THE LAST TACO

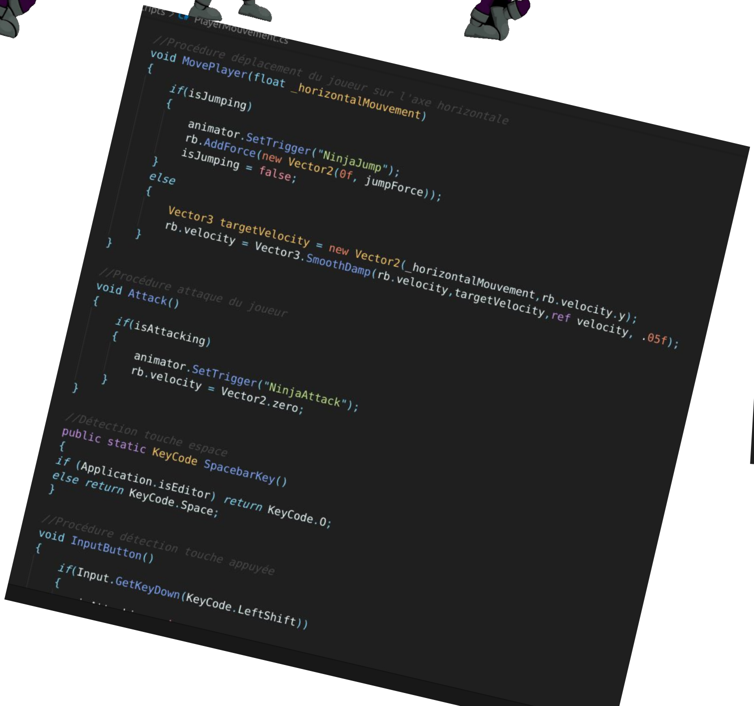
Un projet de RAKOTOMALALA Lucas et BONIS Alexis

Concept

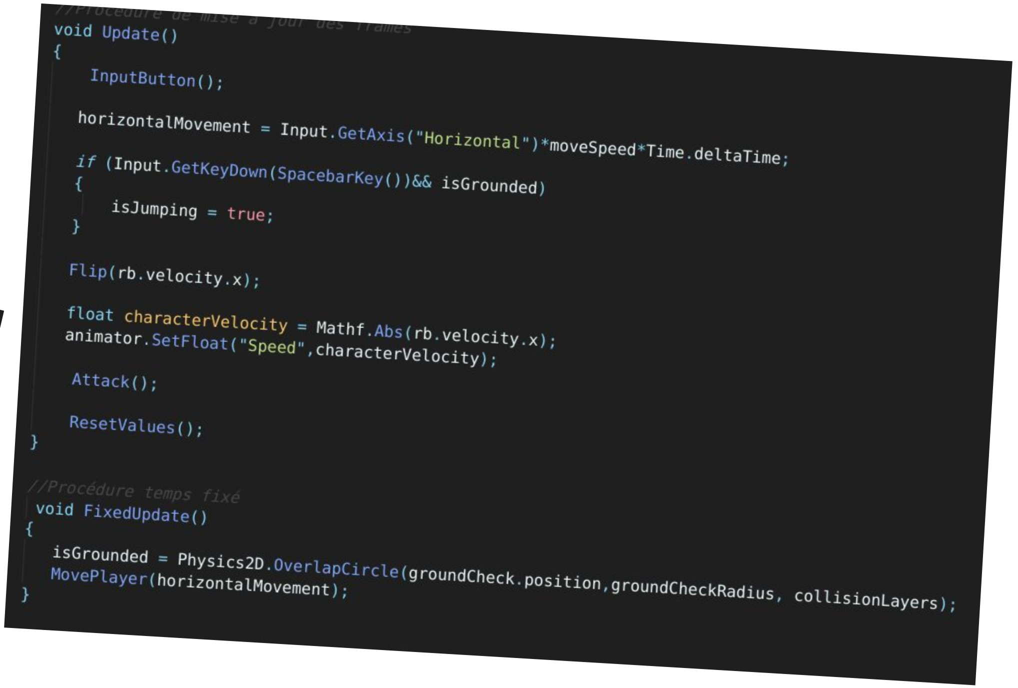
- Jeux de type platformer 2D sur Unity
- A partir de 10 ans, grande importance 12-25
- Durée moyenne du jeux 1h
- Langage c#
- Unity Collab
- Méthode Agile
- Contrainte durée et organisation

SOMMAIRE

Déplacements/Animations

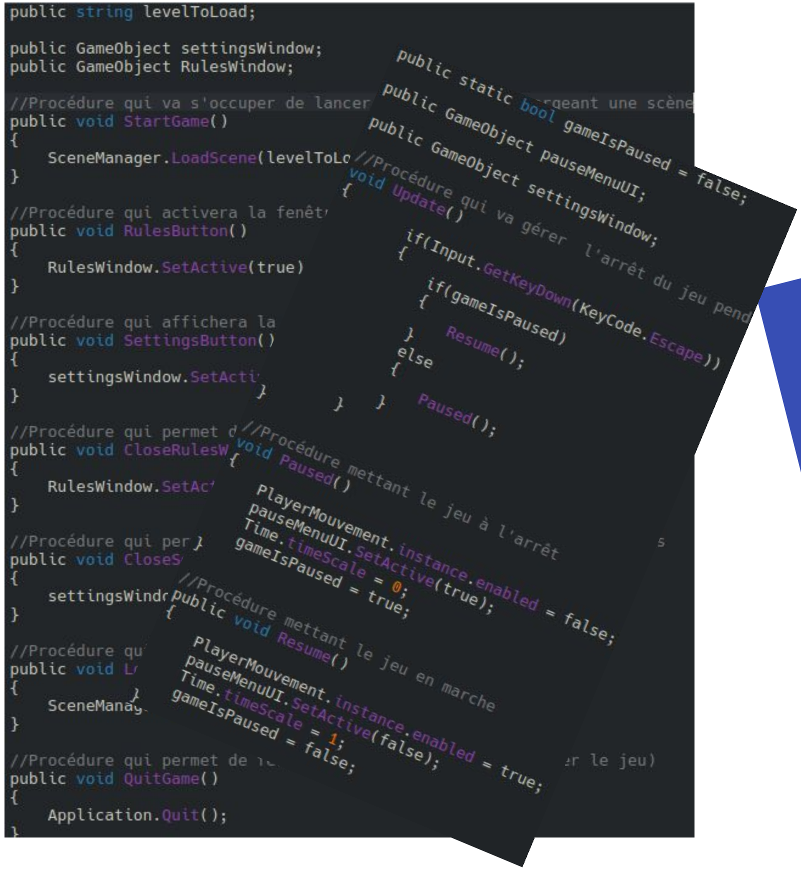


```
ripts > C# PlayerMouvement.cs
//Procédure déplacement du joueur sur l'axe horizontale
void MovePlayer(float _horizontalMouvement)
{
    if(isJumping)
    {
        animator.SetTrigger("NinjaJump");
        rb.AddForce(new Vector2(0f, jumpForce));
        isJumping = false;
    }
    else
    {
        Vector3 targetVelocity = new Vector2(_horizontalMouvement,rb.velocity.y);
        rb.velocity = Vector3.SmoothDamp(rb.velocity,targetVelocity,ref velocity, .05f);
    }
}

//Procédure attaque du joueur
void Attack()
{
    if(isAttacking)
    {
        animator.SetTrigger("NinjaAttack");
        rb.velocity = Vector2.zero;
    }
}

//Détection touche espace
public static KeyCode SpacebarKey()
{
if (Application.isEditor) return KeyCode.O;
else return KeyCode.Space;
}

//Procédure détection touche appuyée
void InputButton()
{
    if(Input.GetKeyDown(KeyCode.LeftShift))
    {
```



```
//Procédure de mise à jour des frames
void Update()
{
    InputButton();

    horizontalMovement = Input.GetAxis("Horizontal")*moveSpeed*Time.deltaTime;

    if (Input.GetKeyDown(SpacebarKey())&& isGrounded)
    {
        isJumping = true;
    }

    Flip(rb.velocity.x);

    float characterVelocity = Mathf.Abs(rb.velocity.x);
    animator.SetFloat("Speed",characterVelocity);

    Attack();

    ResetValues();
}

//Procédure temps fixé
void FixedUpdate()
{
    isGrounded = Physics2D.OverlapCircle(groundCheck.position,groundCheckRadius, collisionLayers);
    MovePlayer(horizontalMovement);
}
```

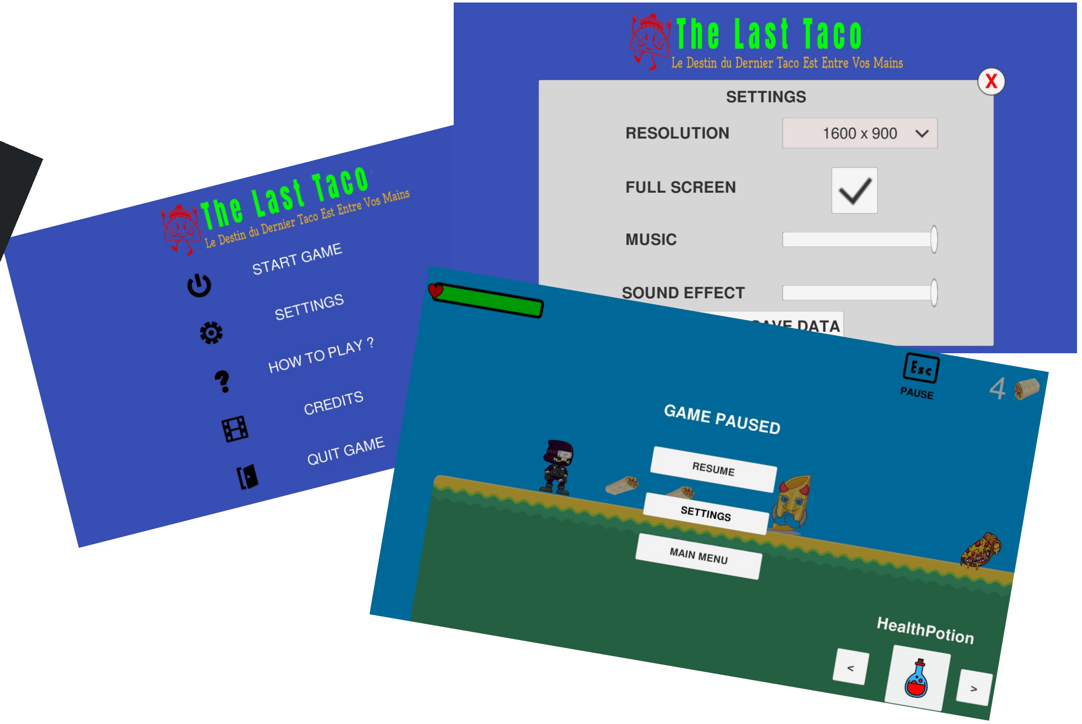
Menu Principal/Menu Pause/Crédits



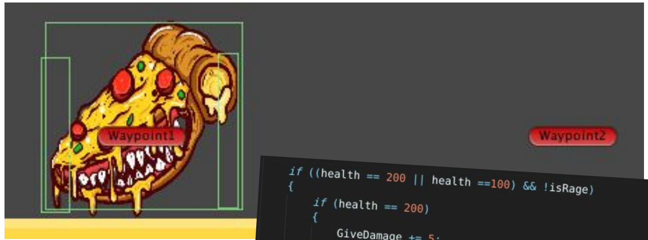
```
public string levelToLoad;

public GameObject settingsWindow;
public GameObject RulesWindow;

//Procédure qui va s'occuper de lancer ... rgeant une scène
public void StartGame()
{
    SceneManager.LoadScene(levelToLo
}

//Procédure qui activera la fenêt
public void RulesButton()
{
    RulesWindow.SetActive(true)
}

//Procédure qui affichera la
public void SettingsButton()
{
    settingsWindow.SetActi
}

//Procédure qui permet d
public void CloseRulesW
{
    RulesWindow.SetAc
}

//Procédure qui per
public void CloseS
{
    settingsWindo
}

//Procédure qu
public void L
{
    SceneManag
}

//Procédure qui permet de ... er le jeu)
public void QuitGame()
{
    Application.Quit();
}
```

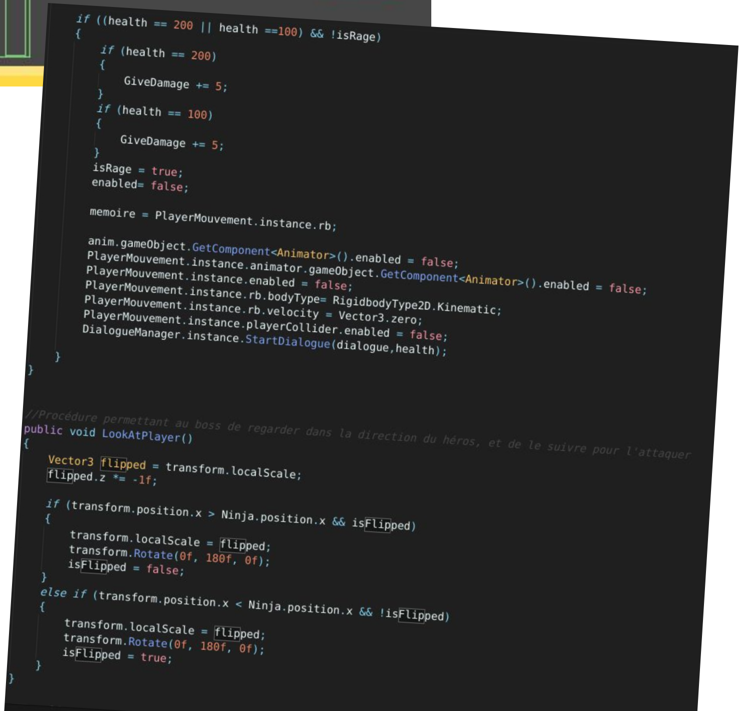
```
public static bool gameIsPaused = false;
public GameObject pauseMenuUI;
public GameObject settingsWindow;
//Procédure qui va gérer  l'arrêt du jeu pend
void Update()
{
    if(Input.GetKeyDown(KeyCode.Escape))
    {
        if(gameIsPaused)
        {
            Resume();
        }
        else
        {
            Paused();
        }
    }
}
```

```
//Procédure mettant le jeu à l'arrêt
void Paused()
{
    PlayerMouvement.instance.enabled = false;
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0;
    gameIsPaused = true;
}
//Procédure mettant le jeu en marche
public void Resume()
{
    PlayerMouvement.instance.enabled = true;
    pauseMenuUI.SetActive(false);
    Time.timeScale = 1;
    gameIsPaused = false;
}
```

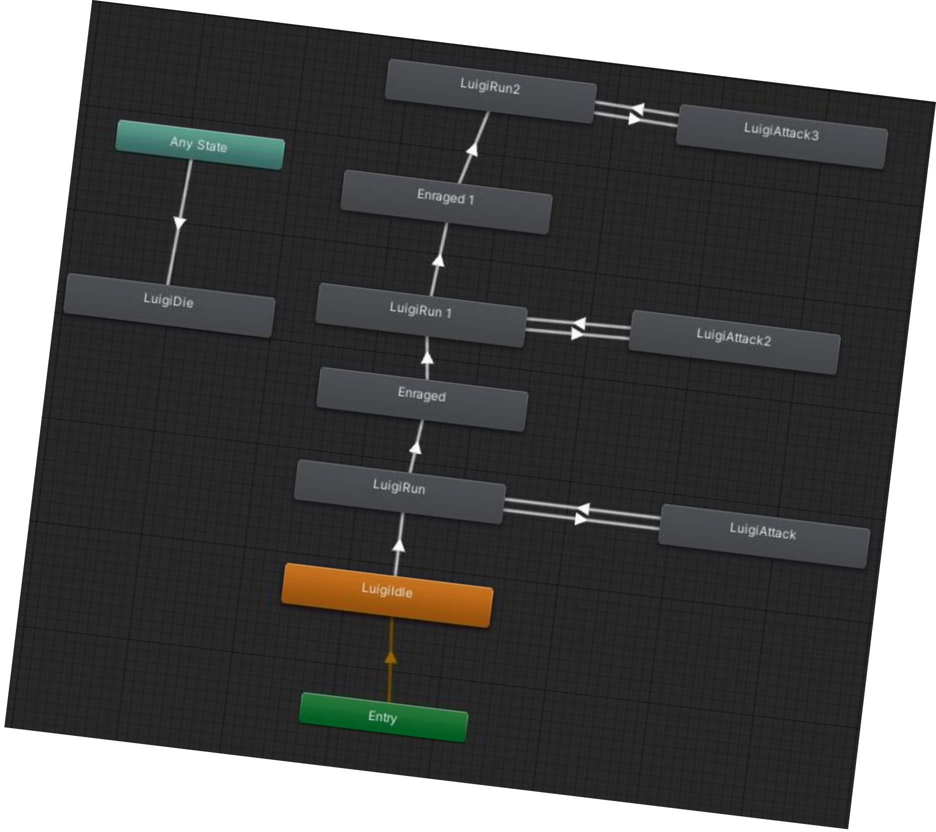


Ennemis/Boss

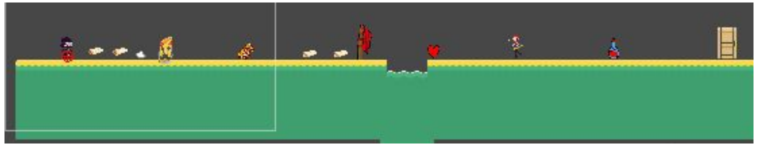


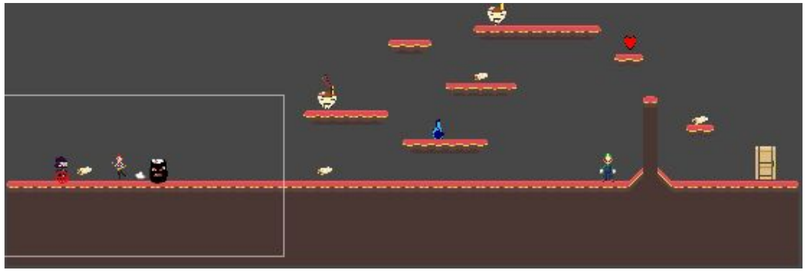


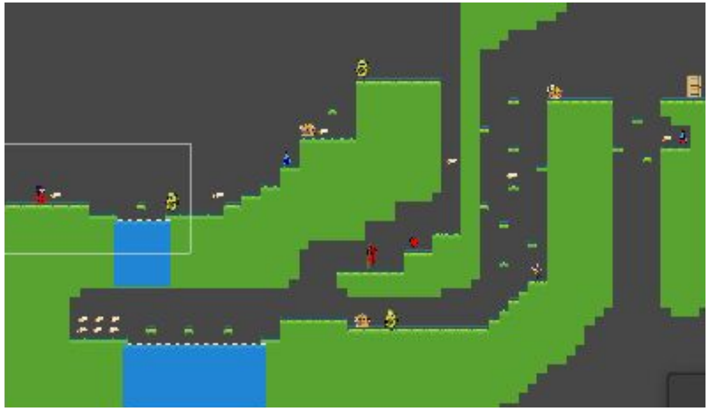
```
    if ((health == 200 || health ==100) && !isRage)
    {
        if (health == 200)
        {
            GiveDamage += 5;
        }
        if (health == 100)
        {
            GiveDamage += 5;
        }
        isRage = true;
        enabled= false;

        memoire = PlayerMouvement.instance.rb;

        anim.gameObject.GetComponent<Animator>().enabled = false;
        PlayerMouvement.instance.animator.gameObject.GetComponent<Animator>().enabled = false;
        PlayerMouvement.instance.enabled = false;
        PlayerMouvement.instance.rb.bodyType= RigidbodyType2D.Kinematic;
        PlayerMouvement.instance.rb.velocity = Vector3.zero;
        PlayerMouvement.instance.playerCollider.enabled = false;
        DialogueManager.instance.StartDialogue(dialogue,health);
    }
}


//Procédure permettant au boss de regarder dans la direction du héros, et de le suivre pour l'attaquer
public void LookAtPlayer()
{
    Vector3 flipped = transform.localScale;
    flipped.z *= -1f;

    if (transform.position.x > Ninja.position.x && isFlipped)
    {
        transform.localScale = flipped;
        transform.Rotate(0f, 180f, 0f);
        isFlipped = false;
    }
    else if (transform.position.x < Ninja.position.x && !isFlipped)
    {
        transform.localScale = flipped;
        transform.Rotate(0f, 180f, 0f);
        isFlipped = true;
    }
}
```

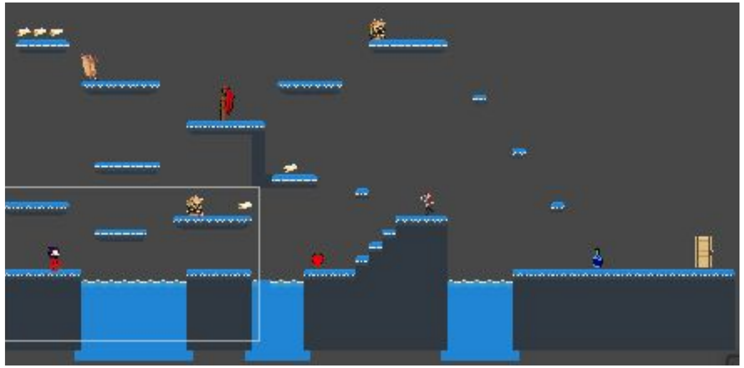


Design Niveaux

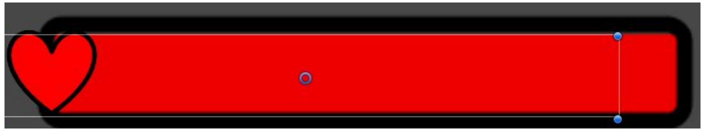




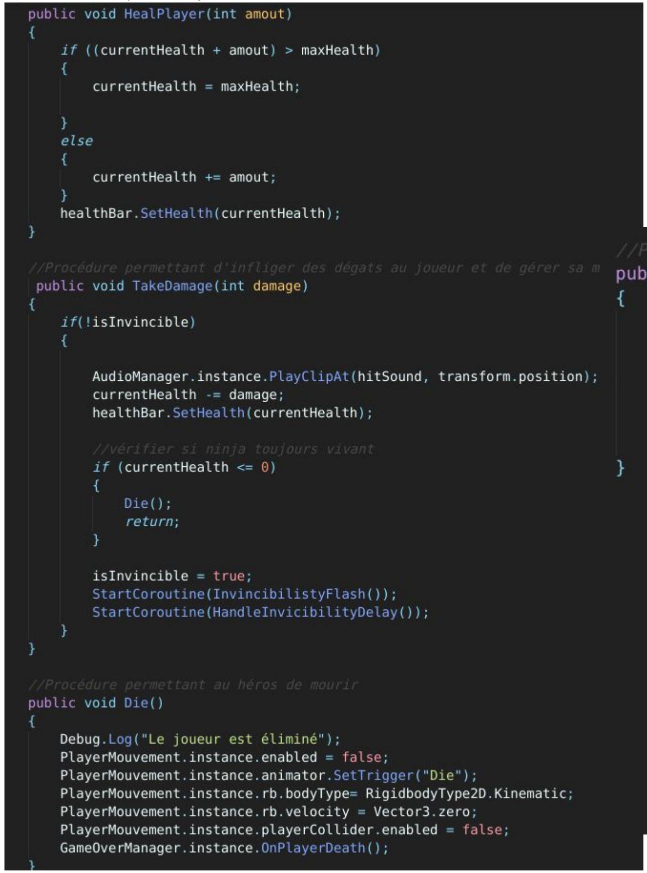




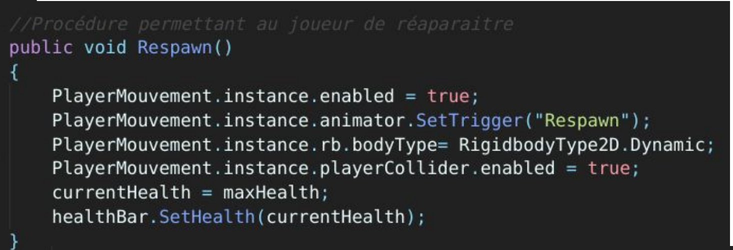
Vie/Mort du personnage



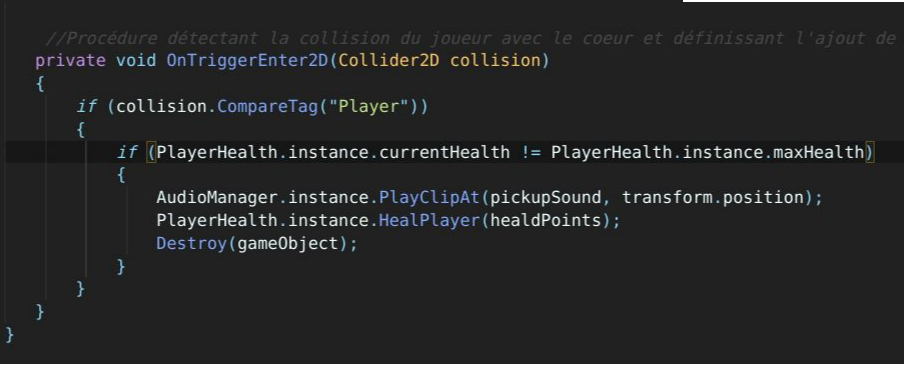
```
public int maxHealth = 100;
public float invicibilityTimeAfterHit = 1.5f;
public float invicibilityFlashDelay = 0.3f;
```



```
public void HealPlayer(int amout)
{
    if ((currentHealth + amout) > maxHealth)
    {
        currentHealth = maxHealth;

    }
    else
    {
        currentHealth += amout;
    }
    healthBar.SetHealth(currentHealth);
}

//Procédure permettant d'infliger des dégats au joueur et de gérer sa m
 public void TakeDamage(int damage)
{
    if(!isInvincible)
    {

        AudioManager.instance.PlayClipAt(hitSound, transform.position);
        currentHealth -= damage;
        healthBar.SetHealth(currentHealth);

        //vérifier si ninja toujours vivant
        if (currentHealth <= 0)
        {
            Die();
            return;
        }

        isInvincible = true;
        StartCoroutine(InvincibilistyFlash());
        StartCoroutine(HandleInvicibilityDelay());
    }
}

//Procédure permettant au héros de mourir
public void Die()
{
    Debug.Log("Le joueur est éliminé");
    PlayerMouvement.instance.enabled = false;
    PlayerMouvement.instance.animator.SetTrigger("Die");
    PlayerMouvement.instance.rb.bodyType= RigidbodyType2D.Kinematic;
    PlayerMouvement.instance.rb.velocity = Vector3.zero;
    PlayerMouvement.instance.playerCollider.enabled = false;
    GameOverManager.instance.OnPlayerDeath();
}
```



```
//Procédure permettant au joueur de réaparaitre
public void Respawn()
{
    PlayerMouvement.instance.enabled = true;
    PlayerMouvement.instance.animator.SetTrigger("Respawn");
    PlayerMouvement.instance.rb.bodyType= RigidbodyType2D.Dynamic;
    PlayerMouvement.instance.playerCollider.enabled = true;
    currentHealth = maxHealth;
    healthBar.SetHealth(currentHealth);
}
```

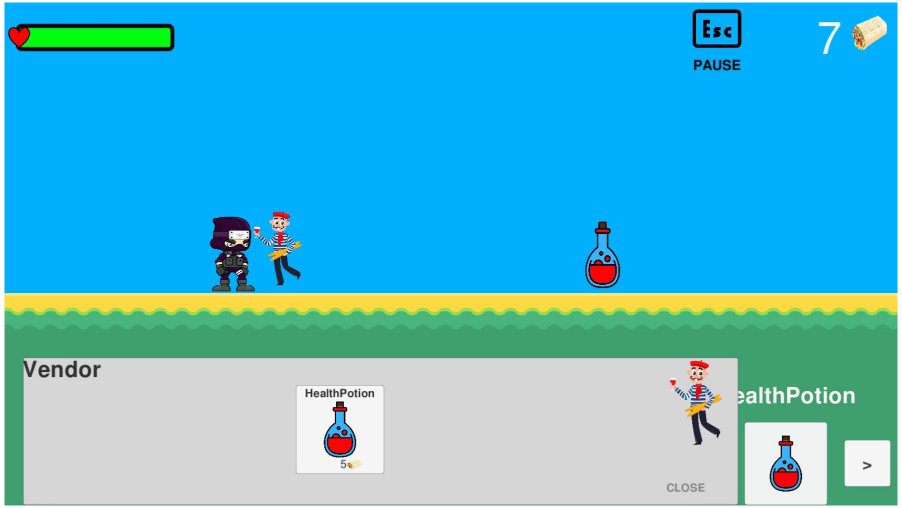


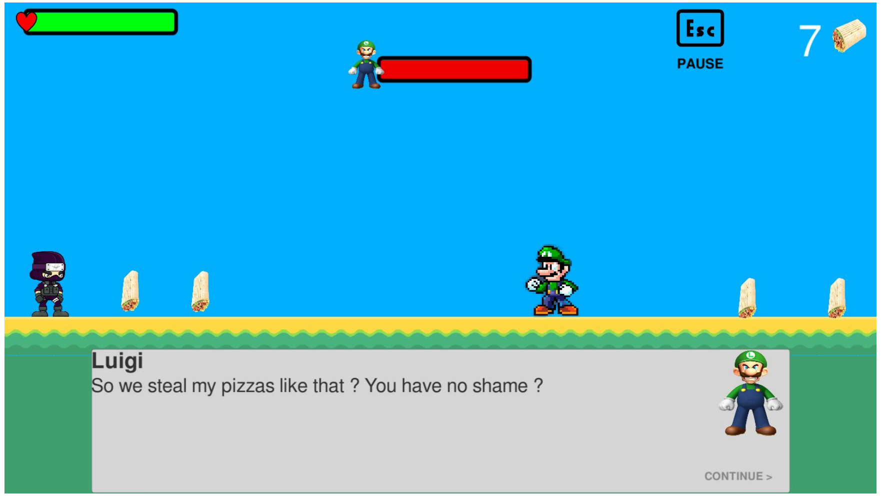
```
    //Procédure détectant la collision du joueur avec le coeur et définissant l'ajout de
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            if (PlayerHealth.instance.currentHealth != PlayerHealth.instance.maxHealth)
            {
                AudioManager.instance.PlayClipAt(pickupSound, transform.position);
                PlayerHealth.instance.HealPlayer(healdPoints);
                Destroy(gameObject);
            }
        }
    }
}
```

PNJs/Dialogues

Création d’une classe dialogue et une autre pour les gérer

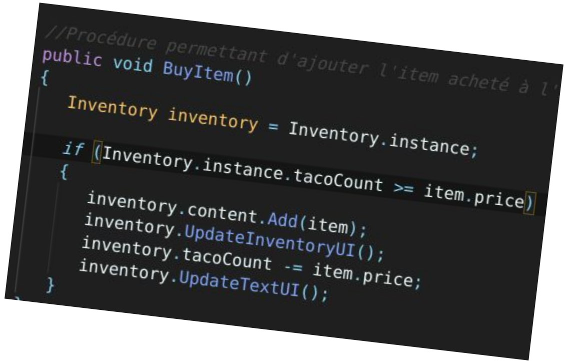
Création d’une classe gérant le marchand et l’achat d’items



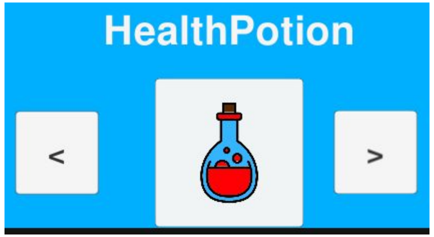


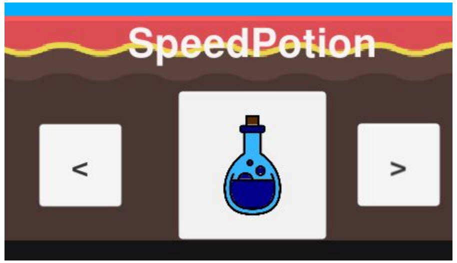
So we steal my pizzas like that ? You have no shame ?

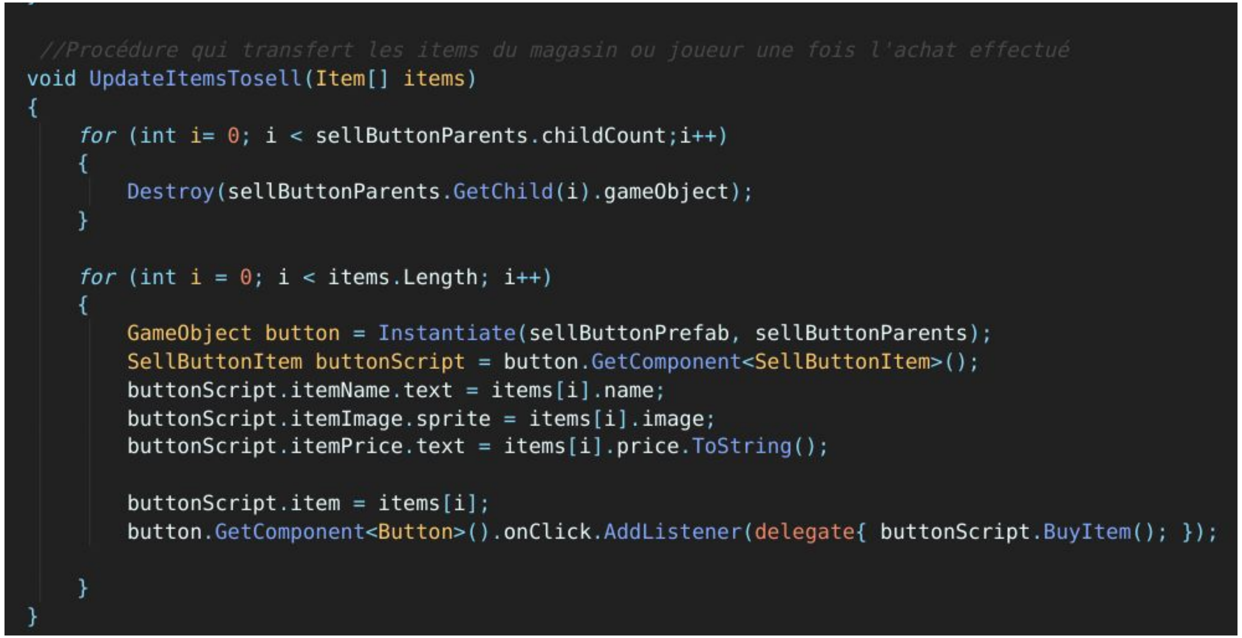
Inventaire/Items



```
//Procédure permettant d'ajouter l'item acheté à l'
public void BuyItem()
{
    Inventory inventory = Inventory.instance;

    if (Inventory.instance.tacoCount >= item.price)
    {
        inventory.content.Add(item);
        inventory.UpdateInventoryUI();
        inventory.tacoCount -= item.price;
        inventory.UpdateTextUI();
    }
```

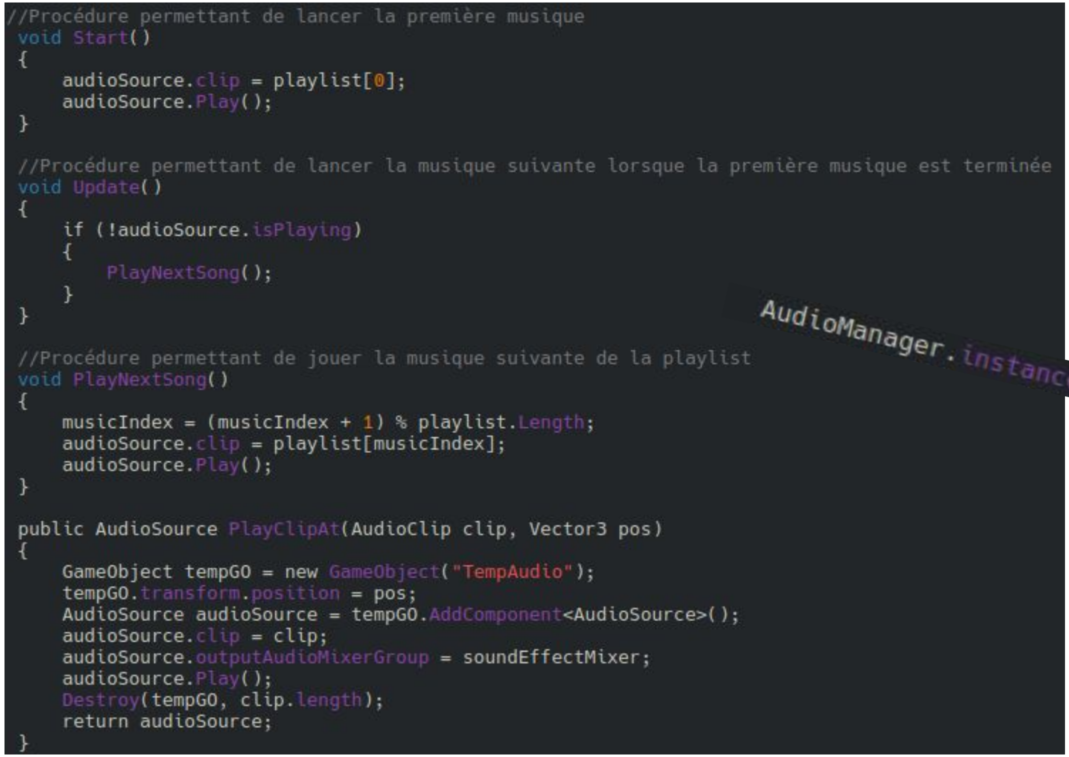




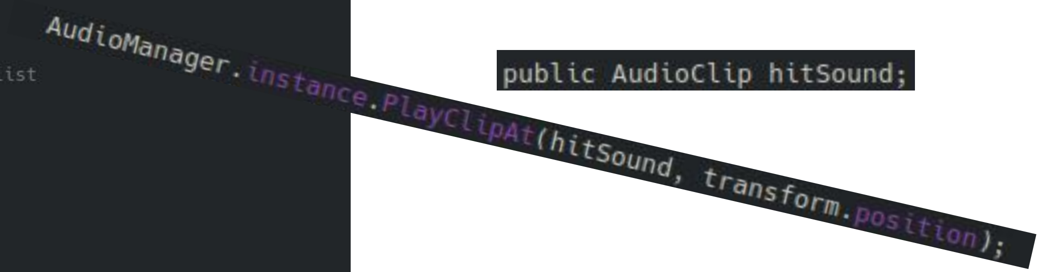


```
//Procédure qui transfert les items du magasin ou joueur une fois l'achat effectué
void UpdateItemsTosell(Item[] items)
{
    for (int i= 0; i < sellButtonParents.childCount;i++)
    {
        Destroy(sellButtonParents.GetChild(i).gameObject);
    }

    for (int i = 0; i < items.Length; i++)
    {
        GameObject button = Instantiate(sellButtonPrefab, sellButtonParents);
        SellButtonItem buttonScript = button.GetComponent<SellButtonItem>();
        buttonScript.itemName.text = items[i].name;
        buttonScript.itemImage.sprite = items[i].image;
        buttonScript.itemPrice.text = items[i].price.ToString();

        buttonScript.item = items[i];
        button.GetComponent<Button>().onClick.AddListener(delegate{ buttonScript.BuyItem(); });

    }
}
```

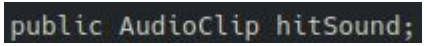
Effets sonores/musiques



```
//Procédure permettant de lancer la première musique
 void Start()
 {
     audioSource.clip = playlist[0];
     audioSource.Play();
 }

 //Procédure permettant de lancer la musique suivante lorsque la première musique est terminée
 void Update()
 {
     if (!audioSource.isPlaying)
     {
         PlayNextSong();
     }
 }

 //Procédure permettant de jouer la musique suivante de la playlist
 void PlayNextSong()
 {
     musicIndex = (musicIndex + 1) % playlist.Length;
     audioSource.clip = playlist[musicIndex];
     audioSource.Play();
 }

 public AudioSource PlayClipAt(AudioClip clip, Vector3 pos)
 {
     GameObject tempGO = new GameObject("TempAudio");
     tempGO.transform.position = pos;
     AudioSource audioSource = tempGO.AddComponent<AudioSource>();
     audioSource.clip = clip;
     audioSource.outputAudioMixerGroup = soundEffectMixer;
     audioSource.Play();
     Destroy(tempGO, clip.length);
     return audioSource;
 }
```

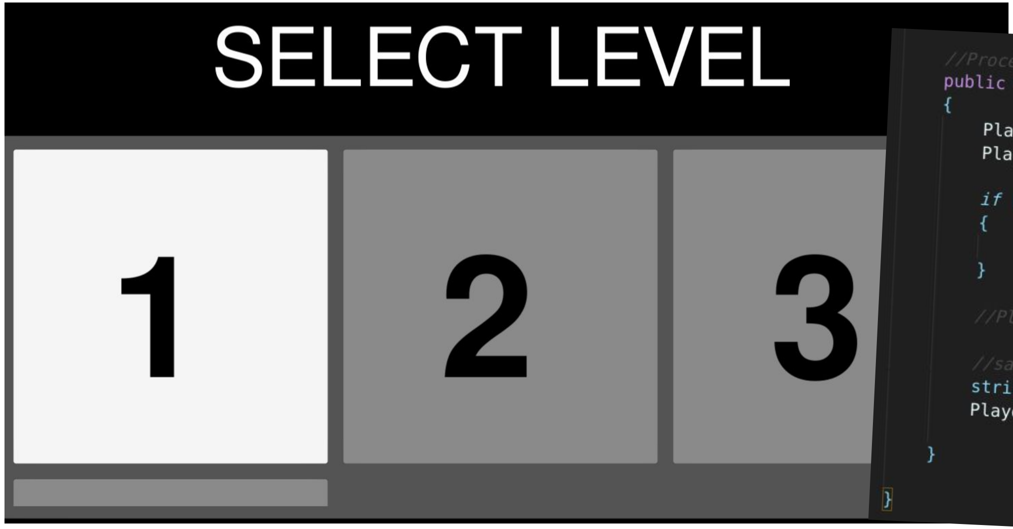


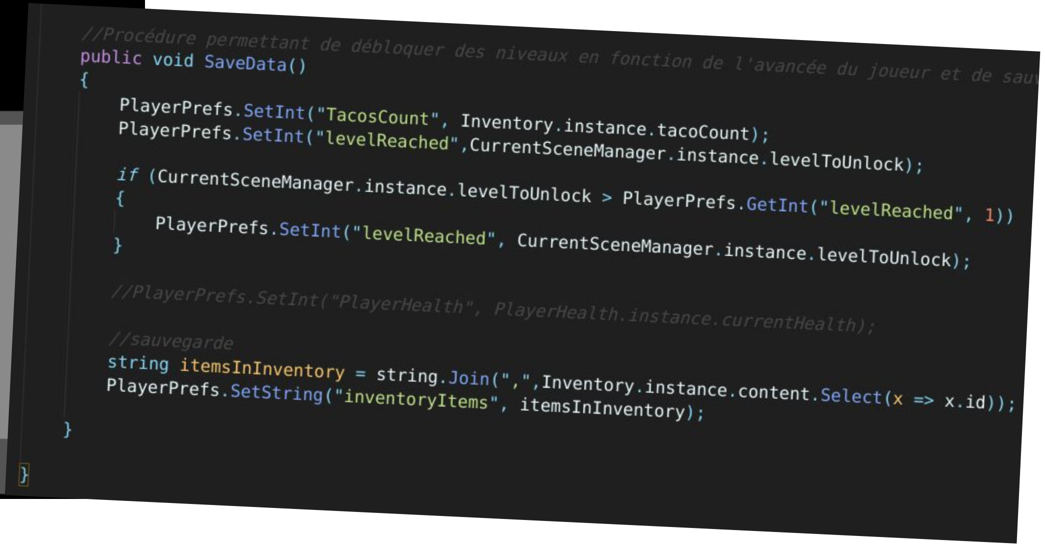
```
AudioManager.instance.PlayClipAt(hitSound, transform.position);
```



```
public AudioClip hitSound;
```

Gestion de niveaux/Sauvegarde





```
//Procédure permettant de débloquer des niveaux en fonction de l'avancée du joueur et de sauv
public void SaveData()
{
    PlayerPrefs.SetInt("TacosCount", Inventory.instance.tacoCount);
    PlayerPrefs.SetInt("levelReached",CurrentSceneManager.instance.levelToUnlock);

    if (CurrentSceneManager.instance.levelToUnlock > PlayerPrefs.GetInt("levelReached", 1))
    {
        PlayerPrefs.SetInt("levelReached", CurrentSceneManager.instance.levelToUnlock);
    }

    //PlayerPrefs.SetInt("PlayerHealth", PlayerHealth.instance.currentHealth);

    //sauvegarde
    string itemsInInventory = string.Join(",",Inventory.instance.content.Select(x => x.id));
    PlayerPrefs.SetString("inventoryItems", itemsInInventory);
}

}
```

Bilan et Difficultés rencontrées

MERCI POUR VOTRE ATTENTION