

# Examen médian de RV01

25 Octobre 2024

---

*Durée : 45 Minutes sur machine*

**Documents autorisés :** Tous documents autorisés, accès à internet autorisé (doc unity, site de l'UV, forums, etc.), corrigés de TD autorisés.

**Interdiction :** L'utilisation d'un projet Unity préparé à l'avance est interdite. L'utilisation de machine personnelle est interdite. **L'utilisation d'IA générative (ChatGPT et autres) est interdite.**

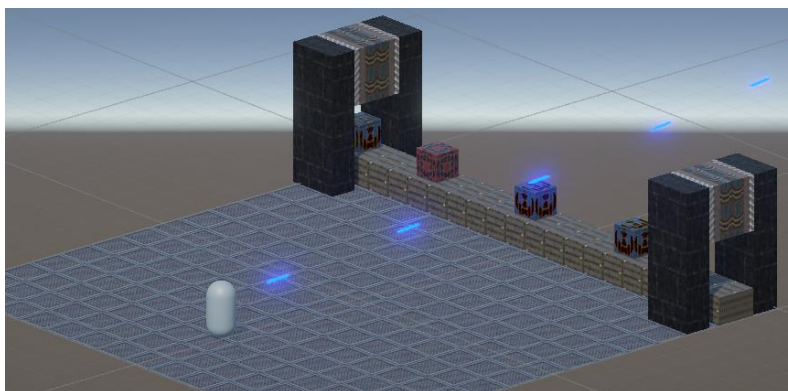
*Il n'existe pas d'unique bonne réponse à une question. L'important est d'obtenir le résultat attendu, peu importe la manière dont il est obtenu.*

*Une attention particulière sera accordée à la qualité et la clarté du code rendu (notamment des nommages), de l'arborescence de projet et des objets dans la scène, qui seront notés sur 2 points. Si vous utilisez des tags et/ou des layouts, n'oubliez pas de renseigner les champs « Tag » et « Layout » du fichier "MEDIAN - TODO at first and at last.txt" qui se trouve à la racine du projet (dossier Assets) avant d'exporter votre projet. **Le rendu doit être fait sur moodle sous forme de package Unity.** Tout manquement sera sanctionné par la perte d'une partie des 2 points de clarté.*

*Il est possible d'obtenir 24 points, mais la note reste sur 20 points.*

---

## Star Wars épisode 24



Il y a bien longtemps... Dans une galaxie très très lointaine...

Vous êtes recruté par l'alliance rebelle contre l'empire galactique. Nous vous demandons de créer une application d'entraînement pour nos soldats. La mission se

déroulera dans un centre de tri de colis, ils devront intercepter le matériel destiné à l'empire, et laisser passer le matériel destiné à l'alliance. Pour cela, ils disposent d'un blaster (pistolet laser) pour détruire les colis de l'empire.

### GamePlay :

Des colis sont générés régulièrement (1 par seconde) et aléatoirement (empire/alliance) au premier poste [A]. Les colis se déplacent ensuite sur un tapis roulant jusqu'au deuxième poste [B] pour être téléportés à leur destinataire final. Le joueur profite de ce passage des colis sur le tapis roulant pour réaliser sa mission. Cependant il ne doit pas s'en approcher pour ne pas être détecté par la sécurité, pour cela, ces mouvements seront limités à une ligne parallèle au tapis roulant (le joueur peut bouger uniquement sur l'axe X). Le joueur apparaîtra au point [C].

### Règles de scoring :

- +1 point si un colis de l'empire est détruit par un tir de blaster
- 2 points si un colis de l'alliance est détruit par un tir de blaster
- +1 point si un colis de l'alliance passe le poste [B]
- 2 points si un colis de l'empire franchi le poste [B]

Le score en temps réel est affiché en permanence sur l'écran du joueur.

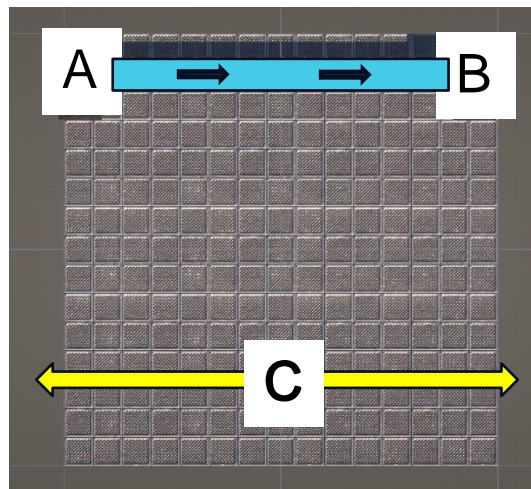


Figure 1 : Vue de dessus de la scène à réaliser.

[A] : Génération des colis.

[B] : Téléportation des colis à leur destinataire

[C] : Point de départ du joueur

[en bleu] : chemin des colis sur le tapis roulant

[en jaune] : mouvements possibles du joueur

Créez un projet Universal 3D (URP), importez les packages « AI Navigation », « Cinemachine », et « Input System » depuis le package manager. Ensuite, importez le package unity fourni avec le sujet puis ouvrez la scène "**medianA24**". Vous utiliserez les éléments déjà présents dans la scène, ainsi que les éléments déjà présents dans le projet (scripts, textures, cursor, etc.).

(En cas d'effacement involontaire du sol : « plane » avec position et rotation = (0, 0, 0) et scale = (2, 1, 2)).

## 1 Préparation des prefabs (1 pt)

1. (0.5 pt) Créez les prefabs ImperialColis et RebelColis en utilisant l'objet approprié, ajouter le script NavMeshScript.cs fourni à chacun d'eux, ainsi que le component permettant de gérer les effets de la physique. [Scale(1,1,1)]

2. (0.5 pt) Créez le prefab blasterRayon en utilisant un objet approprié (la forme d'un rayon laser pourrait être comparée à un grain de riz ou un court spaghetti). Pour des raisons d'orientation, il est possible que vous ayez à utiliser un gameobject vide parent pour que votre rayon soit par défaut dans la bonne direction. Ajoutez à la racine du prefab un component rigidbody et supprimez l'effet de la gravité. [Scale(0.1, 0.5, 0.1)]

## 2 Ajout des textures (1 pt)

3. (0.5 pt) Texturez les prefabs ImperialColis et RebellianColis avec les materials fournis dans le dossier Materials (m\_colisEmpire, m\_colisRebellion).

4. (0.5 pt) Créez un material "m\_blasterRay" pour votre prefab blasterRayon de la couleur qui vous plaira (sauf blanc !) et appliquez-la au prefab blasterRayon.

## 3 Instanciation (1 pt)

5. (0.5 pt) Créez dans la scène un GameObject "Convoyeur" en utilisant l'objet approprié et appliquez-lui le material m\_Convoyeur fourni dans le dossier Materials. [Position(0, 0.5, 8) ; Scale(20, 1, 1.5)]

6. (0.5 pt) Ajoutez une zone de navigation au convoyeur, et ajoutez aux prefabs "ImperialColis" et "RebellianColis" le component nécessaire à la navigation automatique des objets (on ne demande pas encore de faire des scripts).

## 4 Inputs (1 pt) (StarterAssets\InputSystem\StarterAssets.inputactions)

7. (0.5 pt) Modifiez le fichier de définition des inputs pour que le joueur ne puisse ni avancer ni reculer. Il pourra toujours se déplacer de gauche à droite de manière illimitée (tant pis pour lui s'il tombe de la plateforme) (on ne demande pas encore de faire des scripts).

8. (0.5 pt) Modifiez le fichier de gestion des inputs pour ajouter l'action « Shoot » dont le binding sera associé au bouton gauche de la souris.

## 5 Scripts – Apparition, déplacement et disparition des colis (4 pts)

9. (2 pts) Complétez le script ColisGenerator.cs pour implémenter la génération automatique des colis, on attachera ce script à un gameObject dédié. On souhaite instancier un nouveau colis chaque seconde, choisie aléatoirement parmi les deux prefabs disponibles. Pour rappel, les colis apparaissent au point A.

10. (1 pt) En utilisant les scripts NavMeshScript.cs et ColisGenerator.cs, implémentez le fait que chaque colis cherche à rejoindre le point B à une vitesse constante de 5.

11. (1 pt) Quand un colis atteint le point B, on souhaite qu'il disparaisse, quelle que soit son appartenance. Implémentez cette fonctionnalité.

## 6 Scripts – Shoot the package (8 pts)

12. (2 pts) Utilisez les textures croix\_blanche, croix\_rouge et croix\_verte fournies dans le dossier Cursor pour afficher un viseur au joueur et lui montrer s'il vise un colis ami (alliance = croix\_verte) ou un colis ennemi (empire = croix\_rouge), ou rien du tout (par défaut = croix\_blanche). On souhaite également que le curseur reste fixé au centre de l'écran pendant le jeu. Utilisez le script Raycasting.cs pour implémenter cette fonctionnalité et associez-le à votre player.

13. (3 pts) Le joueur doit pouvoir utiliser son blaster pour détruire des colis. À chaque appui sur le bouton gauche de la souris, une instance du prefab blasterRayon doit apparaître devant le joueur et partir en direction du point visé avec une vitesse de 50. Compléter le script Shoot.cs pour implémenter cette fonctionnalité. Vous utiliserez le gameObject vide BlasterRoot (enfant de playerCapsule/PlayerCameraRoot) comme point d'origine du tir de blaster. Vous utilisez l'orientation de la caméra pour orienter correctement votre rayon laser au moment du tir.

14. (3 pts) Lorsqu'un rayon laser touche un colis, alors le rayon laser et le colis sont détruits (quelle que soit l'appartenance du colis). De plus, si le rayon touche un autre élément, alors le rayon doit être détruit, mais sans détruire l'élément (sol, convoyeur, etc.), et enfin, le rayon doit avoir une durée de vie maximale (en temps ou en distance parcourue, au choix). Ajoutez le script BlasterRay.cs au prefab blasterRayon et implémentez cette fonctionnalité.

## 7 Scripts – The Falling Guy (2 pts)

15. (1 pt) Créez le script FallingGuy.cs pour gérer une éventuelle chute du joueur de la plateforme. Lorsque la chute est détectée, le joueur est replacé immédiatement à son point de départ (point C) sans pénalité au score.

16. (1 pt) Nous avons désactivé les inputs de mouvement à droite et à gauche. Cependant, le joueur peut toujours avancer de côté en direction du convoyeur. Sur le même principe qu'à la question précédente, compléter le script FallingGuy.cs pour que le joueur ne se déplace que sur une ligne avec un Z à +/- 0.01 de la valeur Z de la position de départ. (Ligne jaune sur le schéma page 2)

## 8 Scripts – Score et Affichages (4 pts bonus)

17. (0.5 pt) On souhaite gérer le score du joueur. Le score devra être initialisé à 0 au démarrage du jeu, et on souhaite pouvoir l'incrémenter de 1 point et le décrémenter de 2 points. Créez le script GameManager.cs et ajoutez-le à un GameObject dédié.

18. (1 pt) Ajouter les gameobjects nécessaires à l'affichage du score sur la scène. Le score doit être affiché en permanence dans le coin supérieur gauche de l'écran même si l'on redimensionne la fenêtre de jeu. Il doit être affiché sous la forme « Score = xx ». Utilisez le script ScoreDisplay.cs pour mettre à jour l'affichage du score. Il faudra également associer ce script au bon GameObject de la scène pour qu'il fonctionne.

19. (2.5 pt) Modifiez vos scripts pour rajouter les mises à jour du score suivantes :

- Quand un colis de l'alliance arrive au point B, incrémentez le score de 1.
- Quand un colis de l'empire arrive au point B, décrémentez le score de 2.
- Quand un colis de l'alliance est détruit par le joueur (rayon laser), décrémentez le score de 2.
- Quand un colis de l'empire est détruit par le joueur (rayon laser), incrémentez le score de 1.