TP 3 : Box Ninja…. au max!! (/100)

Finalité

- 1 périodes de laboratoire sont prévues pour ce travail.
- Ce travail compte pour 10% de la session.
- Vous familiariser avec le développement de fonctionnalité à partir de requis et d'instructions

Prérequis

- Avoir réalisé avec succès le formatif 5.
- Avoir consulté le document S08- Notes de cours

À remettre

- La remise devra être faite en classe à la semaine 6 en présentant le résultat à l'enseignant

Notes importantes

- Se baser sur le package « **TP3 – True Box Ninja** » fourni avec le TP2
- Se placer en équipe de 1 ou 2 pour le TP

Description du jeu final attendu

- Le jeu est moderne, vous pouvez y intercepter des objets qui saute depuis le bas de votre écran
- Un menu principal vous donne accès à commencer une nouvelle partie, ou continuer une partie qui a précédemment été sauvegardée
- Un écran de paramètre, accessible depuis le menu, qui permet de personnaliser l'expérience de chaque joueur!
- En jeu, un écran de pause qui permet de sauvegarder la progression ou de retourner au menu.

Remarques Générales

- Utilisez une scène mère pour gérer vos scripts globaux/généraux
- Partez du prototype fournis, et ajoutez les fonctionnalités attendues
- Le code doit être commenté de façon adéquate

Nouveautés (capacité à acquérir par soi-même)

- Controller le temps avec **Time.timescale**
- Quitter le jeu (Ferme l'application)
- Interagir avec des handles (**Slider**, **Toggle**, etc.)
- Bloquer l'interaction avec une scène lors d'un écran de pause avec **EventSystem.current.IsPointerOverGameObject()**
- Faire un build du projet

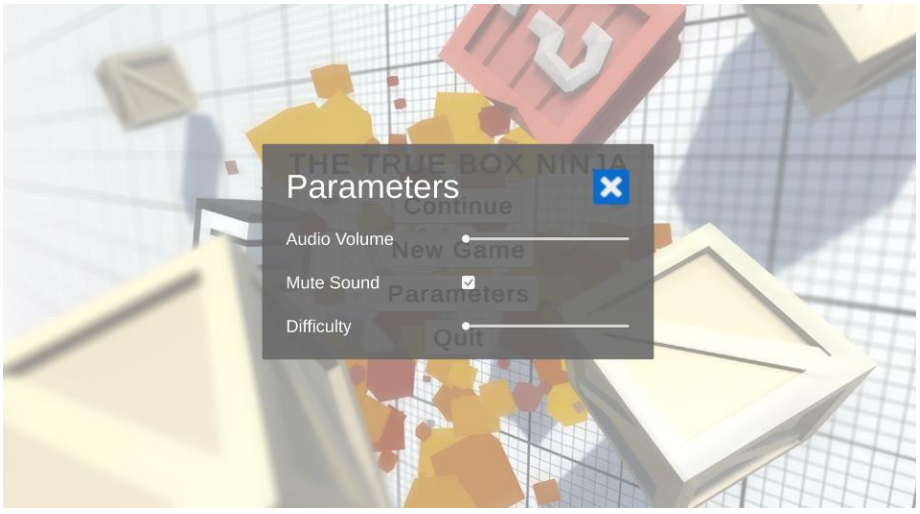
Contenu et fonctionnalités des différentes scènes :

Scène mère ‘’_source’’ :

- Contient un seul objet ‘’Master’’, qui contient vos scripts globaux
- L’objet ne doit pas être détruit lors du chargement d’autres scènes
- Ouvre automatiquement le Menu

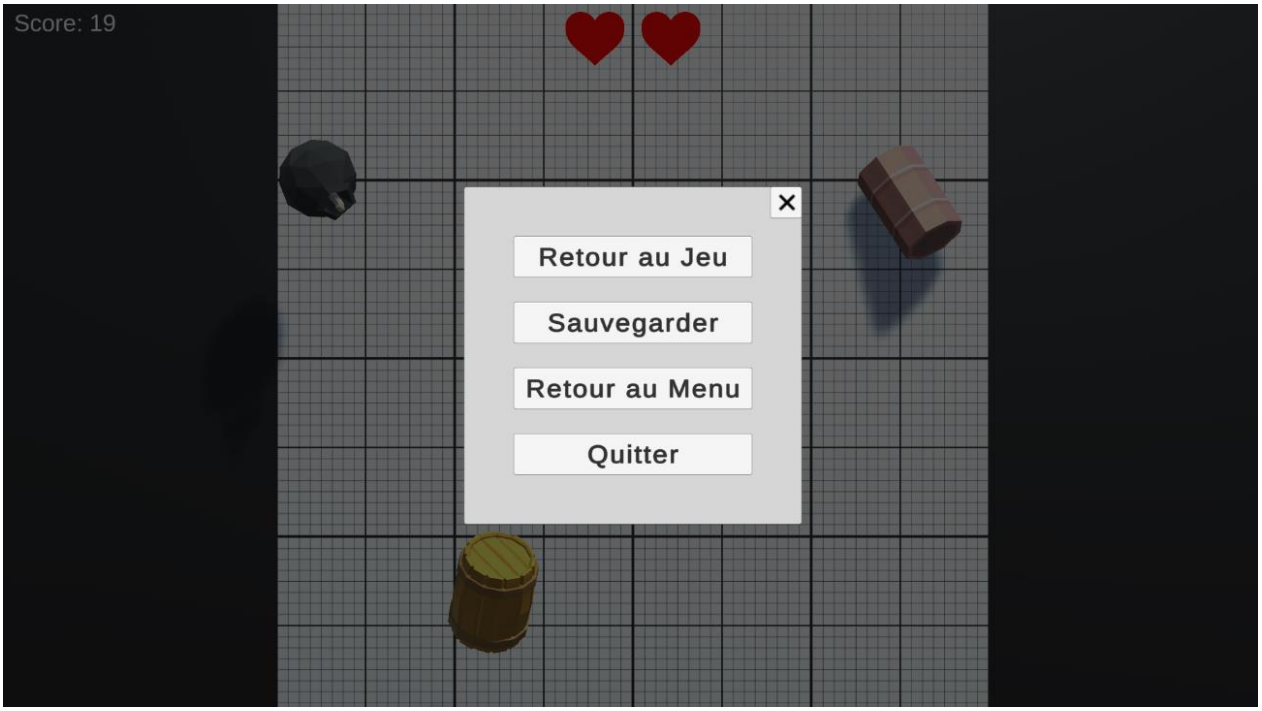
Scène d’accueil ‘’Menu’’ :

- Une image de votre jeu en background
- Le titre de votre jeu
- Un bouton ‘’Continuer’’ pour poursuivre une partie précédente (si existe)
- Un Bouton ‘’Nouvelle Partie’’ Pour commencer une nouvelle partie de zéro
- Un bouton ‘’Paramètre’’, qui ouvre un écran avec quelques paramètres de personnalisation (paramètres détaillés plus loin)
- Un bouton ‘’Quitter’’ pour fermer l’application



Scène de jeu ‘’Game’’ :

- Contient le gameplay existant
- Appuyer sur la touche « Escape » pause le jeu, bloque les interactions, et montres plusieurs boutons :
    - Bouton ‘’Sauvegarder’’ : Sauvegarde la partie présente
    - Bouton ‘’Retour au Menu’’ : remmène le joueur au menu
    - Bouton ‘’Retour au Jeu’’ : ferme l’écran de pause et continue le gameplay
    - Bouton ‘’Quitter’’ pour fermer l’application



Lorsqu’on appuie sur **Escape**, en jeu, l’écran de pause s’ouvre et le temps s’arrête.

Scripts Nécessaires et fonctions (/100):

Scène mère : (/40)

DontDestroyOnLoad.cs (/3)

- Empêche l'objet sur lequel il est attaché d'être détruit

SceneNavigator.cs (/7)

- Fonction Statique pour ouvrir la scène du Menu
- Fonction Statique pour ouvrir la scène du jeu
- Fonction Statique pour fermer l'application

SaveSystem.cs (/15)

- Contient une Classe GameState{} qui contient tous les paramètres d'une partie (score, nombre de vies, difficulté)
- Fonction Statique pour sauvegarder un GameState dans un fichier texte sur le disque
- Fonction CheckHasSave() qui retourne une bool true/false selon si une sauvegarde existe
- Fonction Statique qui retourne le GameState d'une sauvegarde existante

GameSettings.cs (/15)

- Membre statiques qui servent de wrapper à PlayerPrefs
- Définis différents paramètres utilisateurs :
    - Volume de la musique (float)
    - Afficher des particules lorsque les object en jeu son cliqué (bool)
    - 1 autre au choix

Scène « Menu » : (/25)

MenuManager.cs (/15)

- Contient les fonctions appelées par les différents boutons du menu
- Cache le bouton « Continuer » si aucune sauvegarde n'existe
- Référence la fenêtre des paramètres utilisateur.
- Et contrôle sont ouverture/fermeture
- S'assure que la fenêtre des paramètres utilisateurs est fermé lorsqu'on entre dans le menu

GameSettingsPanel.cs (/10)

- Contient une référence vers un handle pour chaque membre de GameSettings.cs
  (Slider pour paramètres de type float ou int, Toggle pour paramètres de type bool)
- Initialise les handles à la correcte valeur pour chaque paramètre
- Une fonction pour update chaque paramètre
- Les handles doivent appeler la fonction d'update de leur paramètre lorsqu'ils sont changés
- Un bouton pour fermer le Panel dans le coin supérieur droit.

Scène « Game » : (/25)

PausePanel.cs (/10)

- Une Fonction pour ouvrir l'écran de pause en jeu
- Contient les fonctions appelée par chaque bouton de l'écran

GameManager.cs (/10) (à modifier)

- Ouvre ou Ferme l'écran de pause lorsque la touche « Escape » est appuyée
- Lors de l'ouverture de la scène (dans Start()), vérifie si il s'agit d'une nouvelle partie ou d'une partie chargée, en initialize la scène correctement
- Initialise le volume de la AudioSource à la bonne valeur selon les préférences du joueur

Target.cs (/5) (à modifier)

- Utilise EventSystem.current.IsPointerOverGameObject() pour s'assurer que l'on puisse cliquer
- Vérifie les préférences du joueur avant de faire apparaitre des particles lorsque clicqué

Faire un build de votre jeu (/10)

Défi (bonus : /10)

- Lors de la sauvegarde, sauvegardez aussi la position et le type d'objets qui étaient présents
- Puis, récréez les lors du chargement.