|  |  |
| --- | --- |
| 6C5- TP1  Simulation d’un personnage  Navmesh & animation | |
| **Nom, prénom et DA** | * Nom, Prénom (DA 1234567) |
| **Cours** | **6C5 – Applications de jeux et simulations II** |
| **Enseignant** | Julien Brunet |
| **Date exigée** | Voir Léa |
| **Évaluation** | Ce travail se fait seul |

Table des matières

[À faire individuellement 2](#_Toc163197386)

[Utilisez le Template de projet (6C5-E24-ProjectTemplate) 2](#_Toc163197387)

[Renommez-le « 6C5E24 - TP1 – suivi de vos initiales ») 2](#_Toc163197388)

[Partie 1 2](#_Toc163197389)

[Construire un terrain de jeu où votre personnage évoluera, le terrain doit avoir les éléments suivants : 2](#_Toc163197390)

[De façon autonome et par NavMesh (AI) ajoutez un personnage qui circule entre des buts reconnaissables par un effet de particules. Pour le moment une capsule suffit. 2](#_Toc163197391)

[Ajouter une caméra qui suit le joueur 2](#_Toc163197392)

[Au moins une caméra doit présenter une vue rapprochée qui présente bien l’animation du joueur. 2](#_Toc163197393)

[Configurer une caméra en plongée qui montre l’ensemble du terrain 2](#_Toc163197394)

[Instancier des obstacles qui tombent du ciel ou d’ailleurs et que le personnage doit éviter. 2](#_Toc163197395)

[ Ajouter au moins un obstacle qui se déplace et que le personnage doit contourner. 2](#_Toc163197396)

[Partie 2 : Animation 3](#_Toc163197397)

[Copier la scène de la partie 1 et la renommer 3](#_Toc163197398)

[Choisir un personnage qui vient de la banque mixamo.com 3](#_Toc163197399)

[Soigner les animations : 3](#_Toc163197400)

[Construire un obstacle au-dessous duquel le personnage doit se pencher. 3](#_Toc163197401)

[Ajuster la vitesse du personnage proche des buts et entre les buts 3](#_Toc163197402)

[Évidemment, l’animation doit constamment être ajustée à la vitesse du personnage. 3](#_Toc163197403)

[Ajouter une animation à l’atteinte du but 3](#_Toc163197404)

[Attention à bien organiser vos fichiers ! 4](#_Toc163197405)

[Partie 3 5](#_Toc163197406)

[Performance de votre jeu en terme CPU ou GPU 5](#_Toc163197407)

[En fonction de ce qui a été vu dans le formatif 3 5](#_Toc163197408)

[Performance d’une méthode de votre simulation 5](#_Toc163197409)

[Améliorez les performances de votre jeu 5](#_Toc163197410)

[Faites-moi une petite vidéo du projet réalisé et ajoutez là au projet dans l’onglet vidéo 5](#_Toc163197411)

[Générez le Build Windows 5](#_Toc163197412)

[Générer un build Windows et le mettre à la racine de votre jeu dans le répertoire build. 5](#_Toc163197413)

# À faire individuellement

|  |  |
| --- | --- |
| 0 | Utilisez le Template de projet (6C5-E24-ProjectTemplate)Renommez-le « 6C5E24 - TP1 – suivi de vos initiales ») Utilisez la version 2021.3.22f1 (LTS) de Unity.  Ajoutez vos éléments personnels dans le Canvas de départ  Remplacez la scène du cube par une nouvelle scène vide |
|  | Partie 1 |
| 1.1 | Construire un terrain de jeu où votre personnage évoluera, le terrain doit avoir les éléments suivants :  * Faire au minimum un terrain et une **plateforme** à une hauteur différente du terrain et accessible par au moins trois rampes inclinées de pentes différentes. * Donnez-y un peu de style! C’est le terrain qui sera utilisé dans tout le TP1. Vous pouvez bien sur utiliser des assets existants. |
| 1.2 | De façon autonome et par NavMesh (AI) ajoutez un personnage qui circule entre des buts reconnaissables par un effet de particules. Pour le moment une capsule suffit.  * Il doit y avoir au moins 3 buts. Le personnage doit circuler entre ceux-ci. Après avoir atteint un but, son prochain objectif (un des 2 autres buts s’il y en a que 3) est choisi de façon aléatoire. * La simulation ne s’arrête jamais. * Il n’est pas nécessaire que le personnage touche l’effet de particules ou l’évite. Cependant, il doit passer suffisamment proche. |
| 2.6 | Ajouter une caméra qui suit le joueurAu moins une caméra doit présenter une vue rapprochée qui présente bien l’animation du joueur. |
| 2.7 | Configurer une caméra en plongée qui montre l’ensemble du terrain Au moins une caméra doit présenter l’ensemble du terrain  Permettre de changer de caméra à l’aide de la **touche C** |
| 1.4 | Instancier des obstacles qui tombent du ciel ou d’ailleurs et que le personnage doit éviter.  * Les obstacles roulent sur le terrain et d’une manière ou d’une autre, sortent du terrain. Évidemment, il faut penser à libérer les ressources. * Soigner l’évitement de ces obstacles à l’aide des paramètres du NavMeshObstacle. |
| 1.5 | **Construire un ou des obstacles qui se déplacent :** Ajouter au moins un obstacle qui se déplace et que le personnage doit contourner.  * Soigner l’évitement de ces obstacles à l’aide des paramètres du NavMeshObstacle. |

|  |  |
| --- | --- |
|  | Partie 2 : Animation |
| 2.1 | Copier la scène de la partie 1 et la renommer Dupliquer le bouton permettant d’Accéder à la première scène pour qu’il accède à la deuxième.  Le but est de conserver les fonctionnalités de la première partie mais de remplacer la capsule par un personnage animé |
| 2.1 | Choisir un personnage qui vient de la banque mixamo.com  * Récupérer les animations appropriées (au minimum : Idle, marcher, courir et ramper et toutes autres nécessaires) * Remplacer votre personnage capsule par le nouveau personnage. * Si votre animation du formatif vous satisfait vous pouvez bien sur l’utiliser. * Attention à ne pas copier-coller de code, la gestion des buts et de l’agent de la première partie devrait être réutilisée (idéalement par héritage) |
| 2.2 | Soigner les animations :  * Le personnage doit être sur le terrain et pas en hauteur ou dans le terrain. * Le personnage doit monter proprement sur la plate-forme. * L’animation doit comporter **au moins un blend tree réalisé par vos soins**! Il faut donc laisser le pilotage de la vitesse à l’agent et adapter l’animation. * L’animation doit être le plus fluide possible |
| 2.3 | Construire un obstacle au-dessous duquel le personnage doit se pencher. |
| 2.4 | Ajuster la vitesse du personnage proche des buts et entre les buts  * Proche d’un but, le personnage marche. * Loin des buts, le personnage court. * Attention à ne pas confondre la **speed** de l’agent, qui représente en fait sa vitesse maximale et sa **velocity** qui est sa vitesse effective (un vecteur).  Évidemment, l’animation doit constamment être ajustée à la vitesse du personnage. |
| 2.5 | Ajouter une animation à l’atteinte du but  * Le personnage doit s’arrêter effectuer l’animation et repartir ensuite. * Il peut être judicieux ici d’utiliser un StateMachineBehaviour qui possède des méthodes OnStateEnter, OnStateExit, etc.. |
| 2.8 | Attention à bien organiser vos fichiers ! |

|  |
| --- |
| Partie 3 |
| Performance de votre jeu en terme CPU ou GPU  * Identifiez ici quel est la faiblesse de votre jeu entre le CPU ou GPU et expliquez et appuyez par une ou des copies d’écran circonscrite(s).      * *Mon jeu présente une performance équilibrée entre le CPU et le GPU, comme le montre les scores de 5 pour le GPU et de 4.1 pour le CPU, indiqué dans les captures d'écran fournies. Cela confirme que le jeu fonctionne bien sur les deux processeurs, sans qu'aucun ne soit significativement en retard par rapport à l'autre. De plus, ils ne semblent pas consommer de ressources excessives, ce qui est un avantage supplémentaire en termes d'efficacité et d'optimisation.* |
|  |
| En fonction de ce qui a été vu dans le formatif 3  * Pour votre scène de jeu, indiquez : * Le temps (max et moyen) qu'il faut pour traiter et rendre une image **en FPS :** *47.0 FPS* * Le temps passé par image par le CPU et le par le thread du rendu. : 21*.3ms / 8.4ms* * Le nombre de triangles et d’arêtes dessinés : 60*.5k / 57.1k* * La taille de l'écran ainsi que l'utilisation de la mémoire : *1119x390 -5.0MB* * Et plus…   Utilisez des copies d’écran circonscrites pour présenter ces informations. |
| Performance d’une méthode de votre simulation  * Indiquez quelle méthode CPU **que vous avez écrite** est la moins performante dans votre jeu. Appuyez par une copie d’écran circonscrite.   *CalculComplexCPU();* |
| Améliorez les performances de votre jeu  * En fonction de ce qui a été vu dans le formatif 3, améliorez les performances de votre jeu.   Expliquez ici ces améliorations, et mettez-les en évidences à l’aide de copies d’écran.  Avant amélioration : **GPU Bound** (CPU : 5,423.96ms GPU :77,78ms)  Après amélioration : **Normal** (CPU : 15,37ms GPU : 15,55ms)  Après avoir supprimé la méthode "***CalculComplexCPU()"***, les performances de mon jeu ont été considérablement améliorées. Avant l'amélioration, le jeu présentait des bugs et des réductions dus à une charge de travail excessive sur le CPU. Il était GPU Bound (CPU : 5,423,96 ms, GPU : 77,78 ms). Après l'amélioration, les performances sont revenues à la normale (CPU : 15,37ms, GPU : 15,55ms). Maintenant, le jeu est fluide et réactif, offrant une expérience de jeu beaucoup plus agréable. |
| Remise |
| Faites-moi une petite vidéo du projet réalisé et ajoutez là au projet dans l’onglet vidéo  * Exécutez le jeu pour montrer toutes ses fonctionnalités   + - * Expliquez ce qui se passe en reprenant les consignes etexigences du travail       * S’il y a lieu, indiquez les ajouts par rapport au travail       * Montrer les bogues et les manques |
| Générez le Build WindowsGénérer un build Windows et le mettre à la racine de votre jeu dans le répertoire build. |
| Faites un **export package** dans un fichier portant votre nom et qui identifie votre projet :  Déposez dans le répertoire à votre nom, non zippé , sur le **dépôt** du département : Q:\Commun\!Dépôt\Julien Brunet\6C5E24\TP1:   * **l’export package** * le répertoire Build * l’énoncé complété avec les imprime écrans |