TP3 : True Box Ninja (/80)

Une image contenant dessin humoristique, capture d’écran

Description générée automatiquement

**Finalité**

* 4 périodes de laboratoire sont prévues pour ce travail (environ deux semaines).
* Ce travail compte pour 20% de la session.
* Vous familiariser avec le développement de fonctionnalité à partir de requis et d’instructions

**Prérequis**

* Avoir réalisé avec succès le laboratoire 5.
* Avoir consulté et compris les notes de cours de la semaine 8 (*S8-Scènes, sérialisation, et PlayerPrefs.pdf*), disponible sur Moodle.

**À remettre**

* La remise, incluant une démo en classe du résultat à l’enseignant, devra être faite via GitHub avant le 5 novembre 2025, 23h59
* Vous devrez me donner un accès à votre repo GIT du TP3 à [mathieu.piette@cegeplimoilou.ca](mailto:mathieu.piette@cegeplimoilou.ca)

(Voir annexe 2 pour les instructions complète sur la création et le partage d’un repo GIT à votre coéquipière ou coéquipier, et l’enseignant du cours.)

**Notes importantes**

* Se baser sur le package « **TP3 – True Box Ninja** » fourni avec le TP3
* Le TP est **à réaliser seul** ou **en équipe de 2**

(ATTENTION : pour celles et ceux qui réaliseront le travail en équipe, il est obligatoire de compléter et remettre la grille de répartition des tâches, présentée en annexe 1).

# Description du jeu final attendu

* C’est un jeu moderne, dans lequel vous devez intercepter des objets qui sautent 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 paramètres, 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 généraux;
* Partez du prototype fournis, et ajoutez les fonctionnalités attendues
* Le code doit être commenté de façon adéquate.

# Éléments à intégrer à votre projet

* Controller le temps avec *Time.timescale*
* Possibilité de quitter le jeu (Fermer 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 (évalué lors de la démo /15)

**Scène mère « \_source » :**

* Contient un seul objet « Master », qui conserve les scripts généraux.
* L’objet ne doit pas être détruit lors du chargement d’autres scènes et ouvre automatiquement le Menu.
* Le lancement du projet ouvre automatiquement le menu principal, et le Master persiste entre les scènes sans duplication (2 pts).

**Scène d’accueil « Menu » :**

* Une image de votre jeu en arrière-plan, un titre, et quatre boutons :
  + **Continuer** (poursuit une partie sauvegardée si disponible)
  + **Nouvelle partie** (commence une partie de zéro)
  + **Paramètres** (ouvre une fenêtre de personnalisation : volume, particules, etc.)
  + **Quitter** (ferme l’application)
* Navigation fluide entre les options du menu, avec un affichage correct des boutons selon l’état de la sauvegarde, et persistance des préférences utilisateur (volume, effets, etc.). (5 pts).

Une image contenant texte, capture d’écran, dessin humoristique, conception

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

**Scène de jeu « Game » :**

* Contient le *gameplay* existant.
* En appuyant sur la touche *Escape*, le jeu se met en pause, empêche les interactions, et affiche un panneau comportant les boutons suivants :
  + **Retour au Jeu** : reprend la partie en cours.
  + **Sauvegarder** : enregistre la partie actuelle.
  + **Retour au Menu** : ramène au menu principal.
  + **Quitter** : ferme l’application.
* Fonctionnement du panneau de pause (avec *Time.timeScale*), sauvegarde et chargement effectifs d’une partie (fichier JSON), et retour fluide au menu principal. (6 pts).

Une image contenant texte, capture d’écran

Description générée automatiquement

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

***Build* final du jeu**

* Un *build* exécutable doit être livré et testé sans erreur, avec toutes les scènes fonctionnelles.
* L’exécutable lancé permet de tester le menu, la partie, la sauvegarde et le retour au menu sans *crash* ni erreur. (2 pts).

# Scripts nécessaires et fonctions (/65):

**Scène mère : (/25)**

**DontDestroyOnLoad.cs (/2)**

* Empêche l’objet sur lequel il est attaché d’être détruit

**SceneNavigator.cs (/4)**

* Fonction statique *GoToMenu()* pour ouvrir la scène du Menu
* Fonction statique *StartGame()* pour ouvrir la scène du jeu
* Fonction statique *ExitApp()* pour fermer l’application

**SaveSystem.cs (/10)**

* Contient une classe *GameState{}* qui contient tous les paramètres d’une partie   
  (score, nombre de vies, difficulté)
* Fonction *public static void SaveGame()* pour sauvegarder un *GameState* dans un fichier texte sur le disque (sérialisation avec *JSONConvert)*
* Fonction *public bool CheckHasSave()* qui retourne une *bool true/false* selon si une sauvegarde existe
* Fonction *public static GameState LoadStateFromSave()* qui retourne le *GameState* d’une sauvegarde existante en lisant le fichier texte (désérialisation avec *JSONConvert*)

**GameSettings.cs (/9)**

* Membres statiques qui servent de conteneur à *PlayerPrefs*
* Définis ces deux paramètres utilisateurs :
  + Volume de la musique (*float*)
  + Afficher des particules lorsque les objets en jeu sont cliqués (*bool*)

**Scène « Menu » : (/16)**

**MenuManager.cs (/9)**

* Contient les fonctions appelées par les différents boutons du menu
* Dans*Start() :*
  + Cache le bouton « Continuer » si aucune sauvegarde n’existe
  + S’assure que la fenêtre des paramètres utilisateurs est fermée lorsqu’on entre dans le menu
* Référence la fenêtre des paramètres utilisateur.
* Contient une fonction *OpenSettings()* qui ouvre la fenêtre des paramètres et contrôle son ouverture/fermeture.

**GameSettingsPanel.cs (/7)**

* 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 valeur correcte pour chaque paramètre
* Une fonction pour mettre à jour chaque paramètre
* Les *handles* doivent appeler la fonction d’*update* de leur paramètre lorsqu’ils sont changés
* Un bouton pour fermer le panneau dans le coin supérieur droit.

Une image contenant texte, capture d’écran

Description générée automatiquement

**Scène « Game » : (/16)**

**PausePanel.cs (/6)**

* Une Fonction *OpenPanel()*
  + Ouvre l’écran de pause, et pause le jeu avec *Time.timeScale*
* Une fonction *ClosePanel()*
  + Ferme l’écran de pause, et poursuit le jeu avec *Time.timeScale*
* Contient les fonctions appelées par chaque bouton de l’écran
  + Retour au jeu : *ClosePanel()*
  + Sauvegarder : *SaveGame()*
  + Retour au Menu : *ReturnToMenu()*
  + Quitter : *QuitGame()*

Une image contenant texte, capture d’écran

Description générée automatiquement

**GameManager.cs (/6) (à modifier)**

* Ouvre ou ferme (*toggle*) l’écran de pause lorsque la touche « *Escape* »est appuyée
* Lors de l’ouverture de la scène (dans *Start()*), vérifie s’il s’agit d’une nouvelle partie ou d’unepartie chargée, et initialise la scène correctement
* initialise le volume de la *AudioSource* à la bonne valeur selon les préférences du joueur.

**Target.cs (/4) (à modifier)**

* Utilise *EventSystem.current.IsPointerOverGameObject()* pour s’assurer que l’on ne puisse pas cliquer sur une cible à travers un menu
* Vérifie les préférences du joueur avant de faire apparaitre des particules lorsqu’une cible est cliquée.

**Faire un *build* de votre jeu (/8)**

**Annexe 1 – Répartition des tâches**

Chaque membre de l’équipe doit remplir cette grille pour détailler sa contribution au travail.

ATTENTION : Utilisez des verbes à l’infinitif (ex. : Programmer le script *PlayerController*, ajouter des sons, créer les animations, etc.).

|  |  |
| --- | --- |
| **Nom et prénom** | **Tâches réalisées** |
| [Étudiant 1] | * […] * […] * […] |
| [Étudiant 2] | * […] * […] * […] |

**Annexe 2 – Création d’un projet Unity, d’un repo GitHub, et partage du repo.**

# Création d’un nouveau projet

1. Ouvrez **Unity Hub**, puis sélectionnez ***New Project***
2. Sélectionnez la bonne version de Unity (**6000.0...**)
3. Choisissez le modèle **3D (Built-In Render Pipeline)**, puis **nommez votre projet en inscrivant tp2-[initiales étudiant 1]-[initiales étudiant 2]-[groupe]**. Par exemple : « **tp3-xy-yz-gr1** »[[1]](#footnote-1).
4. Choisissez l’emplacement local de sauvegarde de votre projet, puis cliquez sur **Create project**
5. **Attendez que le projet soit créé et s’ouvre (cela peut prendre plusieurs minutes…)**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

# Création d’un Repository GitHub et démarrage

1. Une fois la création du projet terminée, fermez Unity et Unity Hub
2. Ouvrez GitHub Desktop, et créez un nouveau repository du même nom que votre projet Unity. Par exemple : « **tp3-xy-yz-gr1** ».
3. Ajoutez une description, puis spécifiez le chemin d’accès.

**ATTENTION** : Le chemin d’accès doit être la racine où vous stockez l’ensemble de vos projets Unity, pas le dossier du projet.

1. Sous *Git ignore*, sélectionnez Unity.
2. Cliquez sur *Create repository* (les captures présentent l’exemple du TP1, à adapter pour le TP2).

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.

1. De retour dans l’interface de GitHub Desktop, cliquez sur *Publish repository*
2. Assurez-vous que *Keep this code private* est bien coché, puis cliquez sur *Publish repository.*

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.

1. **Partage de votre repo à une coéquipière ou coéquipier, ainsi que l’enseignant du cours.**
2. Rendez-vous sur l’interface Web de GitHub (<https://github.com/login>)
3. Une fois connecté, dans le menu de droite, accédez à vos *Repositories*Une image contenant texte, capture d’écran, nombre, Police

   Le contenu généré par l’IA peut être incorrect.
4. Lorsque vous êtes dans votre Repo, cliquez sur *Settings*, *Collaborators*, puis *Add people.*

Une image contenant texte, Police, nombre, logiciel

Le contenu généré par l’IA peut être incorrect.

1. Ajoutez ensuite le courriel de votre coéquipière ou coéquipier (si le travail est réalisé en équipe), ainsi que mon courriel ([mathieu.piette@cegeplimoilou.ca](mailto:mathieu.piette@cegeplimoilou.ca)).

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Les personnes invitées recevront une invitation pour collaborer sur le repo, et pourront écrire sur ce dernier.

Bon travail!

1. Comme nos projets sont hébergés sur GitHub et que le nom de projet doit pouvoir former une URL fonctionnelle, nommez vos projets selon la convention kebab-case. [↑](#footnote-ref-1)