



Table des matières

I. Introduction

II. Répartition des tâches

1. Site Web
2. Gameplay
3. Intelligence Artificielle
4. Graphismes
5. Interface
6. Audio

III. Récit de la réalisation

1. Impression globale
2. Impressions personnelles

IV. Conclusion

I. Introduction

Ce document a pour objectif de montrer l'avancement de notre projet, conformément aux tâches distribuées parmi les différents membres du développement de notre jeu vidéo *Kick Out!*. Notre entreprise *Les 6 Étrangers* est composée de quatres élèves en SUP, la première année de cycle préparatoire à EPITA : Éthan CARROUÉE, Yann FERNANDEZ PUIG, Matteo FLEBUS, Bao-Ahn TRAN.

Tout au long de ce rapport, nous allons détailler les différents avancements ayant été réalisés liés au site web du projet, à la jouabilité, à l'intelligence artificielle, au réseau aux graphismes ou encore à l'audio. Chacun de nous donnera ensuite son impression sur la réalisation du projet.

II. Répartition des tâches

Au début du quatrième bimestre, notre groupe a perdu un de ses membres : Qaiser ALKUBATI. Ainsi, nous avons dû revoir la répartition des tâches au sein de l'équipe pour compenser cette perte. Nous vous présentons ci-dessous notre nouveau tableau de répartition des tâches.

Tâches	Ethan	Yann	Matteo	Bao-Anh
Réseau	R	S		
Design				
World Design		S	R	
Character Design	R		S	
Menu Principal	S		R	
Selection des personnages	S			R
Affichage In Game	R		S	
Fin de partie	S	R		
Son				
Musiques	S		R	
FX			S	R
Programmation				
Déplacements			S	R
Attaques		S		R
IA		R		S
Site web		S	R	

FIGURE 1 - Tableau de répartition des tâches

1. Site web

Le site web de notre projet est essentiel. En effet, sur celui-ci nous pouvons retrouver divers aspects de notre entreprise. Nous retrouvons tout d'abord une page de présentation de notre jeu *Kick Out!* ainsi que des différents combattants qui sont disponibles. On retrouve également une présentation de l'équipe générale et des membres. Dans une autre section, tous les documents retraçant l'avancement de projet sont téléchargeables, tels que le cahier des charges ou les rapports de soutenance. Finalement, dans la dernière page de notre site web, nous pouvons télécharger notre jeu *Kick Out!*.

Pour le thème du site web, nous avons voulu respecter les couleurs de notre entreprise mais également ajouter une touche de notre jeu. Ainsi, nous avons opté pour des couleurs oranges et noires renvoyant aux logos de l'entreprise et du jeu vidéo. La police du site est dans un style pixélisée pour rappeler le côté rétro que nous voulons transmettre à travers notre produit.

Pour la réalisation du site, nous avons préféré le coder en entier plutôt que d'utiliser des logiciels facilitant la création de pages web. En effet, avec ces outils le développement d'un site web se fait plus rapidement et aisement mais l'utilisation des langages de programmation *HTML* et *CSS* permettent une plus grande créativité quant à la réalisation du site.

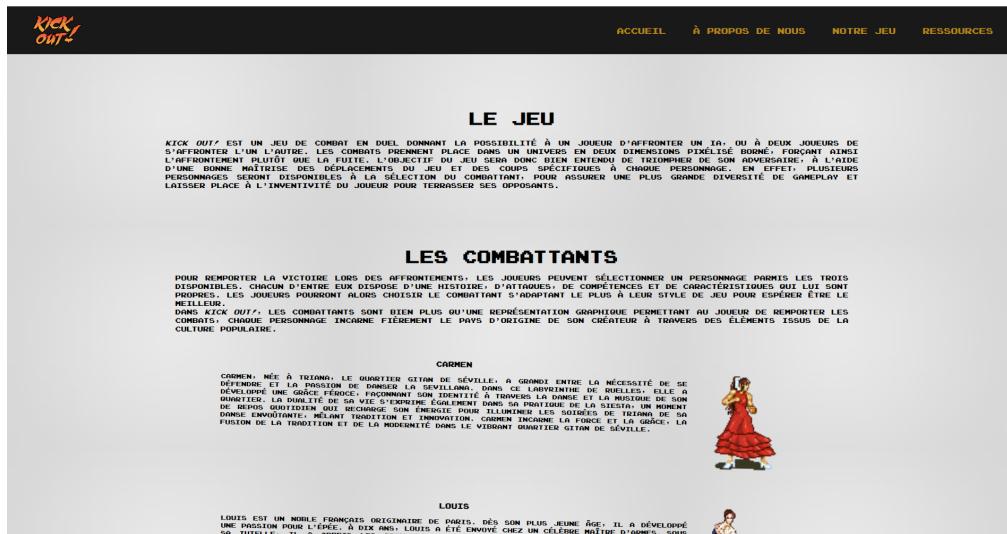


FIGURE 1 - Accueil du site

KICK OUTS

ACCUEIL À PROPOS DE NOUS NOTRE JEU RESSOURCES

NOTRE ENTREPRISE

LE STUDIO CRÉATEUR DE KICK OUTS SE NOME : «LES 6 ÉTRANGERS». EN EFFET, LE GROUPE EST COMPOSÉ DE MEMBRES AYANT TOUS UNE CULTURE QUI LEUR EST INTRINSÈQUE, DE PLUS, CHAQUE MEMBRE A SUIVI UN PARCOURS DIFFÉRENT AU COURS DE SES ÉTUDES, TANT AU NIVEAU DES SPÉCIALITÉS CHOISIES AU LYCÉE QU'AU NIVEAU DES OPTIONS ET DES PROJETS EN DEHORS DES COURS.

NOUS COMPTONS SUR CETTE PLURALITÉ D'EXPÉRIENCES ET DE CULTURES POUR NOUS APPORTER UNE OUVERTURE INTERNATIONALE; UNE GRANDE CAPACITÉ D'ADAPTATION, QUI PERMET DE FAIRE face à DES CHANGEMENTS SURNATURELS. UNE AUTRE VALEUR TOUCHE PERSONNELLEMENT CHAQUE Membre, C'EST LA COMMUNICATION, QUI PASSERA PAR DES RÉUNIONS, DES ÉCHANGES DANS LA RÉALITÉ OU PAR MESSAGES ET DES SONDAGES EN CAS DE DÉBATS.

LES MEMBRES

BAO-ANH TRAN VANN FERNANDEZ PUIG

IL EST UN MEMBRE DU GROUPE MAIS AUSSI DÉSIGNÉ COMME CHEF DE CELUI-CI. SON RÔLE EST DE RASSEMBLER SES ÉQUIPPIERS MUSICAUX ET DE VEILLER AU BON AVancement DU PROJET, SOUFFRANT D'UNE ATMOSPHÈRE PROFESSIONNELLE ET BIENVEILLANTE, IL CHERCHE À DÉVELOPPER LE POTENTIEL DE CHAQUE Membre AVANT EU UNE APPROCHE À L'ORGANISATION D'ÉVÉNEMENTS. LES ATTENDUS DE GROUPE SONT ALORS DÉJA IDENTIFIÉS.

SON OBJECTIF EST APporter LA BONNE HUMEUR AU SEIN DE SES COÉQUIPIERS AVEC SES NOMBREUSES BLAGUES HILARANTES, DOTÉ D'UN SENS DE L'HUMOUR ET D'UN SENS DE L'OBSTACULE, CE QUI PERMETTANT AU PROJET DE SE DISTINGUER DES AUTRES. EN TANT QU'UN DÉVELOPPEUR PASSIONNÉ D'INFORMATIQUE, LE DÉVELOPPEMENT DE CE PROJET LIERAIT LA CHANCE D'OBTENIR DE NOUVELLES COMPÉTENCES DANS LE DOMAINE, MAIS AUSSI DE PARTAGER SON SAVOIR AVEC SES COÉQUIPIERS.

ETHAN CARROUËT MATTEO FLEBUS

ETHAN, LE PLUS GRAND DU GROUPE, VISE À TRAVAILLER EN ENTRE EN COLLABORATION AVEC SES COÉQUIPIERS POUR EXPÉRIENCE DES SOLUTIONS INNOVANTES ET LES METTRE EN ŒUVRE DE MANIÈRE EFFICACE. SON OBJECTIF ULTIME EST DE CONTRIBUER À LA RÉALISATION DES OBJECTIFS SPÉCIFIQUES DU PROJET ET LE MAINTIEN D'UNE ATMOSPHÈRE DE TRAVAIL POSITIVE, OUVERT

MATTEO EST UN MEMBRE DU GROUPE QUI CONSIDÈRE COMME SON PRINCIPAL MOTIVATION LA RÉALISATION DE CE PROJET, SA MOTIVATION ET SA DÉTERMINATION. IL CONSIDÈRE CELUI-CI COMME UN DÉFI À RELEVER, QUI APPORTERA COMPÉTENCES TECHNIQUES ET EXPÉRIENCE DANS LA GESTION D'UN PROJET ET QUI RENFORCERA SON HABILETÉ À TRAVAILLER EN COORDINATION

FIGURE 2 - Présentation de l'entreprise et des membres

KICK OUTS

ACCUEIL À PROPOS DE NOUS NOTRE JEU RESSOURCES

RESSOURCES

ICI, VOUS POUVEZ TÉLÉCHARGER LES RESSOURCES DONT VOUS AVEZ BESOIN QUI SONT LIÉES À NOTRE ENTREPRISE. CLIQUEZ SUR CE QUI VOUS INTÉRESSE !

[CAHIER DES CHARGES](#)

[RAPPORT DE SOUTENANCE](#)

FIGURE 3 - Section pour le téléchargement des ressources



FIGURE 4 - Section pour le téléchargement du jeu

2. Gameplay

Dans les jeux de combats, les deux aspects les plus importants sont les déplacements du joueur qui lui permettent de se tenir à distance de l'adversaire mais aussi de se rapprocher de lui pour lui infliger des dommages. L'autre fonctionnalité dont les jeux de ce style ne peuvent se passer sont les attaques. En effet, que serait un jeu de combat sans celles-ci ? Elles apportent l'opportunité aux joueurs de mettre hors de jeu leur adversaire en les combinants avec les mouvements des personnages.

Déplacements

Dans un premier temps, afin que notre jeu soit le plus réaliste possible, nous avons dû rajouter certains composants à nos objets en jeu pour qu'ils puissent respecter les lois de la physique fondamentale, comme la collision du combattant avec le sol, l'adversaire ou bien l'effet de la gravité sur les personnages. Pour la gestion des collisions avec l'environnement, nous avons ajouté une boîte de collisions nommée *CapsuleCollider2D* à chacun de nos combattants. Selon le personnage, nous avons dû paramétriser la hauteur et la largeur puisque certains de nos combattants sont plus volumineux que d'autres.

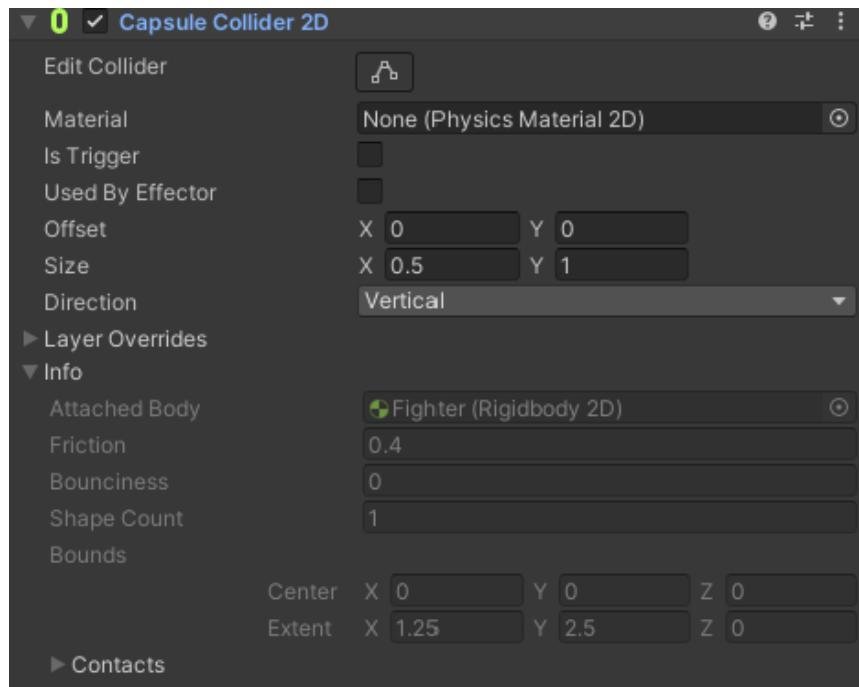


FIGURE 5 - Boîte de collisions

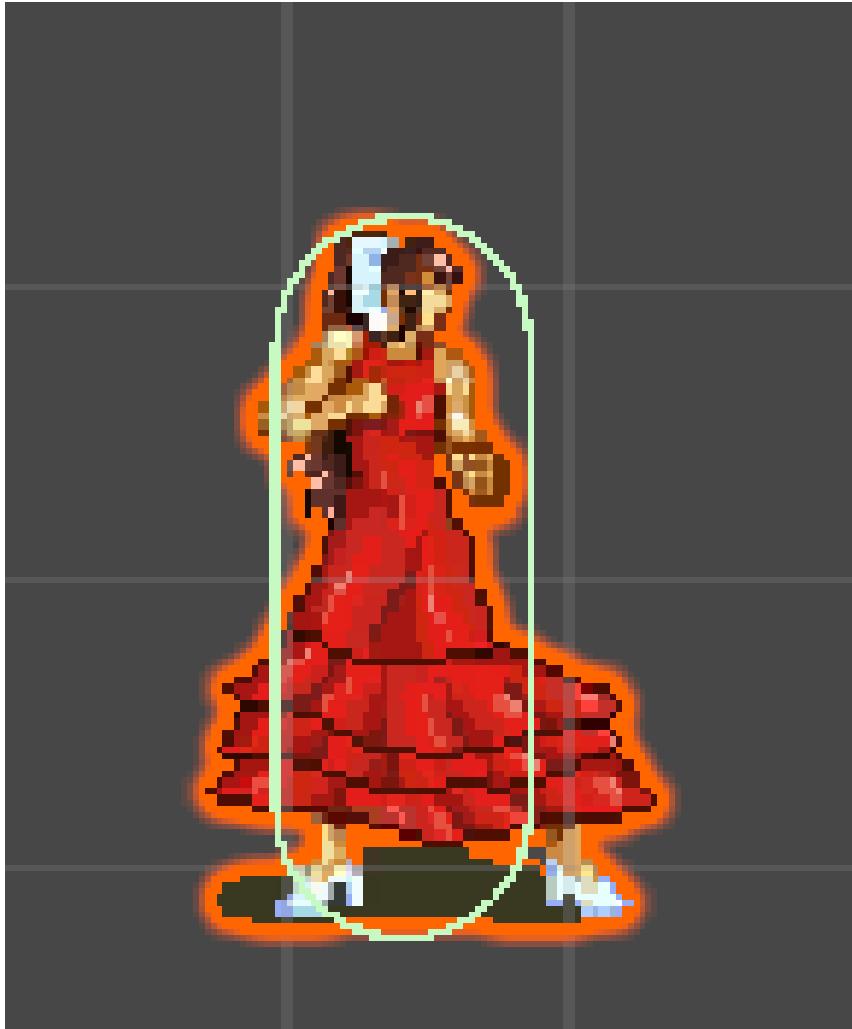


FIGURE 6 - Représentation de la boîte de collisions de Carmen

Pour la gestion des lois de la physique sur nos éléments dans notre jeu, *Unity* présente un composant nommé le *Rigidbody2D* qui permet aux objets d'être soumis à la gravité mais aussi aux forces qui sont grandement utiles lors des déplacements.

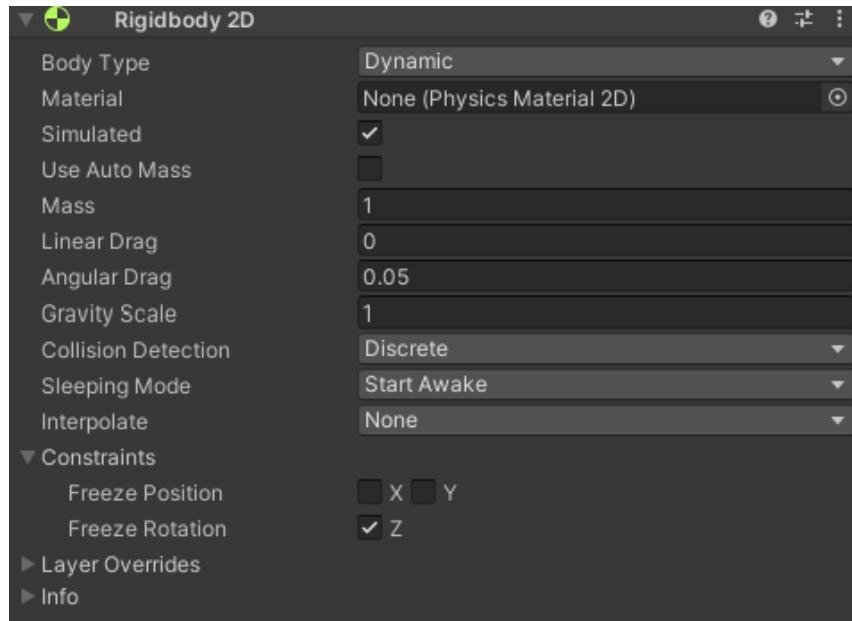


FIGURE 7 - Rigidbody2D

Maintenant que les champions disposent des composants nécessaires au bon déroulement du jeu, nous avons développé le mouvement horizontal du joueur. Pour cela, nous récupérons les saisies du joueur puis nous utilisons ces saisies pour augmenter la vitesse sur l'axe horizontal de notre *Rigidbody2D*.

```
horizontalInput = Input.GetAxis("Horizontal");
```

FIGURE 8 - Code pour récupérer la saisie du joueur

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

**FIGURE 9 - Code pour appliquer la vitesse horizontale au
*Rigidbody2D***

Dans les déplacements des joueurs, nous avons dû faire face à une première difficulté : de quelle façon devait réagir les combattants reculaient. Étant donné que notre principale inspiration nous vient des premiers jeux de combats développés, nous avons décidé que les personnages se feraient toujours face entre eux. Ceci permet ainsi une plus grande rivalité entre les joueurs. Nous avons codé une fonction qui permet aux combattants de se regarder constamment

entre eux même lorsque l'un passe derrière l'autre. Pour cela, nous comparons les coordonnées sur l'axe des abscisses du champions et si elles sont supérieures à celle de l'autre personnage, alors nous le retournons.

```
4 references
public void LookAtEnemy()
{
    if (stats.center.transform.position.x > enemy.GetComponent<FighterStats>().center.transform.position.x)
    {
        spriteRenderer.flipX = true;      You, 1 second ago * Uncommitted changes
    }
    else
    {
        spriteRenderer.flipX = false;
    }
}
```

FIGURE 10 - Fonction permettant aux combattants de se faire face

Une fois les déplacements horizontaux codés, nous nous sommes penchés sur le saut du personnage. Dans le système de saut, une des priorités est de vérifier que le combattant est au sol pour éviter que le joueur puisse sauter à l'infini. Pour cela, nous créons un cercle invisible à la vue à partir d'un point, au niveau des pieds du personnage, et avec un certain rayon. Puis on regarde si l'objet représentant le sol est contenu dans ce cercle. Si c'est le cas, on considère que le combattant touche le sol et est apte à sauter.



FIGURE 11 - Image représentant l'object pour détecter le saut

```
void Jump()
{
    //Cette fonction permet de faire sauter le combattant en appuyant sur la touche espace (modifiable)

    //To jump, the player must press the space bar and be grounded
    if (isJumping && isGrounded)
    {
        animator.SetBool("IsJumping", true);
        rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
        isJumping = false;
        isGrounded = false;
    }

    if (rb.velocity.y >= 0) //if the fighter is going up in is jumping, the gravity is normal
    {
        rb.gravityScale = gravityScale;
    }
    else //when the fighter is falling, increases the gravity so the jump looks more realistic
    {
        rb.gravityScale = fallingGravityScale;
    }
}
```

FIGURE 12 - Fonction de saut

Mouvements défensifs

Pour diversifier le style de jeu des joueurs, nous avons décidé d'implémenter deux mouvements défensifs : l'accroupissement, ou *crouch* en anglais, et le blocage.

L'accroupissement est un mouvement qui diminue la taille du combattant. Néanmoins, celle-ci n'est pas uniquement modifiée visuellement mais la hauteur et la largeur de la boîte de collisions pour éviter que le joueur ne puisse subir des dégâts lorsque l'adversaire ne peut pas le frapper.

Nous avons décidé de ne pas définir un limite de temps pendant lequel le joueur est accroupi, le joueur a juste à maintenir une touche enfoncee pour que le combattant s'accroupisse. En contre-partie, le joueur ne peut ni se déplacer, ni attaquer tant qu'il est accroupi pour empêcher l'anti-jeu potentiel.

Que serait un jeu de combat sans une mécanique permettant de bloquer ? Pour notre part, nous avons voulu imiter en quelque sorte la méthode de blocage sur les jeux *Smash Bros.* Dans ceux-ci, le blocage a un temps limité de 2,5 secondes et qui doit se recharger après chaque utilisation. Pour éviter qu'un joueur bloque pendant tout le long de la partie, nous avons instauré un petit délai d'une seconde lorsque la durée de blocage atteint 0. Le temps restant au joueur pour effecter l'action de bloquer est indiqué par une barre jaune qui se situe sous la barre de vie.

La fonctionnalité de blocage permet au combattant de ne subir aucun dommage lorsqu'il reçoit un coup. Cependant le temps restant du blocage est diminué d'un certain temps selon l'attaque et le personnage attaquant.

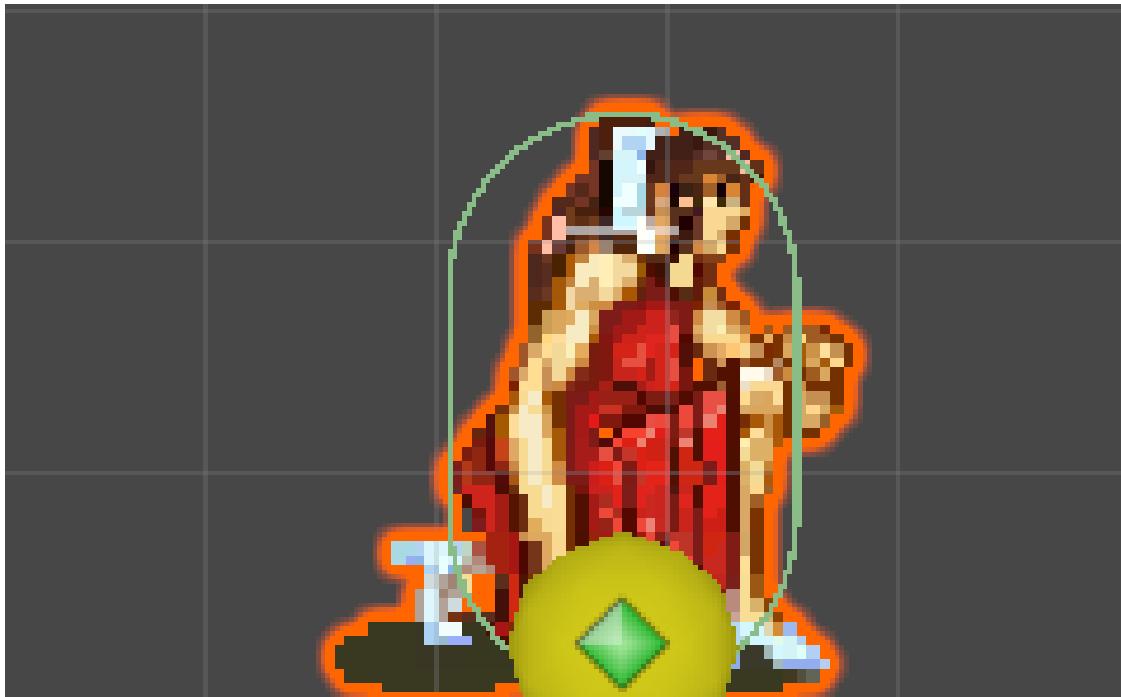


FIGURE 13 - Animation d'accroupissement de Carmen

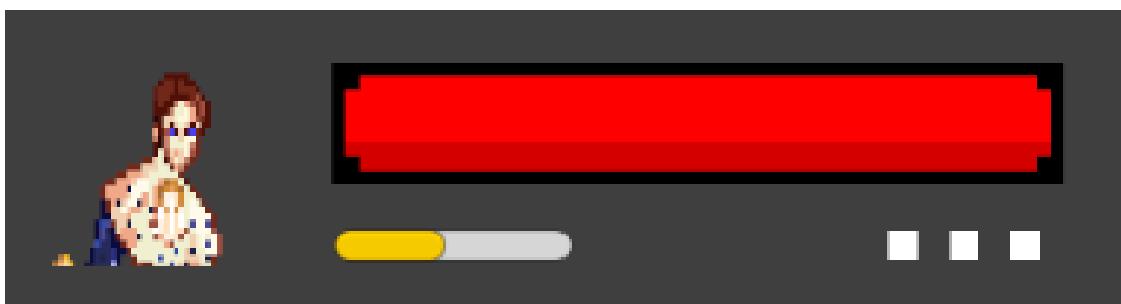


FIGURE 14 - Barre de temps restant de blocage

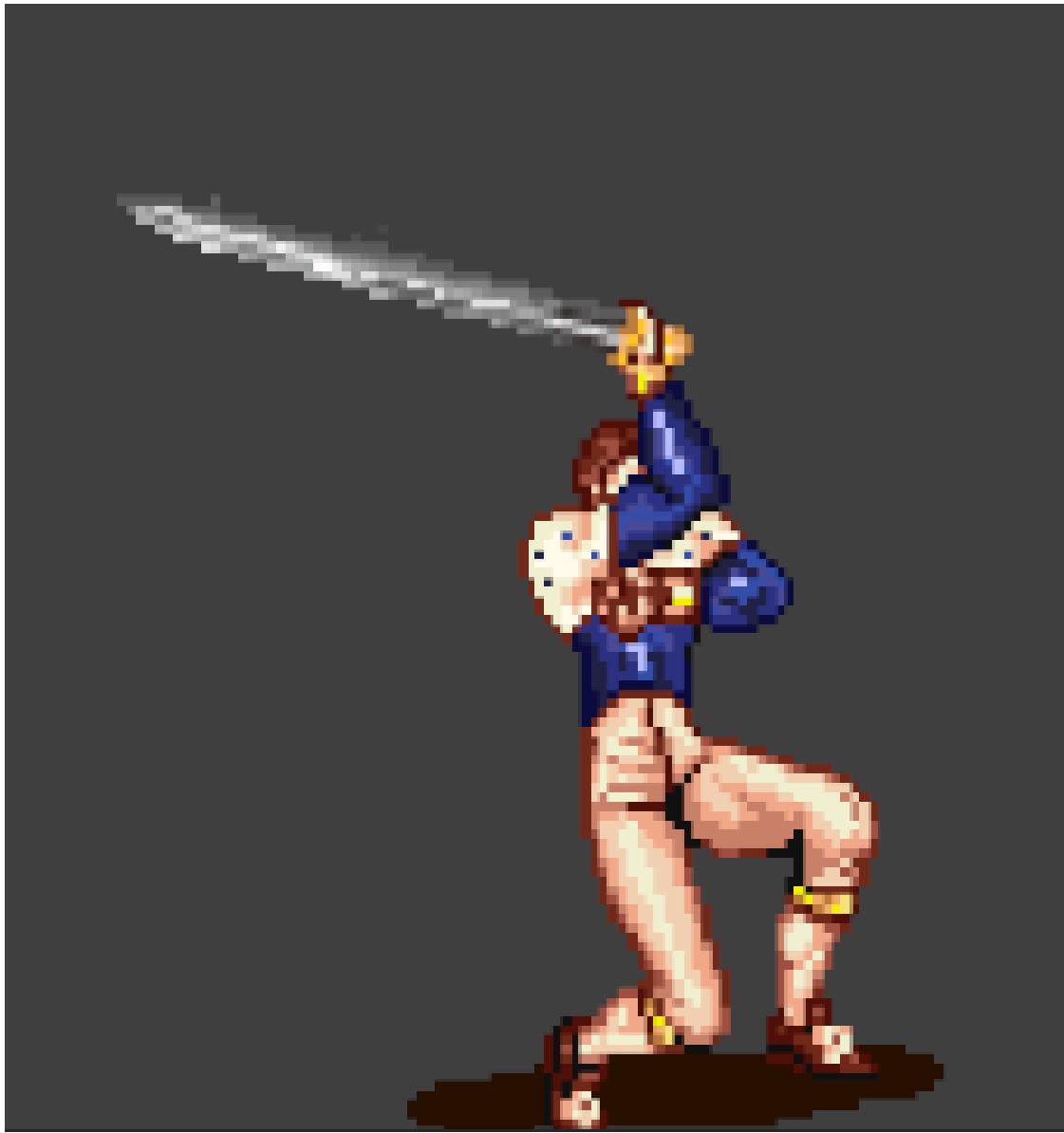


FIGURE 15 - Blocage de Louis

Combat

Dans notre jeu, un des aspects le plus important est le combat. Nous avons implémenté des attaques ainsi qu'un système de points de vie propre à chaque joueur, qui lorsqu'ils tombent à 0, marquent la fin de la manche ou du combat.

Nous avons dans un premier temps développé une barre de points de vie représentant la vie du personnage et associé le code qui permet de la faire diminuer au fur et à mesure que le combattant subit des dégâts.



FIGURE 16 - Barre de points de vie

```
public void SetMaxHealth(float health)
{
    slider maxValue = health;
    slider value = health;
}

//Whenever the health of the character is updated
0 references
public void SetHealth(float health)
{
    slider value = health;
}
```

FIGURE 17 - Fonctions permettant la gestion de la barre de vie

Après avoir implémenté les points de vie de chaque personnage ainsi que la barre de vie permettant de les visualiser, nous avons mis en place les attaques des combattants. Chaque personnage dispose d'une attaque basique et d'une attaque spéciale. Afin d'éviter que les joueurs ne puissent pas lancer plusieurs at-

taques, nous avons mis en place un système de compte à rebours, ou *timer*, entre chaque attaque qui est plus ou moins important selon le combattant et le type de l'attaque : les spéciales ont un délai plus long avant de pouvoir être utilisées à nouveau. Évidemment, ce compte à rebours vise à pouvoir équilibrer le jeu, on ajoute donc l'impact de l'action à la balance.

```
aCD -= Time.deltaTime;
aCDSpe -= Time.deltaTime;

if(Input.GetButtonDown("Punch P1") && aCD <= 0)
{
    animator.SetTrigger("Attack");
    aCD = stats.attackCooldown;
}

if(Input.GetButtonDown("Special P1") && aCDSpe <= 0)
{
    animator.SetTrigger("Special");
    aCDSpe = stats.attackCooldownSpe;
}
```

FIGURE 18 - Implémentation du délai entre les attaques

Les variables *aCD* et *aCDSpe* représentent respectivement les délais pour l'attaque de base et l'attaque spéciale de chaque personnage. Avant que le joueur ne puisse lancer une attaque, il faut que le compte à rebours correspondant soit à 0. Lorsque la touche d'attaque est activée, le *timer* est réinitialisé.

Lorsque le joueur attaque, nous devons vérifier s'il touche l'enemi pour que celui-ci subisse des dommages ou non. Pour cela, quand une attaque est déclenchée, le code génère un rectangle fictif devant le combattant avec pour longueur sa portée d'attaque qui est différente selon les personnages. Si ce rectangle détecte que l'enemi est à l'intérieur alors, il subit les dégâts correspondant au type de

l'attaque.

```
Vector3 center = new Vector3(attackPoint.position.x, attackPoint.position.y, 0);
Vector3 size = new Vector3(stats.attackRange * 2, 0.25f, 0);

Collider2D[] enemiesHitted = Physics2D.OverlapBoxAll(center, size, 0, enemyLayer);
```

FIGURE 19 - Code pour générer le rectangle fictif

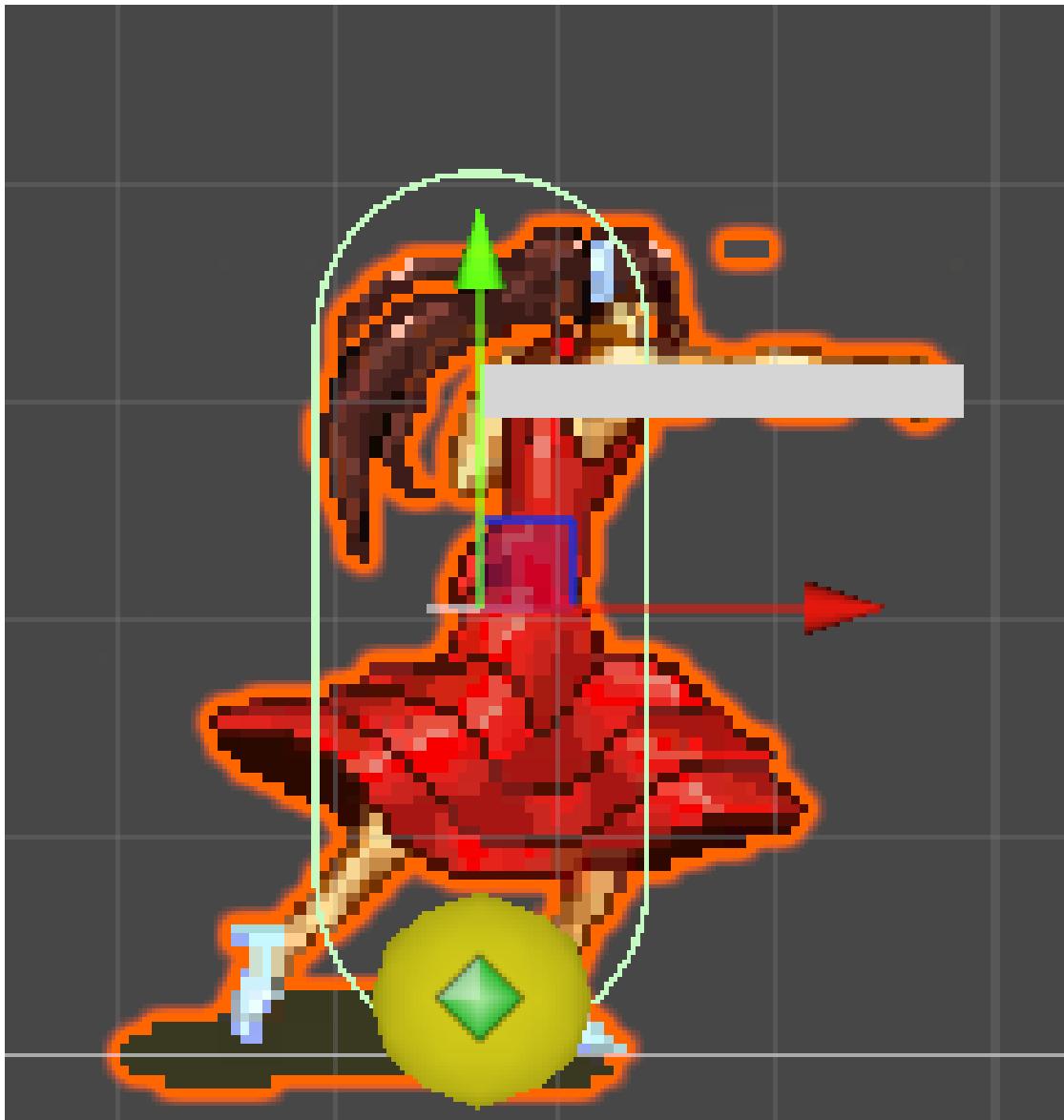


FIGURE 20 - Représentation du rectangle en image

```

0 references
void Attack()
{
    isAttacking = true;

    move.horizontalInput = 0f;

    bool miss = true;

    StartCoroutine(MyFunctionAfterDelay(punch.length));

    //Detect the enemies in range
    //OverlapCircleAll creates a 'circle' around a point (1st parameter) with a certain radius (2nd parameter) and you can apply layers (3rd parameter)
    Vector3 center = new Vector3(attackPoint.position.x, attackPoint.position.y, 0);
    Vector3 size = new Vector3(attackRange * 2, 0.25f, 0);

    Collider2D[] enemiesHitted = Physics2D.OverlapBoxAll(center, size, 0, enemyLayer);

    Debug.Log(enemiesHitted.Length);

    //Damage the enemy
    foreach(var enemy in enemiesHitted)
    {
        enemy.GetComponent<Fighter>().TakeDamage(stats.damage);

        if (enemy.tag == "Player" || enemy.tag == "Player1" || enemy.tag == "Player2")
        {
            if (!enemy.GetComponent<PlayerMovement>().isBlocking)
            {
                enemy.GetComponent<Fighter>().TakeDamage(stats.damage);
            }
            else
            {
                enemy.GetComponent<FighterStats>().blockCD -= stats.reduceCD;
            }
        }
        else if (enemy.tag == "AI")
        {
            if (!enemy.GetComponent<AIMovement>().isBlocking)
            {
                enemy.GetComponent<Fighter>().TakeDamage(stats.damage);
            }
        }
        miss = false;

        soundManager.PlaySFX(stats.punchSound);
    }

    if (miss)
    {
        soundManager.PlaySFX(stats.missShot);
    }
}

```

FIGURE 21 - Fonction pour l'attaque basique des personnages

Les combats se déroulent en trois *rounds*, manches en français, gagnants. Pour gagner une manche, il y a deux options : soit le compte à rebours tombe à 0 indicant que la manche est finie. C'est alors le combattant avec le plus de vie restante qui remporte le *round*. Sinon si un des deux personnages voit sa vie à 0 alors le champion ayant mit K.O. son enemi remporte la manche. Lorsque le joueur remporte la manche, un carré représentant la manche s'illume pour indiquer la victoire. Une fois trois manches remportées, le joueur est désigné vainqueur du combat.



FIGURE 22 - Scène de combat



FIGURE 23 - Carré illuminé indiquant la victoire du *round*

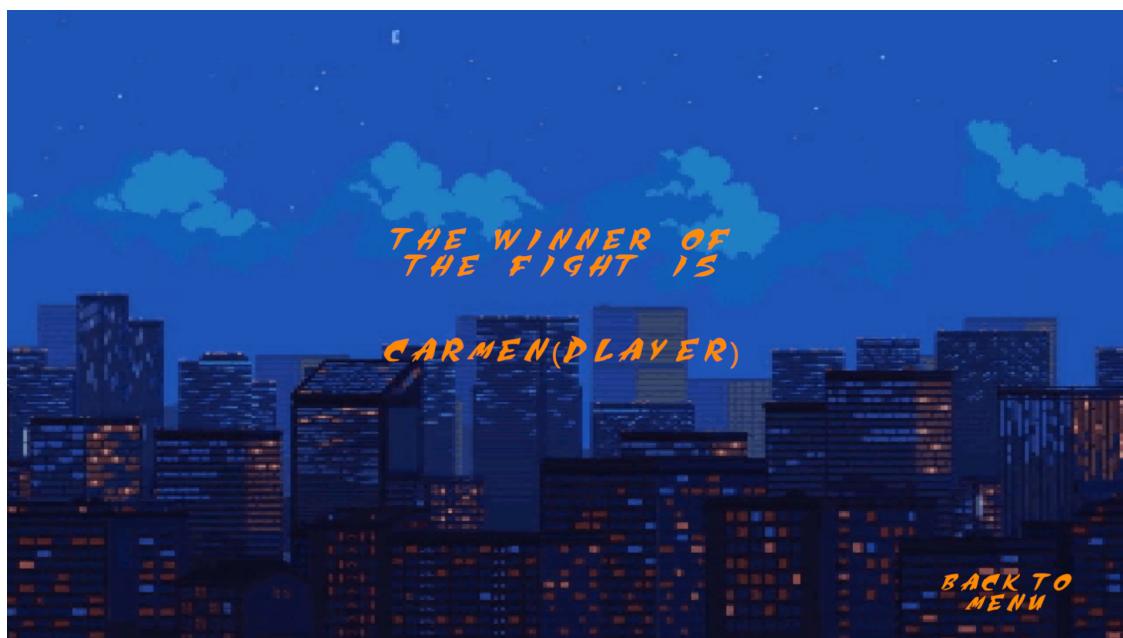


FIGURE 23 - Scène de victoire

Sélection des personnages

Le point mis en avant par notre projet et l'entreprise est le choix qu'apporte notre jeu dans le choix des combattants puisque chacun d'entre eux est unique : il a une histoire, des attaques, une représentation qui lui sont propres. Ainsi, avant le début de chaque combat, les joueurs ont la possibilité de choisir un combattant parmi les trois actuellement disponibles.



FIGURE 24 - Écran de sélection des personnages

Comme vous pouvez le remarquer, le fond de l'écran de la sélection des personnages est une carte de la planète Terre. À travers ceci, nous avons souhaité apporter une touche d'originalité qui permet aux joueurs de voir de quel pays issu leur combattant pour qu'ils puissent commencer à découvrir les champions avant même le début du combat.

Une fois le combattant choisi, le joueur peut alors cliquer sur le boutton *Start* pour lancer le combat. Pour que le personnage sélectionné soit celui qui soit joué en combat, nous avons décidé de passer par une méthode de chargement des données correspondant au champion. Pour cela, avant que le combat démarre, on injecte dans l'object représentant le combattant les statistiques de celui qui a été choisi par le joueur. Avec ce fichier de données, nous récupérons

toutes les valeurs propres à notre champion tel que les points de vie, les dégats d'attaque, la vitesse de déplacement, etc.

```
case "Bob Un":
    spriteRenderer.sprite = Resources.Load<Sprite>("BaseSprites/Bob Un");
    BobUnStats bobUnStats = AddComponentIfNotExists<BobUnStats>(p);
    bobUnStats.Initialize();
    SetAnimatorAndAttack(p, "Animation/BobUn/Bob Un", "Animation/BobUn/bobUnAttack", "Animation/BobUn/bobUnSpecial");
    SetPlayerPosition(p, bobUnStats);
    SetFighterPoints(p, bobUnStats);
    break;
```

FIGURE 25 - Exemple de la fonction de chargement pour Bob Un

Les fonctions dont le nom commence par le mot *Set* permettent de mettre à la position correcte le joueur mais aussi le point permettant de savoir si le joueur touche le sol ou non (cf. la fonction de saut) et le point à partir duquel le rectangle pour l'attaque est généré (cf. la fonction d'attaque). La fonction *SetAnimatorAndAttack* permet quant à elle de charger les animations correspondant au personnage sélectionné.

```
private void SetAnimatorAndAttack(GameObject player, string animatorPath, string attackPath, string specialPath)
{
    Animator animator = player.GetComponent<Animator>();
    if (animator == null)
    {
        animator = player.AddComponent<Animator>();
    }
    animator.runtimeAnimatorController = Resources.Load<RuntimeAnimatorController>(animatorPath);

    Attack playerAttack = player.GetComponent<Attack>();
    if ([playerAttack == null]) You, yesterday * dj
    {
        playerAttack = player.AddComponent<Attack>();
    }
    playerAttack.punch = Resources.Load<AnimationClip>(attackPath);
    playerAttack.special = Resources.Load<AnimationClip>(specialPath);
}
```

FIGURE 26 - Fonction *SetAnimatorAndAttack*

```
private void SetFighterPoints(GameObject player, FighterStats stats)
{
    stats.attackPoint.transform.position = new Vector3(player.transform.position.x + stats.attackPointPos.x, player.transform.position.y + stats.attackPointPos.y * 5, stats.attackPointPos.z);
    stats.groundCheck.transform.position = new Vector3(player.transform.position.x + stats.groundCheckPointPos.x, player.transform.position.y + stats.groundCheckPointPos.y * 5, player.transform.position.z);
    stats.center.transform.position = new Vector3(player.transform.position.x + stats.fighterCenter.x, player.transform.position.y + stats.fighterCenter.y, player.transform.position.z + stats.fighterCenter.z);

    CapsuleCollider2D capsuleCollider = player.GetComponent<CapsuleCollider2D>();
    if (capsuleCollider == null)
    {
        capsuleCollider = player.AddComponent<CapsuleCollider2D>();
    }
    capsuleCollider.size = new Vector2(stats.width, stats.height);
}
```

FIGURE 27 - Fonction *SetFighterPoints*

```
private void SetPlayerPosition(GameObject player, FighterStats stats)
{
    if (gameMode == "tutorial")
    {
        player.transform.position = new Vector3(stats.spawnPoint.x, stats.spawnPoint.y, stats.spawnPoint.z);
    }
    else if (gameMode == "solo")
    {
        if (player.tag == "Player")
        {
            player.transform.position = new Vector3(stats.spawnPoint.x, stats.spawnPoint.y, stats.spawnPoint.z);
        }
        else if (player.tag == "AI")
        {
            player.transform.position = new Vector3(-stats.spawnPoint.x, stats.spawnPoint.y, stats.spawnPoint.z);
        }
    }
    else if (gameMode == "duel")
    {
        if (player.tag == "Player1")
        {
            player.transform.position = new Vector3(stats.spawnPoint.x, stats.spawnPoint.y, stats.spawnPoint.z);
        }
        else if (player.tag == "Player2")
        {
            player.transform.position = new Vector3(-stats.spawnPoint.x, stats.spawnPoint.y, stats.spawnPoint.z);
        }
    }
}
```

FIGURE 28 - Fonction *SetPlayerPosition*

3. Multijoueur

L'implémentation du multijoueur s'est basée sur les anciens jeux de combat tels que *Street Fighter II* ou *Mortal Combat* où les deux joueurs jouent sur la même machine et chacun d'entre eux dispose de boutons spécifiques pour jouer. Ainsi dans *Kick Out!* lorsque les joueurs décident de se battre entre eux, ils sélectionnent alors le mode de jeu multijoueur qui les envoie sur la page de sélection.

Le choix des combattants se fait tour par tour, d'abord le joueur 1 puis le 2. Une fois que les deux joueurs ont sélectionné leur champion, ils peuvent alors lancer le combat qui se déroule de façon normal, en 3 manches gagnantes.

L'une des parties les plus compliquées concernant le multijoueur a été de séparer les touches pour éviter que lorsqu'une touche est pressée celle-ci agisse sur les deux personnages. Pour cela, nous avons dû modifier l'*Input System*, le système de saisie, de *Unity*.

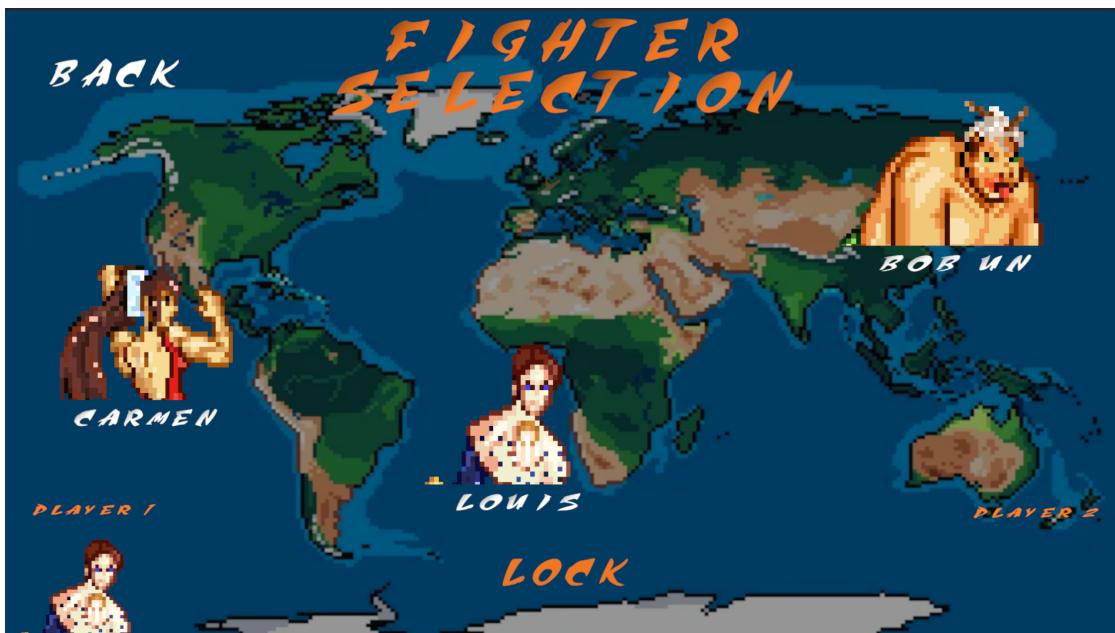


FIGURE 29 - Sélection du multijoueur

4. Intelligence Artificielle

Pour l'implémentation de l'intelligence artificielle, nous nous sommes basés sur les fonctions de déplacements et d'attaques des joueurs. Néanmoins, nous avons dû adapter le code pour que notre intelligence artificielle se déplace d'elle-même et fasse ses propres choix.

Pour cela, nous avons commencé par définir intervalle de temps pendant lequel l'IA décide de la prochaine action qu'elle devra exécuter. Ce choix de décision se fait de façon aléatoire. Nous générerons un nombre entre 0 et 10 et selon le résultat obtenu l'intelligence effectue telle ou telle opération. Cependant, toutes les décisions possibles n'ont pas la même probabilité d'être effectuées. En effet, nous ne souhaitons pas que l'intelligence artificielle passe l'entièreté du combat à avancer de quelques pixels puis d'attaquer, ceci la rendrait beaucoup trop prévisible pour le joueur et le combat en deviendrait ennuyant.

Pour les déplacements de l'intelligence artificielle, nous avons également voulu que ceux-ci soient peu prévisibles pour les joueurs et avons donc implémenté une fonction, en plus de celle qui lui permet de se déplacer vers le joueur, qui rend les mouvements de l'IA imprévisibles. Les directions que peut prendre l'intelligence se font de façon aléatoires et pendant une durée qui l'est aussi. Ainsi, lorsque l'IA s'éloigne du joueur celui-ci se retrouve obligé de l'approcher s'il veut l'attaquer rendant le combat plus dynamique.

```
void MakeDecision()
{
    float distanceToPlayer = Vector2.Distance(transform.position, playerTransform.position);

    if (distanceToPlayer < attack.attackRange)
    {
        You, 6 days ago • correction anime louis et carmen, fin de round ...
        int randomAction = Random.Range(0, 10);

        if (randomAction < 5)
        {
            PerformAttack();
        }
        else if (randomAction < 9)
        {
            PerformBlock();
        }
        else
        {
            PerformSpecial();
        }
    }
    else
    {
        int randomMoveAction = Random.Range(0, 10);

        if (randomMoveAction < 6)
        {
            MoveTowardsPlayer();
        }
        else
        {
            MoveRandomly();
        }
    }
}
```

FIGURE 30 - Fonction de prise de décision de l'intelligence artificielle

```

1 reference
void HandleMovement()
{
    float distance = player.transform.position.x - transform.position.x;

    if (Mathf.Abs(distance) > 1f)
    {
        if (distance > 0)
        {
            rb.velocity = new Vector2(moveSpeed, rb.velocity.y);
        }
        else
        {
            rb.velocity = new Vector2(-moveSpeed, rb.velocity.y);
        }
    }
    else
    {
        rb.velocity = new Vector2(0, rb.velocity.y);
    }
}

```

FIGURE 31 - Fonction déplacement de l'IA

```

1 reference
void HandleBlocking()
{
    if (blockCD > 0 && !isBlockCooldown)
    {
        isBlocking = true;
        blockCD -= Time.deltaTime;
        if (blockCD < 0)
        {
            blockCD = 0;
            isBlockCooldown = true;
        }
    }
    else
    {
        isBlocking = false;
        blockCD += Time.deltaTime * 0.5f;
        if (blockCD > stats.blockCD) blockCD = stats.blockCD;
    }
}

```

FIGURE 32 - Fonction de blocage pour l'IA

```
1 reference
void MoveRandomly()
{
    moveTimer -= Time.deltaTime;

    if (moveTimer <= 0)      You, 4 days ago • ajout ia
    {
        randomDirection = Random.Range(0, 2) == 0 ? -1 : 1; // Randomize direction
        randomMoveDuration = Random.Range(0.5f, 2f); // Randomize duration
        moveTimer = randomMoveDuration;
    }

    movement.horizontalInput = randomDirection;
}
```

FIGURE 33 - Fonction déplacement aléatoire

5. Graphismes

Concernant les graphismes des personnages, nous nous sommes inspirés du style utilisé dans *Street Fighter 2*, un style reposant sur le *pixel art* en 2D.

Chaque personnage est alors l'incarnation d'une identité qui évoque des traditions culturelles. Bien sûr, ces caractères représentent un style de jeu bien à eux.

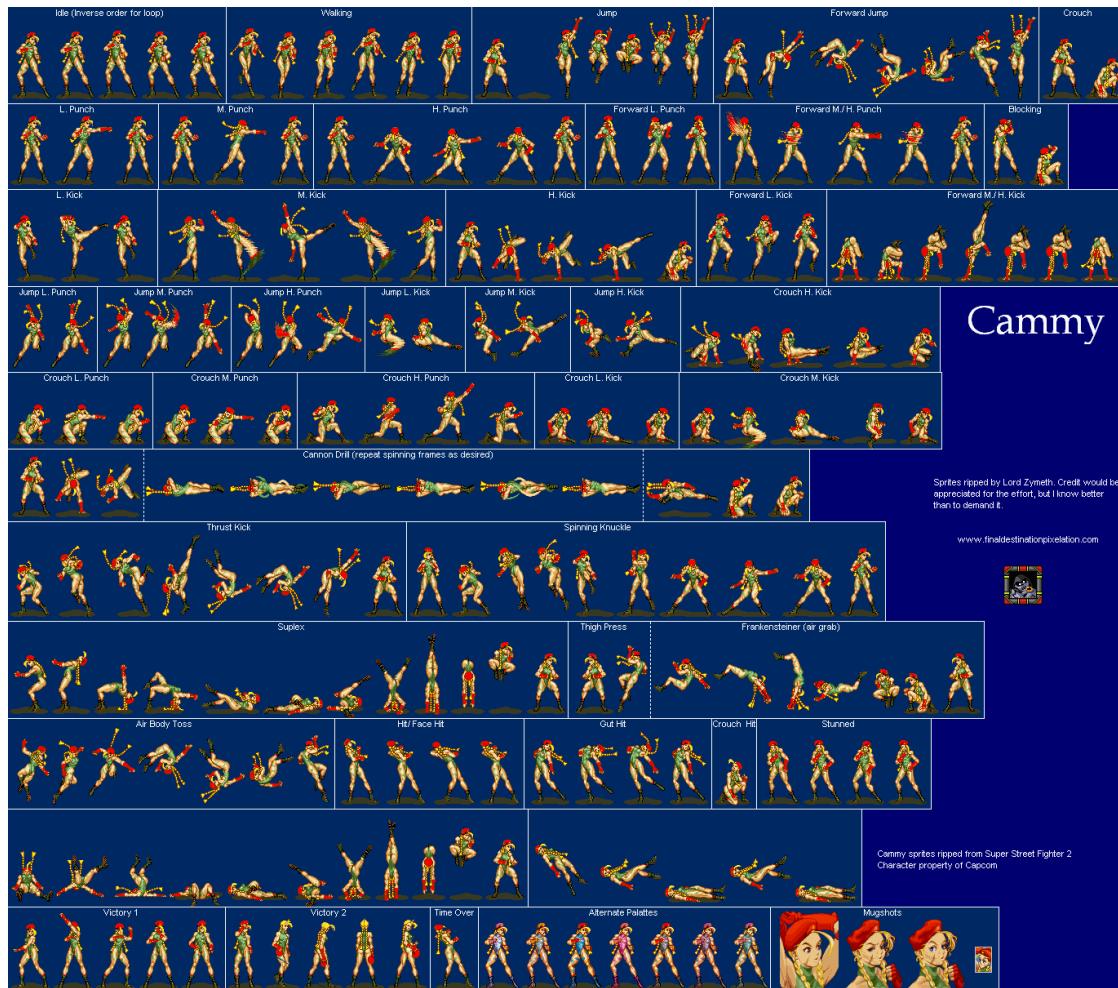


FIGURE 34 - Inspiration graphique pour l'animation et le style des personnages

Personnages

Tout joueur doit pouvoir s'exprimer en jouant à ce jeu, c'est pour cela que nos personnages vont d'un monstre à l'élégance.

Notre jeu dispose actuellement de trois combattants jouables : *Carmen*, *Louis* et *Bob Un*. Chacun d'entre eux incarne un pays à travers des éléments de la culture populaire et dispose d'une histoire qui lui est unique.

Notre premier personnage est *Carmen* qui incarne l'Espagne. Nous avons souhaité qu'elle représente ce pays à travers la robe rouge de la danse typique de Séville : la *sevillana*. Son attaque spéciale reflète aussi la zone qui la caractérise puisque c'est un coup de jambon. Ainsi, pour garder l'esprit de la danse, le style de combat de *Carmen* est plutôt un style rapproché et explosif.



FIGURE 35 - Apparence de *Carmen*[18pt]



FIGURE 36 - Présentation de *Carmen*

Louis incarne la France à travers le côté royaliste qui fait référence à l'époque forte du pays pendant le XVI^e et XVII^e siècle. Son style de combat se rapprochant de l'escrime est fait pour tenir à distance son adversaire et est plutôt lent.



FIGURE 37 - Apparence de *Louis*



FIGURE 38 - Présentation de *Louis*

Bob Un fait aussi partie des personnages jouables, représentant

un plat du Vietnam et plus largement d'Asie. Nous avons décidé d'ajouter une touche originale à ce personnage, lui ajoutant un charme unique. Allié à la silhouette d'un lutteur japonais, s'ajoute aussi une fusion avec un plat vietnamien, le *bo bun*. Le style de combat de ce personnage se base sur les sumos, il est très lent et dispose d'une grande résistance.



FIGURE 39 - Apparence de *Bob Un*



FIGURE 40 - Présentation de *Bob Un*

Menus

Pour les textes affichés dans les menus, nous n'avions à notre disposition aucune police d'écriture préexistante qui correspondait à ce que nous recherchions tout en étant libre de droit. Ainsi, nous avons décidé de produire notre propre police. Celle-ci a été entièrement faite à la main, et à l'aide du site internet *Ico-moon*. Ce site permet de créer un fichier qui respecte le format d'une police d'écriture à partir de plusieurs images vectorisées, qui correspondront chacune au glyphe souhaité. Ainsi, après avoir créées les images avec *Photoshop*, puis les avoir vectorisées, nous avons pu créer notre propre police d'écriture.

Elle nous a permis de proposer des boutons et des textes en accord avec la direction artistique du jeu et de l'entreprise dans nos menus. C'est à travers son style ardent et dynamique que la police réussit à s'intégrer dans le style du jeu que nous voulions créer.



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9

FIGURE 41 - Police d'écriture du jeu



FIGURE 42 - Exemple de boutons utilisant la police d'écriture

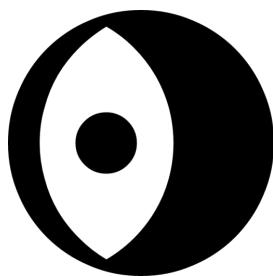


FIGURE 43 - Logo de Icomoon



FIGURE 44 - Logo de Photoshop

6. Interface

Concernant l'interface utilisateur, nous avons créé une interface ergonomique qui se traduit par les boutons essentiels qui permettent de naviguer entre différents endroits. Parmis ces boutons il y a les boutons orientés vers les informations tels que les options, les informations additionnelles sur les personnages. De plus, il y a le bouton *START* qui permet de débuter une partie, puis le joueur a le choix de lancer le tutoriel, jouer seul ou avec une autre personne en partie locale. A chaque section, il est possible de retourner en arrière. Pour finir, sur le menu principal se trouve le bouton *EXIT* pour sortir du jeu.

Pour les boutons les plus importants, nous avons accordé de l'importance au fait de les animer lorsque la souris de l'utilisateur les survole, toujours dans cette optique de rendre les menus plus dynamiques et agréables.

Quant aux fonds d'écran, nous avons cherché à conserver l'ambiance "rétro" du jeu, tout en laissant parler notre créativité par moments. Par exemple, lors de la sélection des personnages, une carte du monde est présente en fond, et lorsque l'utilisateur survole un combattant, une croix correspondant à son pays d'origine apparaît.



FIGURE 45 - Menu principal



FIGURE 46 - Bouton Start animé lors du survol par la souris

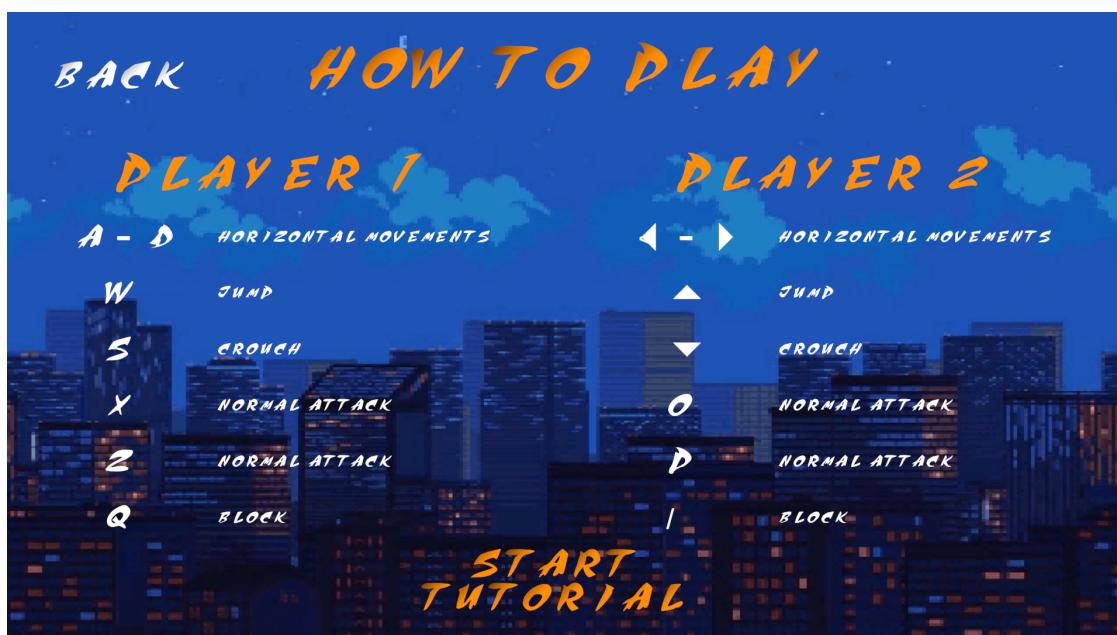


FIGURE 47 - Page de tutoriel



FIGURE 48 - Selection des personnages pour le combat

6. Audio

Musiques

L'identité audio est selon nous un élément majeur dans un jeu de combat. Elle permet de faire entendre toute la pression que le joueur doit ressentir. Bien qu'une musique soit idéale pour notre jeu, elle peut ne pas être libre de droit. Après avoir tenté de contacter un artiste, en vain, nous n'avons pas baissé les bras et avons contacté un autre artiste qui nous a cédé les droits de sa musique.

En écoutant les musiques, l'identité est claire : c'est un combat de rue. Sans règle, sans limitation, les joueurs s'expriment par le langage le plus ancien : le langage des poings. Ainsi, les poings s'expriment sur un fond instrumental de culture populaire.

Chaque scène comporte une ambiance différente, du menu principal, à la montée en pression de la sélection des personnages, puis à la cartharsis lié au combat pour ensuite finir sur le dénouement de la fin du duel.

Nous avons alors deux sources de sons, l'un renvoie la musique et l'autre les effets spéciaux. Il est alors légitime de vouloir changer leur volume. Deux barres coulissantes sont disponibles dans les options, elles changent le volume de la musique ou des effets spéciaux.



FIGURE 49 - Scène des Options pour régler le son

Effets sonores

Les effets sonores représentent la récompense d'une action, que ce soit la réussite d'une attaque ou d'un blocage. Ils permettent aussi de rendre l'expérience de jeu plus immersive. Dans ces cas-là, il faut alors trouver les bons sons. Dépourvus de matériel de bonne qualité, il est plus difficile de créer les sons nous-même. Nous avons alors l'idée de chercher des sons libres de droit sur Internet.

Les effets spéciaux sont adaptés selon chaque personnage mais aussi chaque attaque afin de réellement donner une indication sonore aux joueurs au lieu de devoir bouger son œil sur la barre de points de vie.

Ces sons sont prélevés depuis des banques de données libres de droit. La plus grande difficulté a résidé dans la sélection des œuvres, notamment avec leur qualité ou la correspondance des sons avec ce qu'ils illustrent.

```
0 references
public void ChangeVolume(string param, float val)
{
    if (param == "music")
    {
        VolumeMusic = val;
        music.volume = val;
        instance.music.volume = VolumeMusic;
    }
    else if (param == "SFX")
    {
        VolumeSFX = val;
        SFX.volume = val;
        instance.SFX.volume = VolumeSFX;
        PlaySFX(Resources.Load<AudioClip>("Sound/missedShot"));
    }
}
```

FIGURE 50 - Fonction pour changer le volume des effets sonores

```
9 references
public void PlaySFX(AudioClip clip)
{
    instance.SFX.PlayOneShot(clip);
    SFX.PlayOneShot(clip);
}
```

FIGURE 51 - Fonction pour jouer les musiques et sons

III. Améliorations possibles

Même si nous considérons que la version actuelle de notre jeu est une version finie, nous sommes conscients que de nombreuses améliorations pourraient être apportées à ce premier projet de jeu vidéo. Dans cette section, nous allons donc lister les idées que nous aimerais ajouter à la version actuelle de notre jeu.

1. Améliorations de contenu

- rajouter des personnages uniques correspondant à de nouvelles cultures avec le temps.
 - rajouter le support de la manette pour naviguer dans les menus et pour combattre.

2. Améliorations graphiques

- Utiliser nos propres fonds d'écran exclusifs au jeu.
 - des animations plus fournies en détails et plus fluides.
 - rajouter des effets visuels sur les attaques des personnages lors des combats

3. Améliorations du son

- Ajouter des musiques pour plus de diversité sonore
 - Créer des voix pour les personnages et des lignes de dialogues entre eux.
 - Donner au joueur la possibilité de choisir la musique qu'il souhaite entendre au cours du combat à venir.

4. Améliorations de l'intelligence artificielle

faire en sorte que l'intelligence artificielle s'arrête lorsqu'elle touche le joueur pour ne pas le pousser.

- Donner à l'intelligence artificielle la possibilité de bloquer et s'accroupir.
- Rendre l'intelligence artificielle plus imprévisible pour donner

plus de profondeur de jeu.

- Créer différents niveaux de difficulté disponibles lors de l'affrontement de l'intelligence artificielle.

5. Résolution de problèmes

Nous n'avons pas pu effectuer assez d'essais et dans toutes les dispositions possibles pour garantir un jeu sans aucun problèmes ; nous avons donc conscience que des soucis pourraient sans doute survenir lors de l'utilisation du jeu par les utilisateurs

IV. Récit de la réalisation

1. Impressions globales

Tout au long du développement, l'enjeu majeur a été de se familiariser avec le moteur de jeu *Unity*. En effet, *Unity* est un outil très utile quant au développement de jeux vidéos, néanmoins il regorge de ressources et d'outils qui permettent un développement rapide et efficace mais qui sont assez compliqués dans la prise en main.

Tout au long de ce projet, nous avons appris le fonctionnement du travail en groupe lors d'un travail informatique qui n'est pas toujours simple. En effet, nous avons dû faire face à plusieurs épreuves ralentissant l'avancement de notre projet comme par exemple le départ tardif de notre camarade Qaiser ALKUBATI. Cet évènement nous a retardé puisque nous avons dû répartir à nouveau les tâches dont il devait s'occuper, ce qui a rajouté du travail supplémentaire aux autres membres du groupe.

Ce projet en groupe nous a aussi permis de développer une forte communication de groupe malgré les différentes opinions de chacun.

2. Impressions personnelles

Ethan CARROUÉE

L'ensemble du projet était très enrichissant. Travailler en équipe nous a permis de répartir efficacement les tâches et de tirer parti des forces individuelles de chaque membre.

J'ai trouvé particulièrement intéressant de m'immerger dans la création des animations des personnages à l'aide de sprites et la mise en place des conditions pour gérer les transitions entre les différentes animations. Cette partie du projet m'a permis d'approfondir ma compréhension des mécanismes dans Unity.

Au départ, j'avais des appréhensions concernant la complexité du développement de notre jeu. Cependant, au fur et à mesure que je me suis impliqué dans le processus, j'ai constaté que cela devenait plus simple que prévu.

L'intégration du multijoueur a ajouté une dimension additionnelle de défi et d'intérêt. La coordination entre les joueurs, la gestion des sélections et la synchronisation des actions ont nécessité une attention particulière pour assurer une expérience de jeu fluide et équitable. Travailler sur cette fonctionnalité a non seulement renforcé notre compréhension des concepts de jeu en mode multijoueur, mais a également souligné l'importance d'une interface utilisateur intuitive et réactive. Je me suis rendu compte que la clé résidait dans la persévérance. Plus je m'immergeais dans le processus, plus je me sentais à l'aise avec le code et les outils de développement. Ce qui semblait insurmontable au début est devenu progressivement plus familier et accessible.

Yann FERNANDEZ PUIG

Ayant déjà eu l'occasion de participer à des projets de ce style, j'avais des facilités au début du projet. Néanmoins, le développement de ce jeu vidéo m'a permis d'acquérir de nouvelles compétences et connaissance dans le domaine. Notamment la mise en place de l'intelligence artificielle a été un grand défi pour nous puisque aucun membre du groupe n'y avait jamais été confronté et ceci nous a permis de développer de nouvelles compétences.

Un projet d'une telle ampleur m'as donné l'oppourtunité de développer mes capacités de communication en groupe, qui est un facteur très important dans le travail en équipe. Par ailleurs, j'ai grandement apprécié de travail dans ce groupe vu que l'ambiance dans l'équipe était bonne.

Pendant le développement, nous avons fait face à de multiples difficultés mais en travaillant en groupe, nous avons pu y faire face sans de grandes difficultés en nous entraînant.

Matteo FLEBUS

Ce projet m'a fait découvrir *Unity* et m'a appris une partie de son fonctionnement, en plus de m'avoir poussé à maîtriser le partage en ligne d'un projet via *GitHub*. De plus, concernant le travail en groupe, j'ai dû apprendre à me coordonner avec les autres membres du groupe en me focalisant seulement sur une partie du travail qui m'était attribuée, et à demander de l'aide à ceux plus expérimentés lorsque cela était nécessaire.

Mon travail sur les graphismes a développé ma créativité, et m'a poussé à rechercher une identité graphique pour notre jeu qui lui serait propre. De plus, j'ai été assisté dans ce travail par une étudiante en école d'arts, qui m'a montré comment utiliser certains outils graphiques comme *Photoshop*, et surtout qui a su me prodiguer des conseils très précieux à tous les niveaux de la création.

Durant cette année, j'ai parfois pu rencontrer des difficultés ou des pertes de motivation, mais c'est en demandant de l'aide aux autres membres du groupe et en redoublant de détermination que j'ai toujours réussi à rebondir pour aller de l'avant.

De façon générale, je suis très satisfait de notre projet, et je sens que c'est réellement pour moi une façon d'apprendre qui m'a beaucoup apporté, et sur des domaines variés. Je considère ce projet comme une réussite tant sur le plan personnel que collectif.

Bao-Anh TRAN

Pour moi, *Kick Out!* représente plus qu'un jeu mais surtout la croissance d'un groupe de jeunes passionnés. C'est alors une évidence que je ne peux être que motivé. Au début de ce projet, j'appréhendais l'utilisation de l'outil *Unity*. Bien que je sois habitué avec *Python*, j'avais l'impression de recommencer à zéro. En continuant d'apprendre, je me rends compte que c'est atteignable et je commence à comprendre pourquoi *Unity* est largement utilisé. Ce nouvel outil me permet aussi de découvrir de nouvelles façons de coder. Il est alors évident que ce projet me permet d'ouvrir de nouvelles portes en exploitant de nouveaux éléments. De plus, j'ai eu l'opportunité d'apprendre de mes camarades mais aussi de leur apprendre de nouvelles procédures. Je me sens alors épanoui dans ce projet.

Bien sûr, nous avons rencontré des obstacles dans le projet. Il y a par exemple le départ inattendu de l'un de nos membres, la coordination des travaux.

Pour finir, une ambiance amicale et enflammée règne dans le groupe. Cette flamme pousse chacun à pouvoir offrir le meilleur de lui-même afin de produire le meilleur de nous-mêmes. C'est ce qui nous a animé et c'est grâce à cela que nous avons pu aller jusqu'au bout de ce projet. Je suis fier du résultat final et aussi fier d'avoir pu faire cela aux côtés de mes camarades.

V. Conclusion

Tout au long de ce rapport, nous avons exposé de manière globale tout ce qui a été entrepris depuis le lancement du projet. Nous sommes passés par tous les différents aspects liés au développement. Ce rapport expose les réponses aux attentes établies depuis le début de ce travail.

Ce projet a été le fruit d'un apprentissage intellectuel et humain. Il est aussi le témoin du chemin parcouru de jeunes passionnés. Cela n'a pas été sans difficulté mais le projet n'est pas resté sans solution. Nous sommes alors fiers de pouvoir vous présenter notre jeu *KickOut!*, le sixième étranger de notre équipe.