không) và thực hiện các thay đổi tương ứng.

Nếu nhân vật đang cúi:

if (Input.GetKeyDown(KeyCode.LeftControl))

{

if (is\_Crouching)

{

look\_Root.localPosition = new Vector3(0f, stand\_Height, 0f);

playerMovement.speed = move\_Speed;

player\_Footsteps.step\_Distance = walk\_Step\_Distance;

player\_Footsteps.volume\_Min =walk\_Volume\_Min;

player\_Footsteps.volume\_Max = walk\_Volume\_Max;

is\_Crouching = false;

}

Nếu nhân vật đang cúi (is\_Crouching là true):

* Đặt vị trí của look\_Root (có thể là camera hoặc đầu nhân vật) về chiều cao đứng (stand\_Height).
* Thiết lập tốc độ di chuyển của nhân vật trở về tốc độ đi bộ (move\_Speed).
* Thiết lập khoảng cách bước chân và âm lượng bước chân trở về các giá trị tương ứng khi đi bộ.
* Đặt is\_Crouching thành false, cho biết nhân vật không còn cúi nữa.

else

{

look\_Root.localPosition = new Vector3(0f, crouch\_Height, 0f);

playerMovement.speed= crouch\_Speed;

player\_Footsteps.step\_Distance = crouch\_Step\_Distance;

player\_Footsteps.volume\_Min = crouch\_Volume;

player\_Footsteps.volume\_Max = crouch\_Volume;

is\_Crouching = true;

}

}

}

Nếu nhân vật không đang cúi (is\_Crouching là false):

* Đặt vị trí của look\_Root về chiều cao khi cúi (crouch\_Height).
* Thiết lập tốc độ di chuyển của nhân vật thành tốc độ cúi (crouch\_Speed).
* Thiết lập khoảng cách bước chân và âm lượng bước chân thành các giá trị tương ứng khi cúi.
* Đặt is\_Crouching thành true, cho biết nhân vật đang cúi.

Kết luận:

Hàm crouch() quản lý hành vi cúi người của nhân vật khi người chơi nhấn phím LeftControl. Cụ thể, hàm thay đổi vị trí của camera hoặc đầu nhân vật, tốc độ di chuyển, và âm thanh bước chân khi cúi hoặc đứng lên. Nếu nhân vật đang cúi, hàm sẽ chuyển nhân vật về trạng thái đứng và ngược lại.