Rapport de soutenance- Projet S2

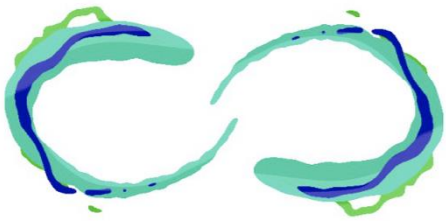
Epita 2022

Sacha Reggiani

Clément Fossati

Aboubakar Charf

Eytan Benhamou



CEAS

Sommaires

1.Les bases :

Dans un premier temps nous avons dû établir différentes choses assez basiques. À savoir sur quelle plateforme nous allions-nous tourner pour la création du projet, le choix s'est fait sur GitHub car c'est la plateforme la plus connue et la plus utilisé pour la création de divers projets. Une fois le repo fait il fallait encore trouver la version d'Unity nous allions utiliser pour ce projet il a en soit assez peu d'importance vu que toute les versions on à peu près les mêmes fonctionnalités, on a donc choisi une version qui date d'un mois ni trop vielle ni trop récente.

Une fois ces modalités finies il fallait encore apprendre à se servir de ces outils, bien que nous ayons déjà une certaine expérience avec eux notamment grâce aux cours de NTS qui nous ont initiés aux bases de Unity et les différents TP des ACDC qui nous ont permis d'apprendre les commandes de bases pour git.

Pour ce faire on a fait tous les Levels de l'excellent site learngitBranching qui permet d'apprendre toutes les bases ainsi que des commandes peu communes bien que très pratique.

Pour ce qui est de la partie Unity le meilleur moyen d'apprentissage trouvé est surement la chaine de Brackeys qui fait de très bons tutoriels sur la quasi-totalité des sujets d'Unity, ses guides ont notamment servis de bonne base pour beaucoup d'aspect de notre jeu comme le menu, les déplacements, les grille de carte.

2.Gameplay et multijoueur :

2.1. Le Commencement du projet :

Pour le début du projet après avoir créé le repo sur GitHub puis l'avoir cloné dans un dossier sur le bureau via l'installation de git pour manipuler l'environnement Windows comme s'il s'agissait d'un terminal linux. Le travail pouvait commencer, tout d'abord il y avait une pléthore de guide sur internet sur comment réaliser un bomber man, le seul qui ait réellement servi est celui de Brackeys, après avoir vu le tutoriel en entier, j'ai vite compris que le projet allait prendre beaucoup plus de temps que prévu. Il fallait tout d'abord comprendre un nouveau système de création de niveau, à savoir les grilles (ou tilemaps).

2.2 Les Tilemaps :

C'est sur cette partie que le projet a véritablement démarré, après être allait voir la vidéo de Brackeys sur les Tilemaps, tout était plus claire mais pour ce faire il fallait des sprites basique de murs, de sol, d'obstacle etc.…

Étant au début du projet et voulant avancer rapidement sans être encombré par la partie graphique, il a fallu importer des sprites venant du Unity asset store, qui était d'ailleurs assez mal adapté à la vue du dessus. L'utilisation d'une tilemap était assez facile à prendre en main malgré le fait que les sprites était initialement conçu pour une vue du dessus incliné vers l'avant.

Une fois la toute première carte finie avec les différentes couches pour le fond et le gameplay, il fallait une vraie carte pour premièrement jouer dessus et faire différents tests et aussi pour la soutenance avoir une carte suffisamment belle pour ne paraitre être une prémices de ce qu'on aurait pu faire pour la première soutenance.

2.3 Les Bombes :

Pour ce qui est des bombes au départ je voulais pouvoir les poser à la souris pour voir déjà si elles marcher et pouvoir faire d'autres test que nous ne pourrions réaliser avec un personnage qui les pose, comme regarder ce qui se passe si on pose une bombe en dehors des murs. Pour créer les bombes à la base j'ai encore une fois importé un modèle déjà existant pour pouvoir tester le code pour voir si tout fonctionne bien. Il y a plusieurs scripts associés aux bombes, tout d'abord il y en a un qui regarde le préfabriquée bombes et qui le détruit au bout de 2 secondes d'existence.

```
public float countdown = 2f;

void Update () {
    countdown -= Time.deltaTime;

    if (countdown <= 0f)
    {
        FindObjectOfType<MapDestroyer>().Explode(transform.position);
        Destroy(gameObject);
    }
}
```

Vient ensuite celui qui permet d'instancier les bombes via l'emplacement de la souris sur la grille

```
void Update () {
    if (Input.GetMouseButtonDown(0))
    {
        Vector3 worldPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        Vector3Int cell = tilemap.WorldToCell(worldPos);
        Vector3 cellCenterPos = tilemap.GetCellCenterWorld(cell);

        Instantiate(bombPrefab, cellCenterPos, Quaternion.identity);
    }
```

Les différentes méthodes que nous voyons la sont assez complexe à appréhender mais seront expliqué plus tard dans la partie script.

On peut d'ailleurs voir dans le premier script lié aux bombes on peut lire map destroyer qui est le dernier script relié aux bombes du jeu, il indique simplement à l'objet bombes de lancer l'animation associé sur plusieurs cases à la suite tout en détruisant les différents obstacles qui dressent sur son chemin (évidemment les murs ne sont pas pris en compte dans l'explosion, seuls les objets avec le tag destructible peuvent être détruit).

```
bool ExplodeCell (Vector3Int cell)
{
    Tile tile = tilemap.GetTile<Tile>(cell);

    if (tile == wallTile)
    {
        return false;
    }

    if (tile == destructibleTile)
    {
        tilemap.SetTile(cell, null);
    }

    Vector3 pos = tilemap.GetCellCenterWorld(cell);
    Instantiate(explosionPrefab, pos, Quaternion.identity);

    return true;
}
```

Cette méthode comme dit juste au-dessus détruit les blocs, il y a donc une autre fonction qui permet d'appeler cette méthode sur toutes les cases souhaitées, actuellement la façon de le faire est assez peu ergonomique, cela ne fait que répéter le script sur toutes les cases en forme de croix toute autour. Le grand problème de cette méthode est donc que ça ne laisse pas la possibilité d'ajuster la taille de l'explosion, ce qui sera problématique pour plus tard lorsque ne nous ajouterons les power-up.

2.4 Le Personnage :

C'est la première partie ou il y a des repères, c'est quelque chose d'assez facile à prendre en main et intuitif, on a eu les cours NTS ou justement le personnage prenez une grande partie du TP à faire, donc pour ce qui en est actuellement, le visuel du personnage est loin d'être achevé (c'est une capsule bleue) maintenant mais au vu du temps restant on a préféré se focaliser sur tout le reste à savoir le gameplay, le multijoueur et le menu. Donc pour ce qui est du personnage en lui-même on a quelque chose de très simple mais de fonctionnel, en termes de déplacement le personnage se déplace via un rigidBody sur lequel on applique une force. Pour tout ce qui est collisions avec les murs c'est directement gérer pas la tilemap gameplay où il y a un rigidBody homogène à tous les murs.

```
void FixedUpdate()
{
    rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime);
}
```

Dans ce screen on voit aussi la méthode est défini dans FixedUpdate, cela permet ne pas dépendre du nombre de FPS de l'utilisateur (ce qui est le cas de la fonction Update), donc FixedUpdate sera constant dans son appelle ce qui le mieux pour gérer le déplacement d'un personnage. Juste au-dessus rb est le rigidBody du joueur et movement est défini en fonction de si le joueur appui sur ses touches de déplacements

```
movement.x = Input.GetAxisRaw("Horizontal");
movement.y = Input.GetAxisRaw("Vertical");
```

3. Le Multijoueur :

3.1 Loading :

Avec l'ajout du multijoueur dans le projet, certaines fonctionnalités ont disparu comme le fait de pouvoir poser des bombes directement sur le personnage mais pas de panique toutes ces erreurs ont vite été réparées, mais penchons-nous plutôt sur comment le multijoueur a été implémenté.

La majeure aide pour la confection du multijoueur fut le guide de BlackThornProd qui montre comment ajouté cette fonctionnalité en 9 étapes assez simple. Pour le multijoueur on s'est tourné vers les serveurs de Photon, qui sont une très bonne méthode car gratuit, fiable et marche bien avec Unity. Donc après avoir créé le serveur Photon et importé l'asset PUN2 du Unity asset store, tout été prêt.

Maintenant on a créé une nouvelle scène qui permet de se connecter au serveur et de lancer une nouvelle scène qui sera là où on pourra créer un lobby ou en rejoindre un.

```
public class ConnectToServer : MonoBehaviourPunCallbacks
{
    void Start()
    {
        PhotonNetwork.ConnectUsingSettings();
    }
```

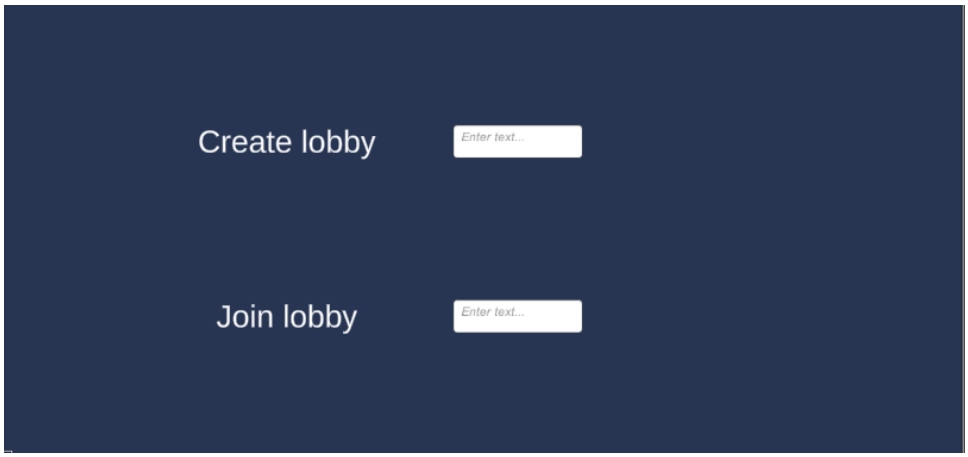
Dans le code juste au-dessus on voit le script associé à la scène de chargement, à la place de l'habituelle MonoBehaviour cette fois on a MonoBehaviourPunCallbacks ce qui va rappeler le serveur jusqu'à être connecté sur celui-ci. La fonction start launch justement cette connexion.

```
public override void OnConnectedToMaster()
{
    PhotonNetwork.JoinLobby();
}

public override void OnJoinedLobby()
{
    SceneManager.LoadScene("Lobby");
}
```

Le code du dessus est la suite directe au code d'avant, une fois connecté il va rejoindre le lobby (le serveur dédié que j'ai créé), puis quand on est connecté on va changer de scène pour enfin pouvoir créer ou rejoindre des parties.

3.2 Lobby :



Voilà la scène du Lobby, pour l'instant c'est très épuré mais ce n'est pas encore le rendu final.

Sur cette scène on peut observer deux boutons create lobby et join lobby qui respectivement prennent les contenus des champs d'écritures à droite et appliquent l'effet associé.

```
public class CreateAndJoinRooms : MonoBehaviourPunCallbacks
{
    public InputField createInput;
    public InputField joinInput;

    public void CreateRoom()
    {
        PhotonNetwork.CreateRoom(createInput.text);
    }

    public void JoinRoom()
    {
        PhotonNetwork.JoinRoom(joinInput.text);
    }

    public override void OnJoinedRoom()
    {
        PhotonNetwork.LoadLevel("PlayTest");
    }
}
```

Voilà le script de la scène on peut voir les deux méthode CreateRoom et JoinRoom qui respectivement crée une nouvelle partie et rejoignent une partie déjà existante. Vous l'aurez probablement compris, mais le système de création de partie repose juste sur un nom pour rejoindre une partie. D'ailleurs pour jouer en solo il sera donc impératif de se connecter d'abord au serveur

4.Menu/Options :

4.1 Menu :

En ce qui concerne le menu, on s'est aidé des vidéos de Brackeys, qui nous explique comment implémenter un simple menu, que l'on étoffera pour la seconde soutenance.

Comme on peut le voir sur le screen ci-dessous, il y a pour le moment un bouton Play fonctionnel qui nous permet d'arriver sur la loading scene.

Pour ce faire on a créé un nouveau panel qui servira de background, sur lequel on a importé une

Image de fond, et contrasté les couleurs de façon à ce que l'on puisse avoir un bon fond pour le menu.


```
public void play()
{
    SceneManager.LoadScene("Loading");
}
```

Pour ce qui est du bouton play, on a créé la fonction play() qui permet donc d'accéder au lobby.

Pour le bouton exit, on a créé la fonction exit() qui permet donc de fermer le jeu.

4.2 Options :

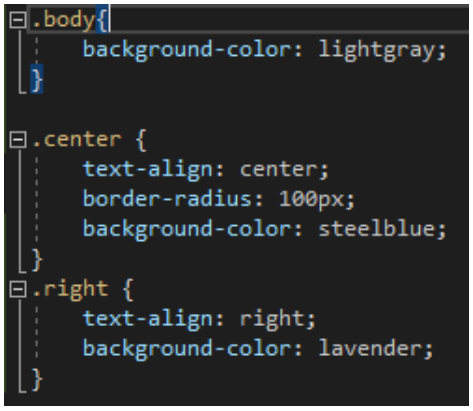
Le menu sera amélioré d'ici la prochaine soutenance, incluant quelques options fonctionnelles.

5.Le Site :

Nous avons mis en ligne un site via GitHub, le but étant de créer un site partant de zéro, avec html et css, nous avons donc un site disponible à l'adresse : https://yeckes.github.io/Ceas.website/Index.html

Ce site contient un dossier image, contenant pour l'instant seulement 2 images : le logo du site, et la liste des logiciels utilisés.

Il contient aussi un fichier en .css nommé "style.css" qui permet de créer des classes pour la mise en forme, style.css contient pour l'instant que 3 classes.



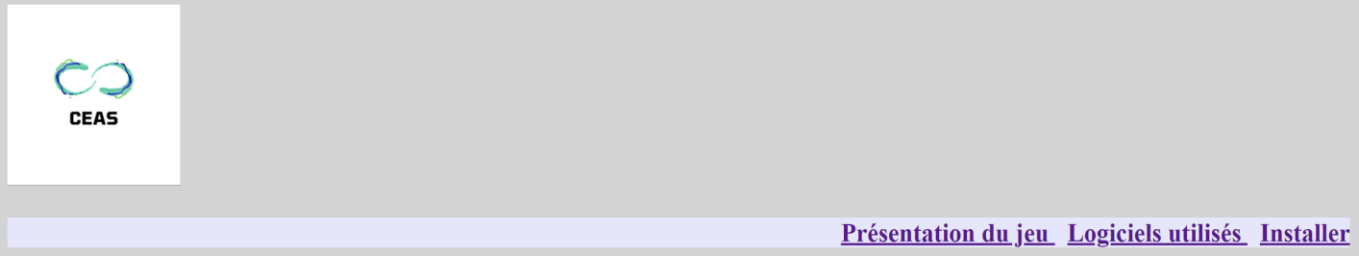
La 1er nommé "body" est la classe générale de toutes les pages, elle permet notamment de changer la couleur de l'arrière-plan des pages en gris.

La 2eme est nommé "center" permet de mettre du texte sous forme de titre, c'est à dire ile permet de centrer du texte et de lui donner une couleur de fond.

La 3ème et dernière appelé "right" sert pour créer l'en-tête car elle permet de décaler le texte à droite et de mettre une couleur de fond.

Le site contient principalement quatre pages html :

_ La 1er page qui est la principale est nommée index.html, cette page contient une en-tête avec le logo de Ceas ainsi que les noms des autres pages auxquelles sont associés les liens vers celle-ci.



En dessous de l'en-tête sur la page se trouve nos noms ainsi que celui du jeu.

Les liens présents dans l'en-tête, qui est d'ailleurs présente sur toutes les autres pages du site, mène donc aux 3 autres pages du site :

_Presentation.html est une page qui contient tout d'abord la même en-tête que les autres ainsi qu'une description du jeu et du déroulement d'une partie.

_Logiciels.html contient l'en-tête ainsi qu'une liste de tous les logiciels utilisés.

_Installateur.html possède toujours l'en-tête identique aux autres ainsi qu'un bouton qui pour l'instant ne fait rien, mais qui comme son nom l'indique "Installer Bomberman Windows" va permettre d'installer notre jeu.

Le site va bien sûr être mis à jour, il doit déjà être remplis avec toutes les informations manquantes comme les logiciels utilisé. Mais la mise en forme est aussi à revoir pour obtenir un site plus esthétique.

6.Objectifs :

Objectif 1er soutenance (semaine du 7 mars 2022) :

Pour cette première soutenance les objectifs sont tout d'abord dans un premier temps de démarrer le projet. Les fonctionnalités attendues sont :

- Avoir une première carte pour pouvoir tester le jeu, pas nécessairement avec d'excellent graphismes ni le style graphique du rendu final.

- Avoir le cœur du jeu, les déplacements, les collisions, le système de bombes que le joueur peut poser.

- Commencer le multijoueur pour avoir au moins une base dès le début du projet pour ne pas avoir à gérer deux versions jeu, une en solo, et une en multijoueur car il pourrait y avoir du retard sur la version en solo.

- Avoir un menu opérationnel qui permet de lancer le jeu et le quitter.

-Avoir un site en ligne.

Ceci sont nos objectifs pour la 1ère soutenance. Comme on peut le voir les objectifs sont pour la plupart atteint ce qui est une bonne chose en soit

Objectif 2ème soutenance (semaine du 25 avril 2022) :

Pour cette seconde soutenance l'objectif principale est d'avoir quelque chose de jouable sur lequel on peut lancer une partie, ce ne sera évidemment pas le rendu final, il manquera surement de nombreuses fonctionnalités. Les fonctionnalités attendues sont :

- Avoir une ou deux cartes avec le style graphique souhaité qui sont terminées.

- Le cœur du jeu est fini, avec toutes les différentes idées proposer au début du projet, quelque power-up seront ajoutés dans les parties.

- Le multijoueur doit être stable et fonctionnel.

- Le menu/option doit être avancé notamment sur la partie configuration des touches, gestion du volume et modification de la résolution.

- Une musique devrez aussi être présente dans le menu ou les parties.

- L'intelligence artificielle devrez être commencé bien que très limité dans ses choix et actions.

-Avoir un site présentable avec plusieurs pages et une mise en page correcte

Objectif 3ème soutenance (semaine du 6 juin 2022) :

Si tout c'est bien passer à cette soutenance, le jeu devrait être fini. Il doit comporter un menu/option fonctionnel avec un bouton jouer qui vous connecte directement au serveur de Photon pour jouer en multijoueur comme en solo (pour jouer en solo il faudra quand même être connecté à internet). Le multijoueur sera fini, on pourra se connecter pour jouer à 20 joueurs maximum (limite imposée pas photon) et choisir d'inclure ou non une ou plusieurs intelligences artificielles. Le gameplay sera fini avec tous les power-up et l'IA doit avoir une difficulté faible. La musique et les bruitages doivent aussi être fini.

7. Conclusion :

En conclusion, au travers des divers points abordés ci-dessus, nous espérons être en bonne voie pour créer un bon jeu, qui soit amusant, sans bug, bien fini et surtout rendu dans les temps. Cependant, nous ne prétendons pas avoir inventé un style de jeu nouveau ou révolutionnaire, mais nous comptons bien faire de ce jeu la meilleure expérience possible, qui je l'espère saura vous plaire.