Date de remise : **9 mai 2024, 23h59**
Pondération : 20% de la note finale (40% total avec le test sur les design patterns)
Travail : Individuel
Pénalité de retard : 15% par jour.
Remise : Déposez dans pédago un **build** de votre projet suivant la nomenclature (NOM_PRENOM) afin de valider votre remise. Votre code source sera récupéré sur Git, ajouter le compte (00532@bart.ca) à votre répertoire Git comme collaborateur.

Description du travail

Le troisième travail pratique de la session consiste à faire un petit jeu de style « *Life Simulation Game* ». Des inspirations peuvent venir de Stardew Valley, Rune Factory, ou Animal Crossing.

Situation du travail à travers le cours

Ce troisième travail pratique résume tous les éléments de la session. Vous aurez besoin des éléments présentés en début de session (Action et Scriptables Objects). Les coroutines peuvent aussi vous être utile. Vous aurez à intégrer des patrons de conception tel que nous les avons vu en classe afin de remédier à des problèmes techniques reliés à la conception logicielle.

Fonctionnalités à implémenter

- Le jeu est dans l'environnement 3D.
- Le jeu est « *single player* »
    - Le joueur peut se déplacer.
    - Le joueur ne peut pas se déplacer lorsqu'il entreprend une activité.
    - L'objectif du joueur est de compléter des activités afin de remplir son inventaire.
- 2 points (endroits) où faire des activités clairement identifiées.
    - Chaque activité propose un « *minigame* » différent
    - Il est possible de « rater » la complétion d'une activité
    - Le personnage est animé en fonction de l'activité
    - Un *minigame* est démarré à l'appui d'une touche (indiquée dans l'écran) lorsque le joueur se situe à la bonne position.
- Un « *pause menu* »
    - « *Load Game* »
        - Ce bouton permet de charger la partie (c'est-à-dire l'inventaire du joueur).
    - « *End Game* »
        - Ce bouton est grisé par défaut, il devient cliquable lorsque l'objectif du joueur est accompli.

        - Au clic, le joueur fait une animation de victoire. Après 2 secondes, le joueur est replacé à sa position initiale et l'inventaire est réinitialisé.
- Un inventaire
    - L'inventaire possède 5 emplacements exactement.
    - La complétion d'une activité donne au joueur un objet qui s'affiche dans l'inventaire.
    - Lorsque l'inventaire est plein, l'objectif du joueur est atteint (et le bouton « *End Game* » devient cliquable).
    - Une énumération existe pour déclarer les objets possibles de l'inventaire.

Consignes techniques à respecter

- Intégration d'un **système de sauvegarde**
    - Créez un système de sauvegarde (avec JSON) dans lequel vous sauvegarderez l'inventaire du joueur.
    - Sauvegarde automatique après chaque activité réussie.
    - Charger la partie repositionne le joueur à la position de départ et rafraîchi les objets dans l'inventaire.
- Intégration d'un **système d'événements**
    - Utilisé pour avertir le UI de la complétion des différentes activités
    - Utile pour avertir les différents objets du jeu que la partie est gagnée
- Intégration d'une **machine d'états**
    - Le joueur est codé avec une machine d'état
    - Vous devez avoir au minimum 4 états.
- Intégration d'un **système audio**.
    - Le AudioManager doit être Singleton.
    - Il fonctionne avec du Pooling.

Document de design.

Vous devez remettre un document de design (rédigé avec traitement de texte). Il doit inclure :

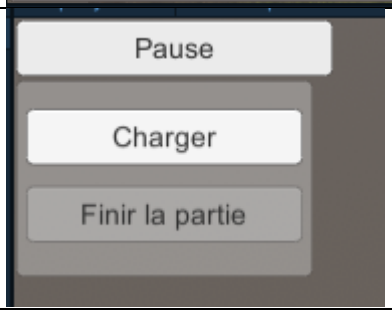
- Les contrôles de votre personnage;
- Une description des activités
    - Fonctionnalité du minigame
    - Objet récolté en cas de réussite

Remettez un document soigné, ce point est « *quasiment* » donné.

Outil de gestion de version

Créez un nouveau projet Git pour ce projet. Dans ma correction, je vais corriger la **fréquence** et la **clarté du nom des *commits***. Erreurs fréquentes : écrivez des phrases complètes, évitez les « *jokes* ».

Images de référence

Une première activité On peut afficher une touche à l'écran pour dire au joueur comment démarrer le minigame.	
Une deuxième activité On peut afficher au joueur un icone pour l'inciter à entrer une touche.	
L'inventaire Lorsque plein, on peut finir la partie	
Le menu pause Bouton « Charger » toujours disponible.	

Grille de correction

Contraintes techniques	
Sauvegarde : les bons éléments se trouvent dans la sauvegarde.	**1 pts**
Sauvegarde : une sauvegarde est lancée après chaque activité réussie.	**1 pts**
Sauvegarde : charger la partie repositionne le joueur et rafraîchi l'inventaire.	**2 pts**
Événement : Implémentation d'un système d'événements utilisés à travers l'application.	**4 pts**
État : Le joueur fonctionne avec une machine d'état.	**3 pts**
État : Au moins 4 états dans la machine.	**1 pts**
Audio : Fonctionne avec Pooling.	**2 pts**
Audio : Implémente un Singleton.	**2 pts**
Respect des éléments de l'énoncé	
Personnage : qualité des 3C (*contrôles, caméra, character*) • Se déplace • Est animé • Etc.	**2 pts**
Activités : le *gameplay* de chaque activité est différent et fonctionnel.	**2 pts**
Pause menu : les boutons « *Load* » et « *End Game* » sont fonctionnels.	**2 pts**
Inventaire : • Comporte 5 champs • La complétion d'une activité donne le bon objet dans l'inventaire • Une énumération existe pour décrire les différents objets	**4 pts**
Qualité de vie : Ajout de musiques et d'effets sonores.	**2 pts**
Rubrique « professionnalisme »	
Git : Clarté et fréquence des *commits*.	**2 pts**
Qualité du code : Le code est propre et respecte les normes de programmation.	**4 pts**
Document de design : Exactitude du document et professionnalisme.	**2 pts**
Total	**36 pts**