Taller sobre videojuegos, experiencias interactivas y el mundo libre.

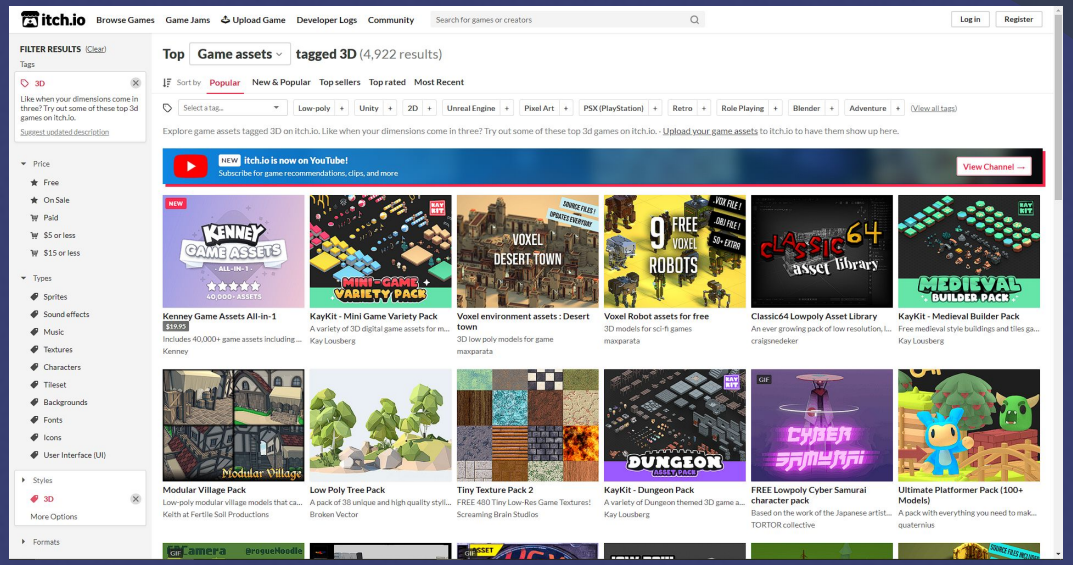
Cristhian Andres Grajales Perez.

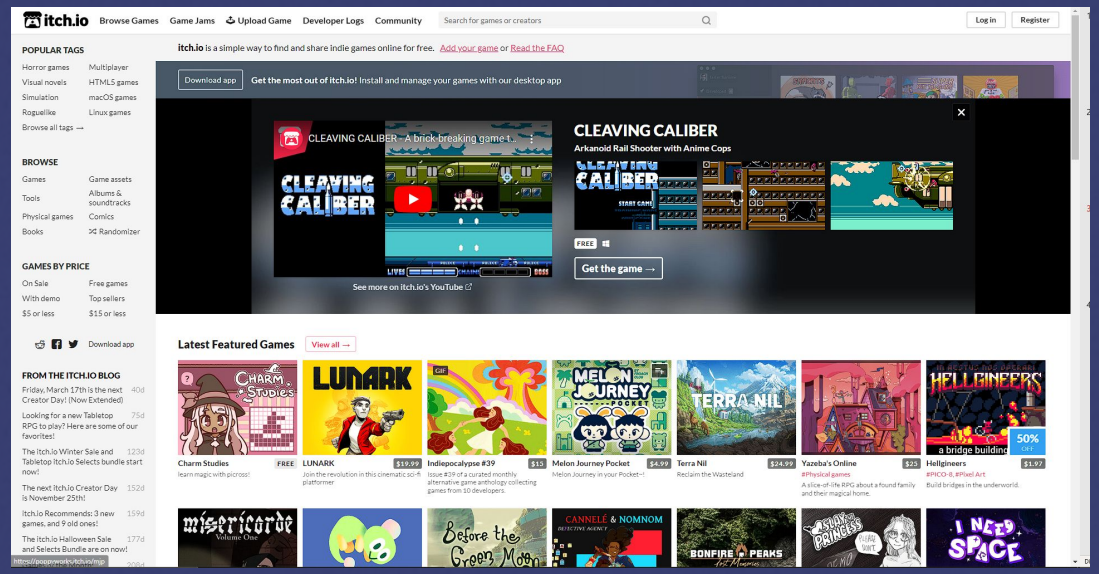
¿Qué se realizará?

- 1) ¿Qué es itch.io y como para que sirve?
- 2) Assets seleccionados.
- 3) Herramientas Necesarias.
- 4) Implementación de un juego isométrico.
  - ¿Que son las colisiones?
  - ¿Qué son las físicas?
  - Jugador y enemigos usando NavMesh
  - Cámara que persigue al jugador.
  - Interacción con el entorno (efecto de clic, Abrir puertas)
  - Tips de optimización
- 5) Repositorio.

¿Qué es itch.io y como para que sirve?

Es una tienda digital de videojuegos para PC y dispositivos móviles, enfocado principalmente en creadores independientes.





Assets seleccionados.

- Nombres de los paquetes:
  - **Ultimate Platformer Pack**
  - **Skeleton Outlaw 3D Character**
  - **3d low poly modular dungeon**

Herramientas Necesarias

- Unity 2021.3.15f1
- Visual Studio 2019

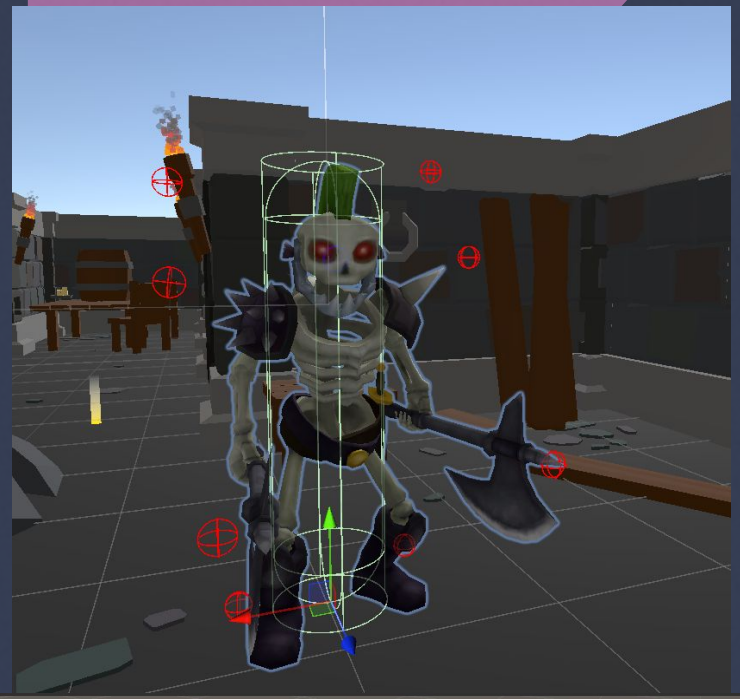
Otras Posibles

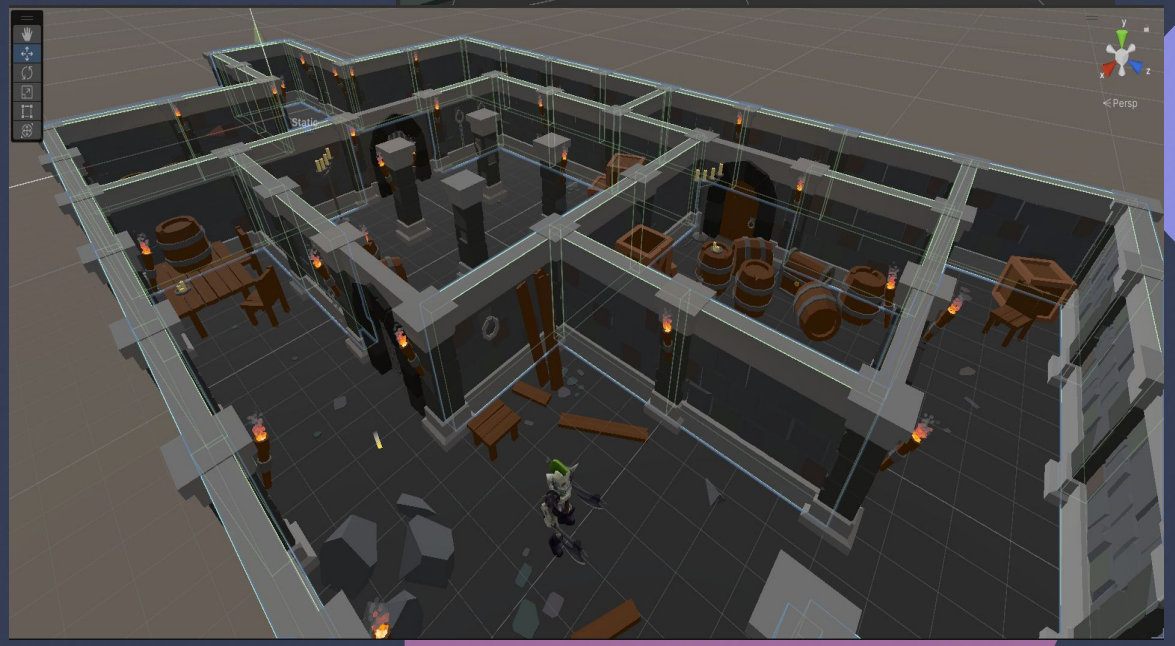
- Blender
- Gimp o Photoshop

Implementando un videojuego 3D Isométrico.

Construyendo el entorno.

- Importando assets.
- Colisiones.
- Físicas.



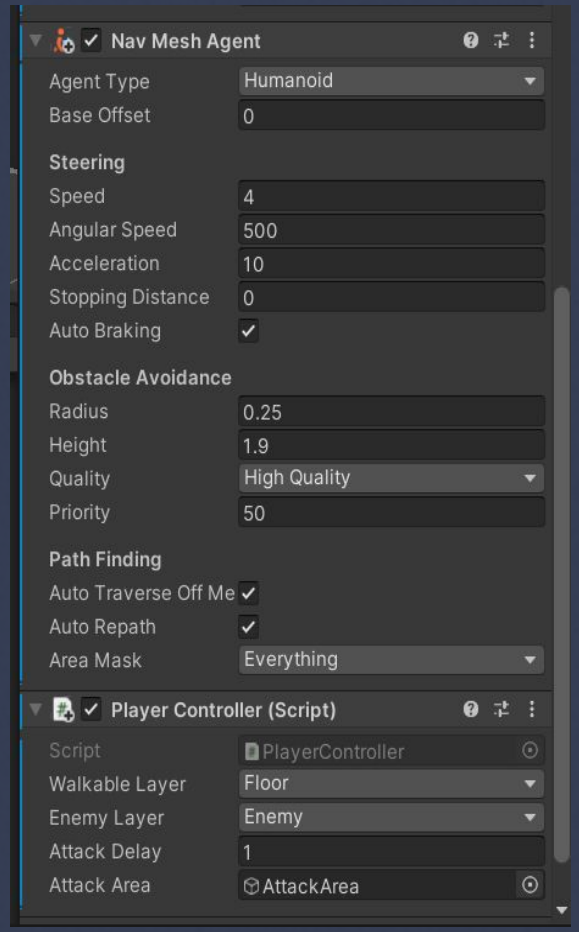


Desarrollando al jugador

- Movimiento usando NavMesh y la posición del mouse
- Ataque y área de daño.

Movimiento usando NavMesh

NavMesh es un sistema de navegación por mallas que se puede usar como un componente de unity para definir áreas por las cuales puede transitar un objeto de juego.



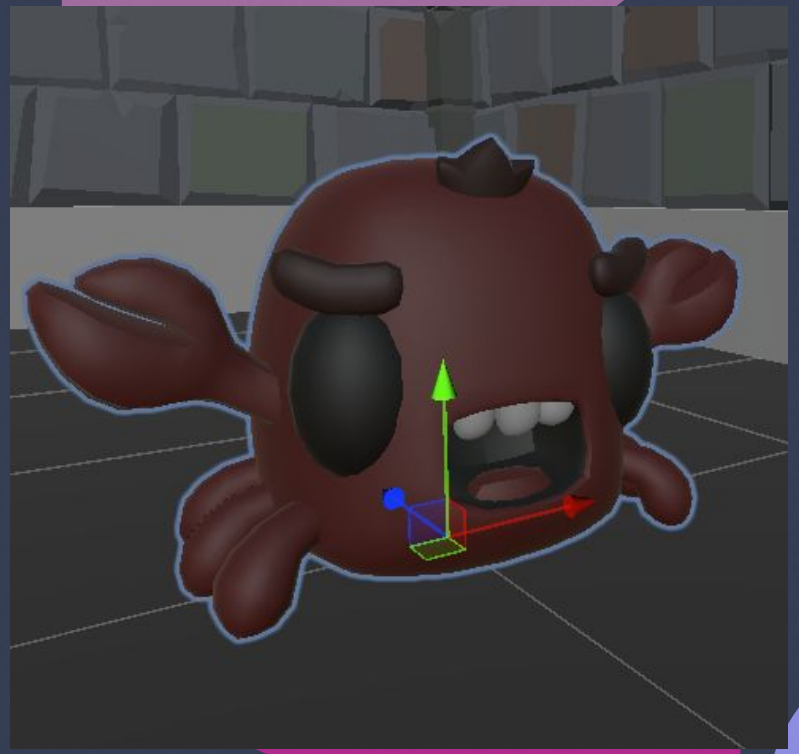

```
Mensaje de Unity | 0 referencias
void Update()
{
    //Si hizo clic en el juego verifica si es un enemigo o el suelo
    if (Input.GetMouseButtonDown(0))
    {
        _currentRay = Camera.main.ScreenPointToRay(Input.mousePosition);

        if (Physics.Raycast(_currentRay, out _rayData, 100, _enemyLayer.value))
        {
            Attack();
        }
        else if (Physics.Raycast(_currentRay, out _rayData, 100, _walkableLayer.value))
        {
            Run();
        }
    }
    FinishRun();
}

//Mueve al jugador a la posición que hizo clic
1 referencia
private void Run()
{
    _playerAgent.SetDestination(_rayData.point);
    _isRunning = true;
    _playerAnimation.SetBool("IsRunning", true);
    GameManager.Instance.CreateClickEffect(_rayData.point);
}

//Si el jugador está quieto, deja de hacer la animación de correr
1 referencia
private void FinishRun()
{
    if (_playerAgent.velocity == Vector3.zero)
    {
        _isRunning = false;
        _playerAnimation.SetBool("IsRunning", false);
    }
}
```

Ataque y área de daño

```
//Si la distancia es menor a 2 atacar y si no, persigue
1 referencia
private void Attack()
{
    if (_isAttacking) return;

    if(Vector3.Distance(transform.position, _rayData.point) < 2)
    {
        _isAttacking = true;
        _playerAnimation.SetBool("Attack", true);
        _attackArea.SetActive(true);
        _rayData.collider.gameObject.SetActive(true);
        StartCoroutine(FinishAttack());
    }
    else
    {
        _playerAgent.SetDestination(_rayData.point);
    }
}

//Espera a que acabe el tiempo delay de ataque para desactivarlo
1 referencia
public IEnumerator FinishAttack()
{
    yield return new WaitForSeconds(_attackDelay);
    _attackArea.SetActive(false);
    _playerAnimation.SetBool("Attack", false);
    _isAttacking = false;
}
```

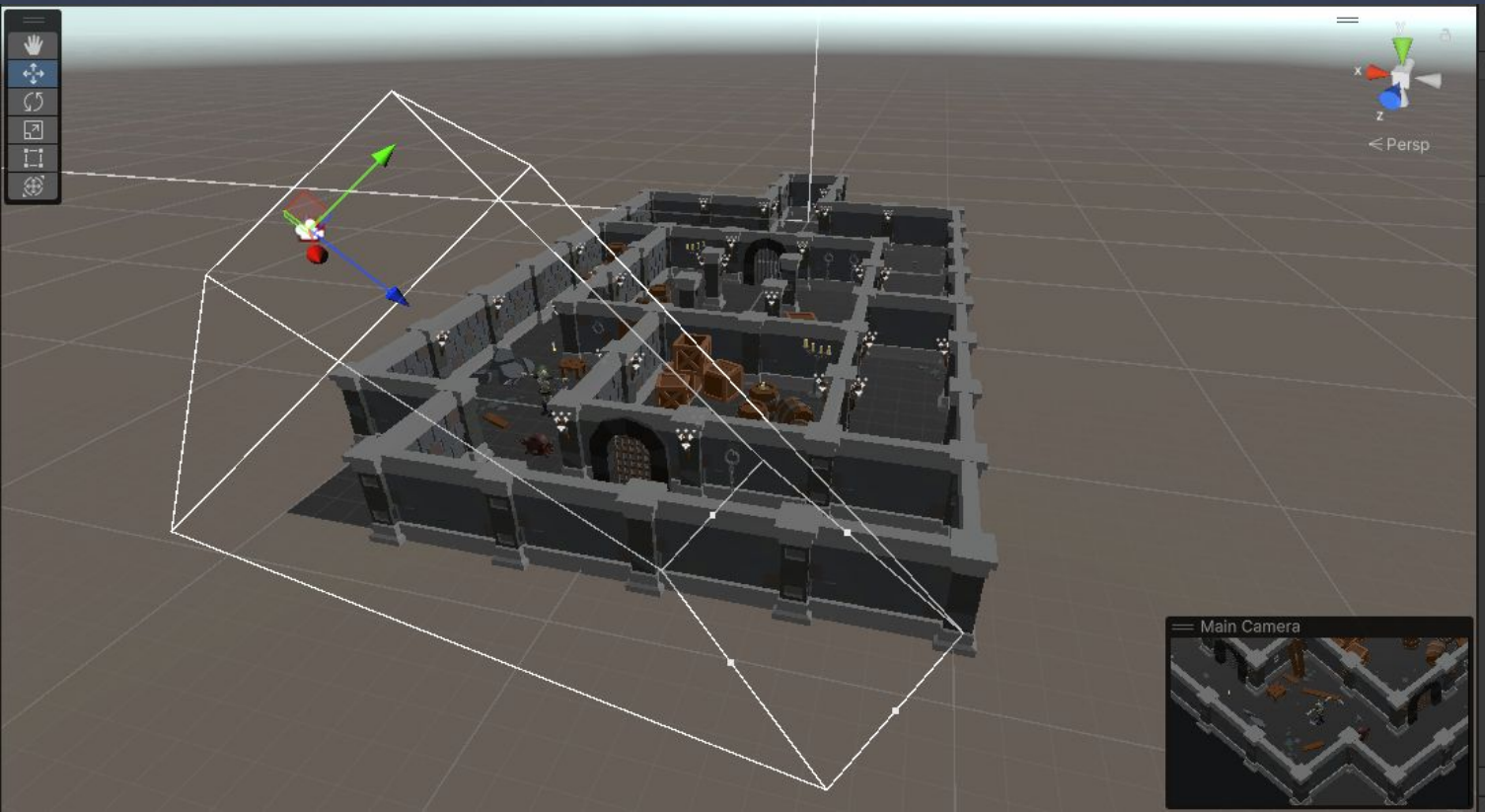
Desarrollando al Enemigo

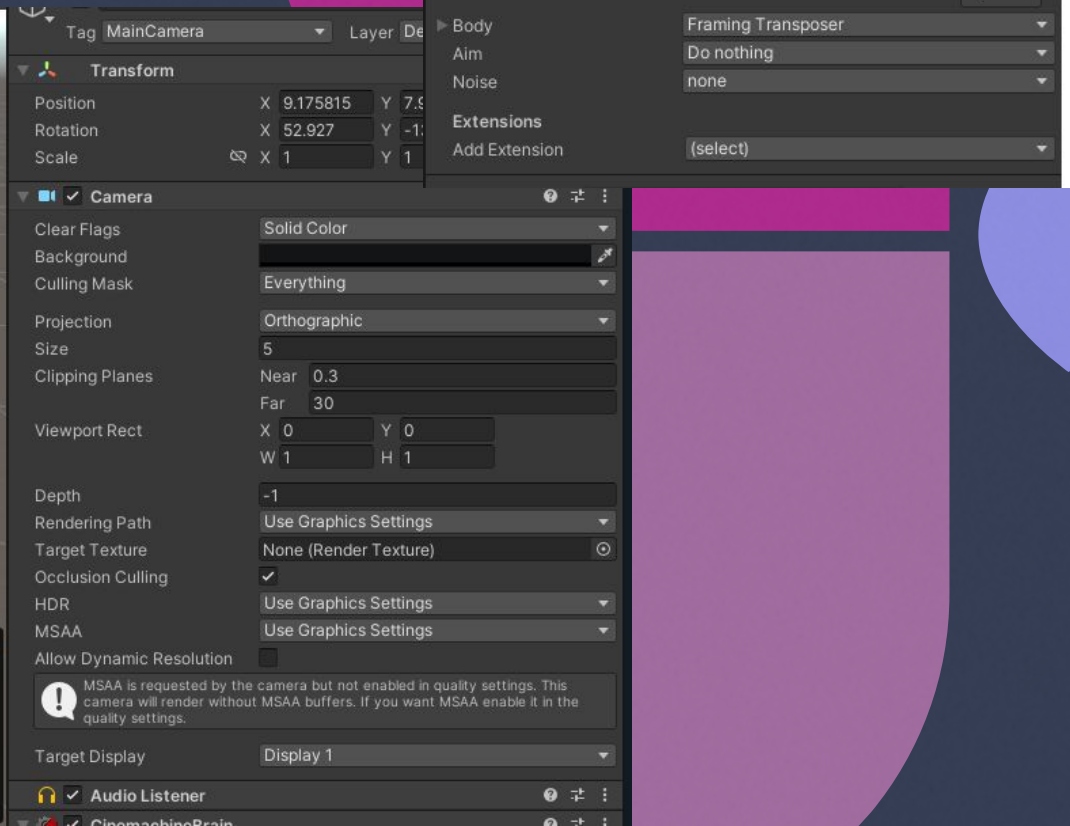
- Movimiento usando NavMesh
- Asignando tags para verificar la colisión



```
Script de Unity (1 referencia de recurso) | 0 referencias
public class Enemy : MonoBehaviour
{
    private NavMeshAgent _enemyAgent;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        _enemyAgent = GetComponent<NavMeshAgent>();

    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        _enemyAgent.SetDestination(GameManager.Instance.GetPlayer().transform.position);
    }
}
```

Cámara que persigue al jugador.

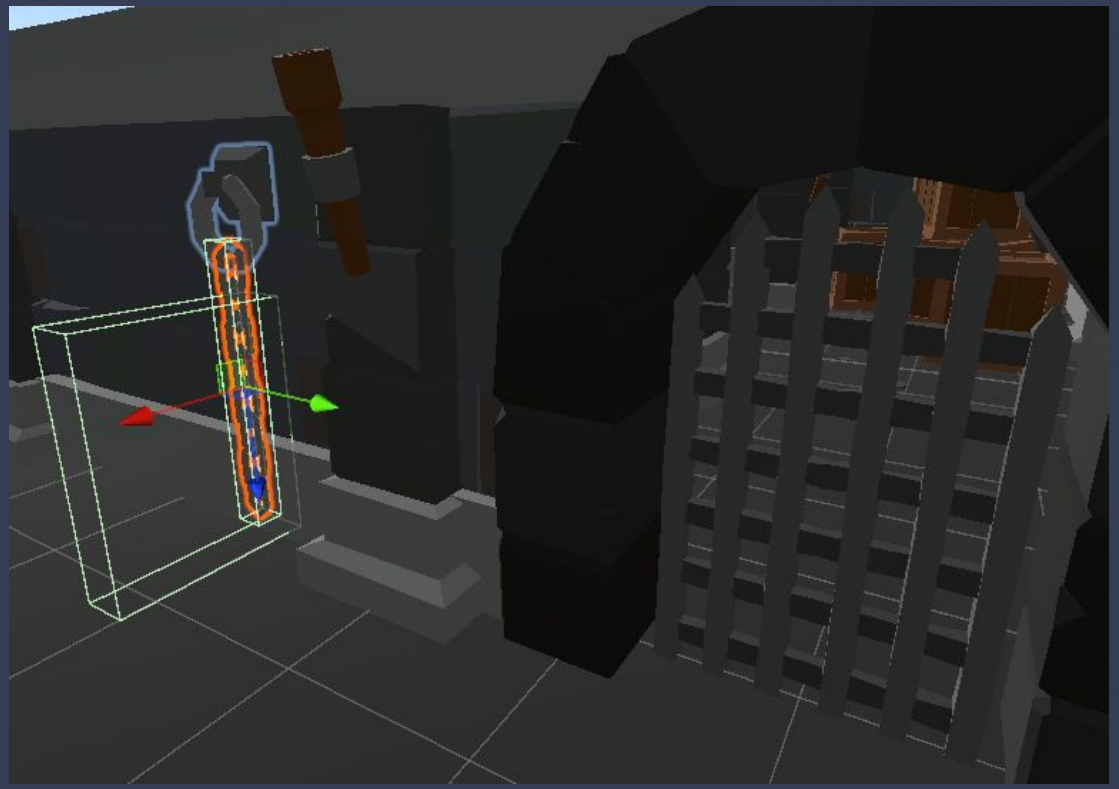




Interacción con el entorno

- Creando áreas de detección con Triggers.
- Abrir puertas al presionar una tecla.
- Efecto de Clic en el suelo

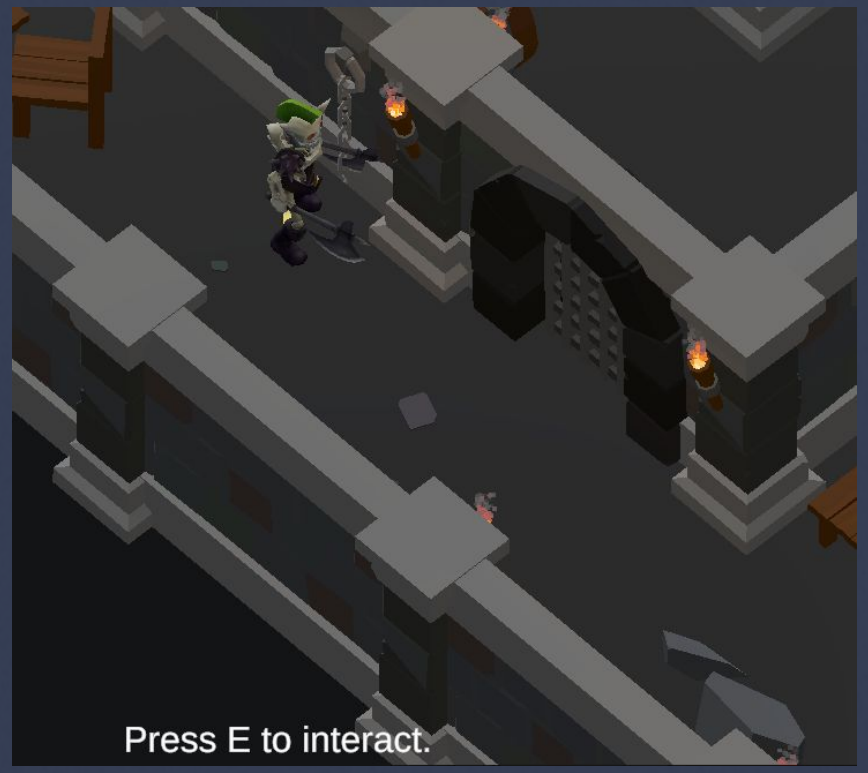
Creando áreas de detección con Triggers.



```
private bool _isDetecting;

//Asigna el tipo de objeto interactuable
1 referencia
public enum InteractionType
{
    OpenDoor,
    DialogWithNPC,
    LootItem
}

[SerializeField] private InteractionType _currentInteractionType;
[SerializeField] private GameObject _doorToOpen;

//Verifica si el jugador entra al area de detección para activar el texto de interaccion
Mensaje de Unity | 0 referencias
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player"))
    {
        _isDetecting = true;
        GameManager.Instance.OpenInteractText(_isDetecting);
    }
}
```

```
//Verifica si el jugador sale del area de detección para desactivar el texto de interaccion
Mensaje de Unity | 0 referencias
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("Player"))
    {
        _isDetecting = false;
        GameManager.Instance.OpenInteractText(_isDetecting);
    }
}
```

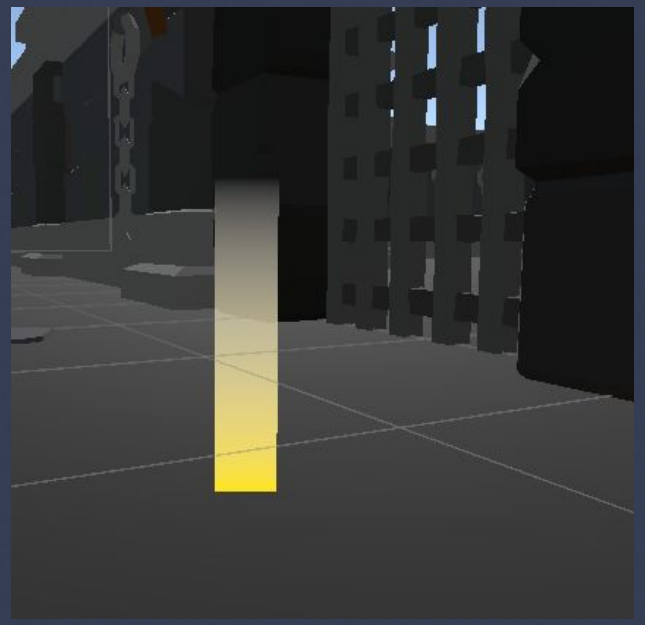
Creando áreas de detección con Triggers.




```
//Desactiva la puerta si presiona E mientras está dentro del area de interacción
Mensaje de Unity | 0 referencias
private void Update()
{
    if (_isDetecting && Input.GetKeyDown(KeyCode.E))
    {
        _doorToOpen.gameObject.SetActive(false);
    }
}
```

```
//Activa el texto de interaccion cuando entra en el area de deteccion
2 referencias
public void OpenInteractText(bool state)
{
    _interactUIText.SetActive(state);
}
```

Efecto de Clic en el suelo



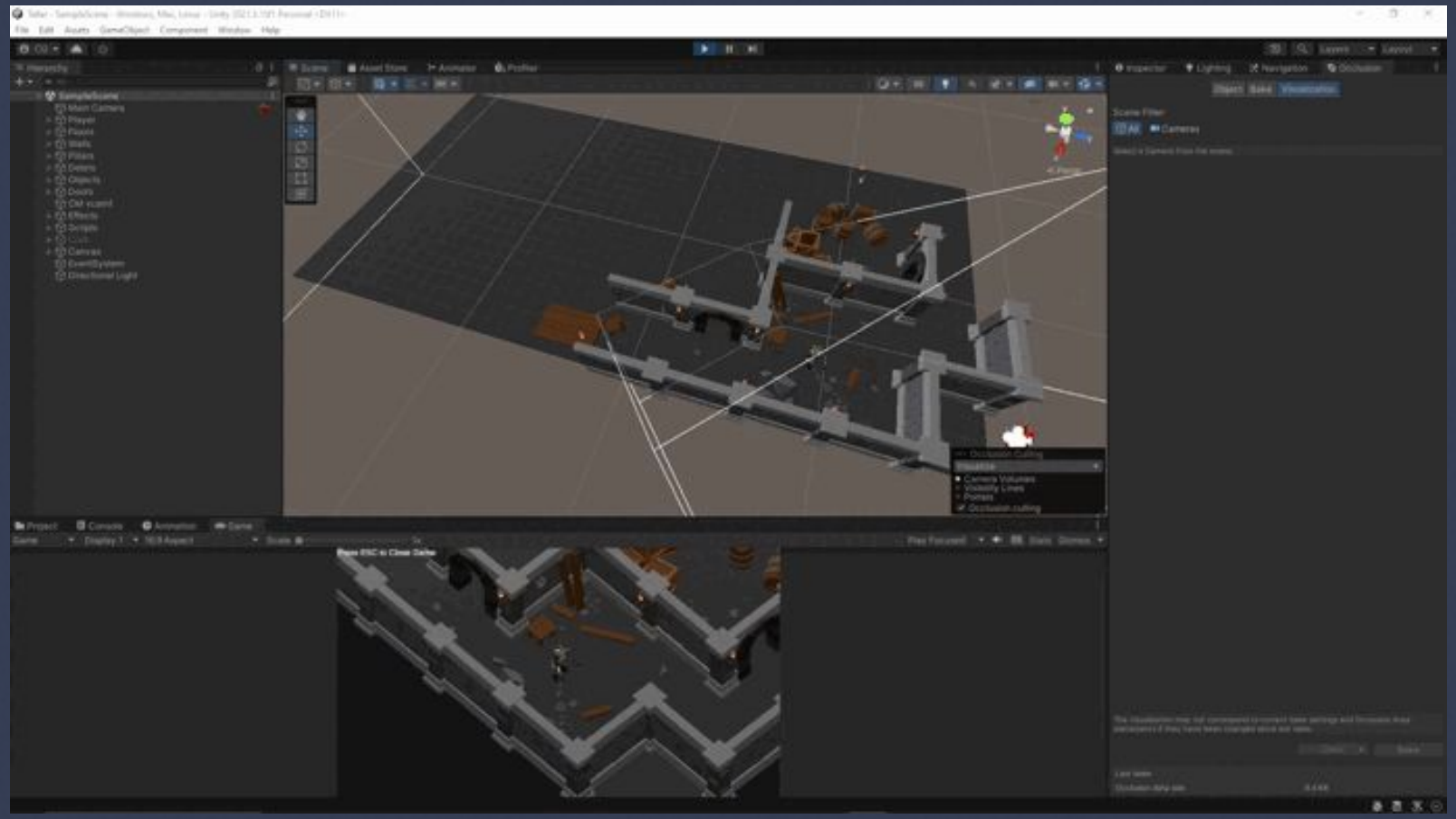
```
//Ubica el efecto del click en una posición especifica
1 referencia
public void CreateClickEffect(Vector3 effectPosition)
{
    _clickEffect.DoEffect(effectPosition);
}
```

```
//Cambia el objeto del efecto de acuerdo a la posición que se hizo clic
1 referencia
public void DoEffect(Vector3 effectPosition)
{
    _clickCheck.transform.position = effectPosition;
}
```

Tips de optimización de videojuegos.

- Occlusion Culling
- Objetos estáticos y dinámicos

Occlusion Culling



Objetos estáticos y dinámicos

Repositorio

- https://github.com/Cristhian47/SkullDungeon