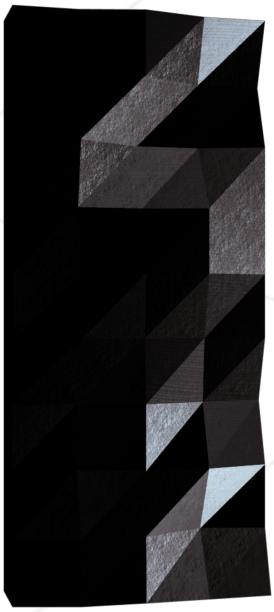

# Piscine Unity - j03

## Les inputs avancés et les GUI 2D

Staff staff@staff.42.fr

*Résumé: Ce document contient le sujet du jour 03 de la piscine Unity de 42.*

# Table des matières

# Chapitre I

# Consignes generales

- La piscine Unity est à faire entièrement et obligatoirement en `C#` uniquement. Pas de `Javascript/Unityscript`, de `Boo` ou autres horreurs.
- L'utilisation de fonctions ou de namespaces non autorises explicitement dans le header des exercices ou dans les regles de la journee sera considéré comme de la triche.
- Contrairement aux autres piscines, chaque journée ne demande pas un dossier `ex00/`, `ex01/`, ..., `exn/`. A la place pour la piscine `Unity`, vous devrez rendre votre dossier projet qui aura pour nom le nom de la journee : `d00/`, `d01/`, .... Toutefois, un dossier de projet contient par defaut un sous-dossiers inutile : le sous-dossier `"projet/Temp/"`. Assurez-vous de ne **JAMAIS** pusher ce dossier dans votre rendu.
- Au cas ou vous vous poseriez la question, il n'y a pas de norme imposée à `42` pour le `C#` pendant cette piscine Unity. Vous pouvez utiliser le style qui vous plaît sans restriction. Mais rappelez-vous qu'un code que votre peer-evaluateur ne peut pas lire est un code qu'elle ou il ne peut noter.
- Vous devez trier les assets de votre projet par dossier. Chaque dossier correspond à un et un seul type d'asset. Par exemple : `"Scripts/"`, `"Scenes/"`, `"Sprites/"`, `"Prefabs/"`, `"Sounds/"`, `"Models/"`, ...
- Assurez-vous de tester attentivement les prototypes fournis chaque jour. Ils vous aideront beaucoup dans la compréhension du sujet et du travail attendu.
- L'utilisation de l'Asset Store d'Unity est interdite. Vous êtes encouragés à utiliser les assets fournis chaque jour (quand nécessaire) ou à en chercher d'autres sur le net s'ils ne vous plaisent pas, sauf bien entendu pour les scripts car vous devez avoir écrit tout ce que vous rendez (hors scripts fournis par le staff, obviously). L'Asset Store est interdit car quasiment tout le travail que vous avez à faire s'y trouve déjà sous une forme ou sous une autre. Néanmoins l'utilisation des Standard Assets de Unity est autorisée voir meme conseillée pour certains exercices.
- Pour les corrections à partir du d03 il vous sera demandé de builder les jeux pour les tester. `C'est le correcteur` qui doit build le jeu vous devez donc évidemment toujours push vos projets/sources. De ce fait votre projet doit correctement configuré pour le build. `Aucun` réglage de dernière minute ne doit être toléré.

- Important : Vous ne serez pas évalués par un programme, sauf si le contraire est explicite dans le sujet. Cela implique donc un certain degré de liberté dans la façon que vous choisissez de faire les exercices. Toutefois, gardez en tête les consignes de chaque exercice, et ne soyez pas FAINÉANTS, vous passeriez à coté de beaucoup de choses intéressantes.
- Ce n'est pas grave d'avoir des fichiers supplémentaires ou inutiles dans votre dossier de rendu. Vous pouvez choisir de séparer votre code en différents fichiers au lieu d'un seul, sauf si le header d'un exercice mentionne explicitement les fichiers à rendre. Un fichier ne doit définir qu'un et un seul comportement, pas de namespaces donc. Toute cette consigne ne s'applique bien evidement pas au sous-dossier `"projet/Temp/"` qui n'a pas le droit d'exister dans vos rendus.
- Lisez le sujet en entier avant de commencer. Vraiment, faîtes-le.
- Le sujet pourra être modifié jusqu'à 4h avant le rendu.
- Meme si le sujet d'un exercice est relativement court, ca vaut le coup de passer un peu de temps à comprendre parfaitement le travail attendu pour le faire au mieux.
- Parfois il vous sera demandé un soin particulier sur la qualité artistique de votre rendu. Dans ce cas, cela sera mentionné explicitement dans le sujet correspondant. N'hésitez alors pas à tester plein de choses différentes pour vous donner une idée des possibilités offertes par Unity.
- Par Odin, par Thor ! Refléchissez !!!

# Chapitre II

# Foreword

Aujourd'hui nous allons jouer à ça.

# Chapitre III

# Exercice 00 : Un simple menu

|  | Exercice : 00 |
|---|---|
| Exercice 00 : Un simple menu | |
| Dossier de rendu : *ex*00/ | |
| Fichiers à rendre : `La scène "ex00" et tout ce qui vous semble utile` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

`N'hésitez pas à tester 5 minutes le prototype de jeu fourni si vous avez des doutes sur le résultat attendu dans un des exercices.`

Créez une scène avec un menu pour notre tower defense. On reste simple ici, un bouton Play ou Start ou Whatever qui lance la partie et un bouton Quit qui ferme l'application.

Le bouton Play doit pointer sur la scène "ex01" qui n'existe pas encore mais que vous créerez dans le prochain exercice.

A vous de trouver un fond sympa et une mise en forme qui donne envie !

`Le programme d'aujourd'hui est particulièrement chargé donc ne perdez pas trop de temps à personnaliser votre menu !`

# Chapitre IV

# Exercice 01 : Drag and drop

|  | Exercice : 01 |
|---|---|
| Exercice 01 : Drag and drop | |
| Dossier de rendu : $ex01/$ | |
| Fichiers à rendre : `La scène "ex01" et tout ce qui vous semble utile` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

`http://docs.unity3d.com/Manual/Layers.html est votre ami ! Aussi bien pour la gestion du drag and drop, que pour éviter que les ennemis/missiles/etc passent sous vos tours, par exemple.`

Créez une barre en bas de l'écran dans laquelle sont visibles les 3 tours basiques que le joueur peut acheter.

Pour acheter une tour il faut la drag and drop depuis la barre jusqu'à l'endroit voulu. Attention à ce que les éléments de l'UI restent bien au premier plan et ne passent pas sous d'autres sprites !

La tour n'est placée que si la case cible est vide et que le joueur a suffisamment d'énergie en réserve. Dans ce cas le coût de la tour est retiré de la réserve d'énergie du joueur.

La barre doit aussi afficher les dégâts, le prix, la portée et le temps de recharge de chaque tour ainsi que la vie et l'énergie du joueur. Vous ne devez pas écrire ces valeurs en dur mais les récupérer depuis le GameManager.

Il faut un feedback visuel - changer la couleur de la tour par exemple - pour savoir en un coup d'oeil s'il y a assez d'énergie dans la réserve pour acheter la tour. On ne doit pas pouvoir déplacer la tour si l'énergie en réserve est insuffisante.

La propriété color est intéressante car on peut la SET mais aussi la GET. On peut donc tester la couleur d'un sprite, ce qui peut s'avérer très pratique dans certains cas. Allez lire la doc pour plus d'infos.

# Chapitre V

# Exercice 02 : Menu pause

|  | Exercice : 02 |
|---|---|
| Exercice 02 : Menu pause | |
| Dossier de rendu : $ex02/$ | |
| Fichiers à rendre : `La même scène "ex01" et tout ce qui vous semble pertinent` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

Vous devez maintenant ajouter un menu pause lorsque le joueur appuie sur la touche échap. Une fonction est déjà prête dans le GameManager pour gérer la pause. Il vous suffit de mettre true ou false en paramètre pour stopper ou reprendre le jeu :

Protoype :

```
public void pause(bool paused);
```

Ce menu doit proposer de reprendre la partie ou de quitter.

Si le joueur quitte, un second menu de confirmation doit apparaître. Si le joueur valide à nouveau il revient sur la scène du menu principal.

Tant qu'on est dans la gestion du temps sachez qu'une fonction est aussi présente dans le GameManager pour accélérer ou ralentir le jeu :

Protoype :

```
public void changeSpeed(float speed);
```

Vous êtes libres de choisir votre implémentation mais il faut au moins un bouton pour accélérer le jeu et un autre pour le ralentir/mettre en pause. Vous gagnerez ainsi un temps monstrueux lors de vos phases de test et de correction. Les fonctions pause

et changeSpeed sont compatibles entres elles donc vous retrouverez automatiquement la bonne vitesse en sortie de pause, même si vous jouiez en accéléré.

Un jeu avec son propre curseur est quand même beaucoup plus stylé. Trouvez donc un moyen de le changer ! Un petit tour dans la doc d'unity devrait règler le problème en 5 minutes.

`Un curseur est fourni avec les sources d'aujourd'hui, mais vous avez le droit de le trouver moche et d'en chercher un autre !`

# Chapitre VI

# Exercice 03 : Rox or sux ?

|  | Exercice : 03 |
|---|---|
| Exercice 03 : Rox or sux ? | |
| Dossier de rendu : $ex03/$ | |
| Fichiers à rendre : `La scène "ex03" et tout ce qui vous semble pertinent` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

Vous devez créer un écran qui affiche le score ainsi que le rang obtenu par le joueur à la fin de la map.

Un score est déjà calculé dans le GameManager donc vous pouvez le récupérer ou créer le votre.

Vous pouvez faire des rangs de F à SSS+ ou trouver des noms un peu plus créatifs ! La seule contrainte est de donner un rang au joueur qui suive une logique en fonction du nombre de points de vie et de l'énergie qui lui reste, avec un minimum de 5 rangs différents.

L'écran doit proposer un bouton pour rejouer si le joueur a perdu, ou pour passer au niveau suivant si le joueur a éliminé toutes les vagues d'ennemis.

Vous devez également créer une nouvelle map avec une difficulté un peu plus élevée.

`Si vous voulez changer la difficulté attention à ne pas changer les caractéristiques des ennemis sur les prefabs d'origine car leurs stats seront modifées sur TOUTES les maps (les préfabs sont indépendants des scènes). Il faut donc changer leurs stats à postériori dans un script ou modifier d'autres paramètres de jeu.`

# Chapitre VII

# Exercice 04 : Encore du travail ?

|  | Exercice : 04 |
|---|---|
| Exercice 04 : Encore du travail ? | |
| Dossier de rendu : $ex04/$ | |
| Fichiers à rendre : `La scène "ex04" et tout ce qui vous semble pertinent` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

`Toutes les upgrades de tours sont déjà préparées dans le dossier préfab afin de gagner du temps.`

Placer des tours c'est sympa mais ça reste assez basique. Maintenant améliorons-les !

Créez un menu radial apparaissant autour d'une tour lorsqu'on clique droit dessus.

Le menu doit proposer d'upgrader la tour. On peut améliorer une tour de niveau 1 au niveau 2 et une de niveau 2 au niveau 3. L'interface doit montrer le coût de l'amélioration au joueur. Bien entendu on ne doit pas pouvoir améliorer la tour si l'énergie en stock est insuffisante. Lorsque la tour est améliorée le coût en énergie correspondant est déduit de la réserve principale.

L'interface radiale propose également de downgrader une tour du niveau 3 au niveau 2 ou du niveau 2 au niveau 1 ainsi que de la vendre si elle est niveau 1. L'interface doit montrer combien d'énergie on peut récupérer lorsqu'on downgrade/vend une tour. A vous de choisir le montant mais en général on récupère la moitié du coût d'achat de la tour.

Un troisième bouton doit tout simplement permettre de fermer le menu, à moins que vous le gèriez autrement, par un clic droit dans le vide par exemple. Vous êtes assez libre dans façon d'implémenter tant que le menu se ferme et ne bug pas.

Vous devez également créer une nouvelle map un peu plus difficile afin de tester ces tourelles et leur possibilité de destruction.

# Chapitre VIII

# Exercice 05 : Pour les progamers

|  | Exercice : 05 |
|---|---|
| Exercice 05 : Pour les progamers | |
| Dossier de rendu : $ex05/$ | |
| Fichiers à rendre : `La scène "ex05" et tout ce qui vous semble pertinent` | |
| Fonctions interdites : `Aucune` | |
| Remarques : `n/a` | |

Il manque une dernière chose pour que le jeu soit finalisé : des raccourcis clavier ! Plutôt que bêtement glisser-déposer les tours une à une ce serait beaucoup plus simple de les sélectionner depuis le clavier.

Créez un système de mappage de touches pour sélectionner les tours. Par exemple lorsqu'on appuie sur "q" ou "1" le curseur de la souris est remplacé par la tourelle canon, il ne reste plus qu'à cliquer sur une zone disponible pour poser la tour.

Dernière touche finale, un sort de destruction d'urgence si un ennemi arrive à se faufiller. Créez une icone de boule de feu dans la barre du bas qu'on peut glisser-déposer n'importe où sur le terrain pour créer une explosion. Il faut également un raccourci clavier pour cette boule de feu, ainsi qu'un cooldown pour que le joueur ne puisse pas l'utiliser en boucle.

Vous trouverez également des images qui n'ont pas été utilisées dans les précédents exercices, comme les gabarits de portée. A vous de jouer avec pour finaliser l'interface et donner un maximum d'informations au joueur afin de pouvoir monter la difficulté et d'avoir un jeu clair et exigeant.

Et n'oubliez pas de faire un écran de fin stylé si jamais un joueur arrive à terminer votre dernier niveau ! Il faut toujours penser à conclure un jeu et pas bêtement retourner sur l'écran titre.

Çette fois-ci il n'y a aucun préfab ou script pré-écrit pour les dernières features à ajouter mais si vous en êtes à ce stade je pense que vous saurez comment faire.