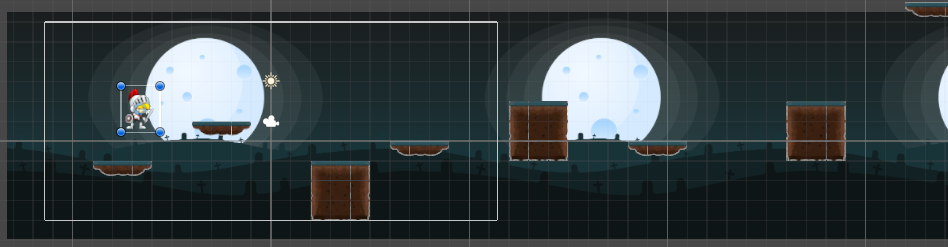
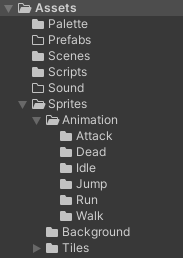
1. Animation của sprite (dự án jumping)

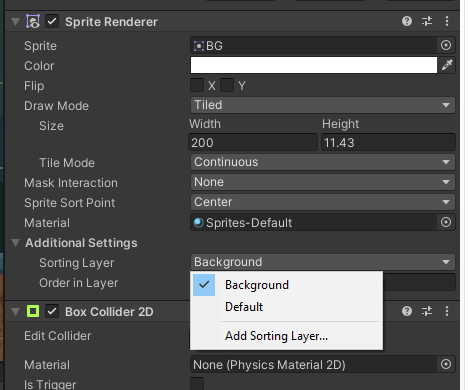
* Tạo project 2D kiểu như này



* Asset free lấy ở đây: <https://www.gameart2d.com/freebies.html> chọn một package bất kỳ để thiết kế
* Tạo cấu trúc thư mục trong project như sau:



* Level design sử dụng tilemap, background nhiều lớp có hiệu ứng parallax, Tạo mới order layer có tên là background rồi kéo lên phía trên layer Default, sau đó gán layer đó cho những hình ảnh background để bảo đảm chúng luôn nằm dưới khi render



**Tạo animation cho player**

* Player lấy sprite idle đưa vào Sprite Renderer để làm hình ảnh ban đầu cho player, add collider, rigidbody 2d cho player
* Add script PlayerController để điều khiển player di chuyển và nhảy như sau

public class PlayerController : MonoBehaviour

{

public float moveSpeed;

public float jumpForce;

public float radius;

Rigidbody2D rb;

private bool m\_FacingRight = true;

float horizontalInput;

float verticalInput;

[SerializeField] private Transform groundCheck;

[SerializeField] private LayerMask layerMaskGround;

private bool isGrounded =false; // kiem tra xem player co dang dung tren ground hay khong

private void OnDrawGizmos() // vẽ vòng tròn từ tâm của vị trí groundcheck trên player, tương ứng với Physics2D.OverlapCircle

{

Gizmos.color = Color.red;

Gizmos.DrawWireSphere(groundCheck.transform.position, radius);

}

// Start is called before the first frame update

void Start()

{

rb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

horizontalInput = Input.GetAxis("Horizontal");

verticalInput = Input.GetAxis("Vertical");

}

bool GroundCheck()

{

Collider2D hit = Physics2D.OverlapCircle(groundCheck.transform.position, radius, layerMaskGround);

if(hit != null) return true;

return false;

}

private void FixedUpdate()

{

rb.velocity = new Vector2(horizontalInput \* moveSpeed, rb.velocity.y);

isGrounded = GroundCheck();

// Kiểm tra xem nếu player đang đứng dưới ground và người chơi nhấn phím W

if (isGrounded && Input.GetKey(KeyCode.W))

{

// Add a vertical force to the player.

isGrounded = false;

rb.AddForce(new Vector2(0f, jumpForce), ForceMode2D.Impulse);

}

// điểu khiển player quay sang trái hoặc sang phải theo chiều nhấn

if (horizontalInput > 0 && !m\_FacingRight) Flip();

else if (horizontalInput < 0 && m\_FacingRight) Flip();

}

private void Flip()

{

// Switch the way the player is labelled as facing.

m\_FacingRight = !m\_FacingRight;

// Multiply the player's x local scale by -1.

Vector3 theScale = transform.localScale;

theScale.x \*= -1;

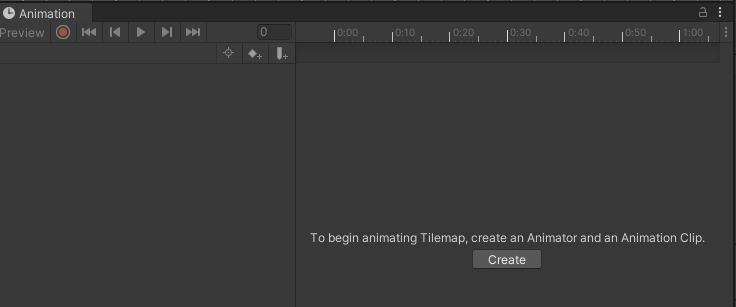
transform.localScale = theScale;

}

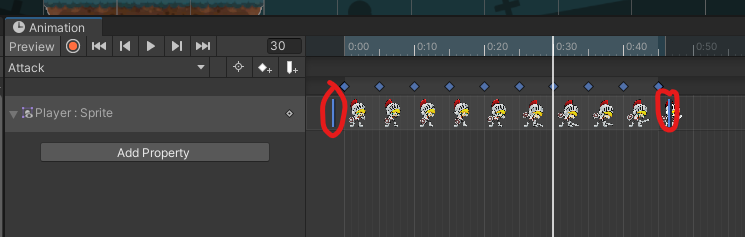
}

Tạo layermask để

* Click vào player, mở cửa sổ animation, click vào button create để tạo animation đầu tiên, đồng thời cũng tạo luôn animator và tự động gán vào component cho player

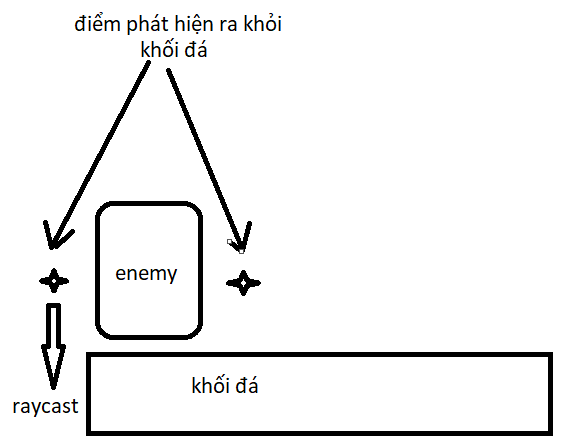


* Tạo animation bằng các kéo các sprite để tạo thành chuyển động vào trong cửa sổ frame của animation. Muốn tăng giảm tốc độ chuyển động của nhân vật thì ta thay đổi thời gian giữa mỗi frame(càng cao thì animation càng chậm). Chọn toàn bộ các key và click vào thanh dọc như hình dưới và kéo để thay đổi thời gian



Tạo Enemy và điều khiển enemy di chuyển trong game

* Tạo enemy tương tự như player, tạo prefab cho enemy
* Tạo các điểm có thể sinh enemy ở trong game, các điểm đó sẽ được thiết kế sẵn nằm trên các khối đá, tất cả điểm đó sẽ nằm trong 1 parent spawnEnemyGroup ở tọa độ 0,0,0. Viết script lấy tọa độ các điểm spawn là các child trong group để sinh enemy
* Điều khiển enemy di chuyển trên các khối đá, mỗi enemy sẽ có 2 điểm ở bên trái và phải để nhận biết là mình có di chuyển ra khỏi vị trí khối đá hay không, nếu raycast từ 2 điểm đó chiếu xuống không chạm vào collider của khối đá thì tức là nó đã ra tới mép của khối đá, cần di chuyển theo hướng ngược lại



* Để kiểm tra player đã tới gần enemy hay chưa, thì mỗi enemy sẽ có 1 raycast hướng ra phía ngoài hướng di chuyển để phát hiện
* Độ dài raycast phát hiện player, tốc độ di chuyển của mỗi enemy sử dụng các biến nhận giá trị random để tạo sự đa dạng trong game

Hiệu ứng paralax (game jumping, game endless run)

Animation của third person (làm demo một project third person)

Animation của UI (zoom in out, fade in out) Demo ứng dụng Dental

Chuyển động của các đối tượng khác trong game (rơi nhờ gravity), change position = script