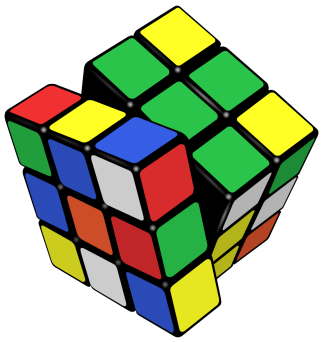
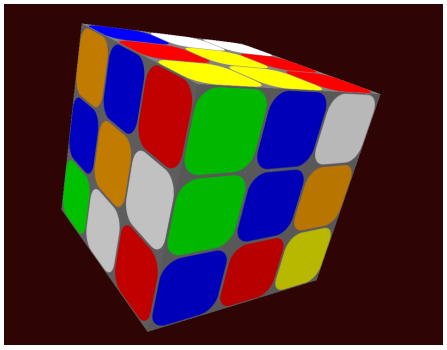
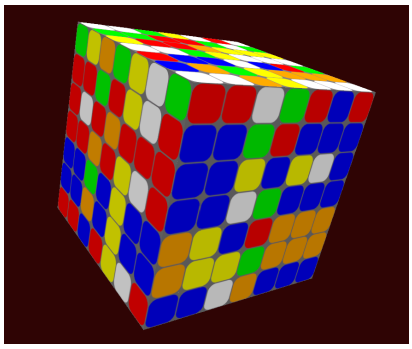
Maths & Unity3D - Quaternion & Plane

Développement d'un jeu du type "Rubik's Cube" ™

A rendre avant le 20/12/2020 à 23h59

Ce TP a pour objectif de vous faire travailler les structures **Quaternion, Plane et Vector3** de l'API Unity3D. Il vous est demandé de développer en binôme, à partir d'un package fourni (le package ne contient que des textures), un jeu du type "Rubik's Cube", dont voici les fonctionnalités et contraintes techniques.

Fonctionnalités de base:

- Les 6 couleurs sont les couleurs traditionnelles du Rubik's Cube, à savoir: blanc,jaune,orange, rouge,bleu et vert (des images vous ont été fournies dans le projet de base)
- Le cube est affiché en 3D au milieu de l'écran,
- Les contrôles du cube s'effectuent à la souris:
    - clic **droit** maintenu + mouvement → rotation du Rubik's Cube selon un axe perpendiculaire au mouvement.
    - clic **gauche** maintenu sur une face du cube + mouvement → rotation de la tranche du Rubik's Cube logiquement impactée par la face cliquée et le mouvement de la souris. Lorsque le bouton gauche est relâché, la tranche en rotation doit rejoindre l'orientation d'équilibre (multiple de 90°) la plus proche.
    - molette: zoom in/out
    - tous les mouvements de transition doivent être fluides et agréables
- Le joueur peut choisir la taille de son Rubik's Cube, c'est-à-dire le nombre de petits cubes sur une arête. Cette taille peut varier entre 2 et 10. Les screenshots ci-dessus présentent des cubes de taille 3 et 7
- Le joueur peut choisir la profondeur du mélange initial du cube.
- UI
    - Le titre du jeu figure dans un coin de l'écran
    - Un slider permet de définir la taille du cube: de 2 à 10

    - Un slider permet de définir la profondeur du mélange initial du cube: de 0 (cube résolu) à 100
    - Un bouton permet de relancer le jeu en prenant en compte les paramètres susmentionnés
    - Lorsque le Rubik's Cube est résolu, un message "Solved" apparaît à l'écran.
    - Un bouton permet d'accéder aux instructions succinctes de navigation
- La configuration courante du jeu (taille, profondeur du mélange, disposition courante des faces, orientation générale) est conservée entre deux sessions de jeu.

Contraintes techniques:

- Utilisez la version de Unity3D préconisée par votre responsable pédagogique,
- Une seule scène Unity3D, nommée *main* et placée dans le répertoire "Scenes".
- A minima un fichier de code C#, nommé *Rubikscube.cs* et placé dans le répertoire "Scripts".
- Le cube doit être généré procéduralement à partir du prefab d'un cube unitaire neutre . Attention la structure de ce cube doit être réfléchie afin de pouvoir d'une part éliminer les faces inutiles et non visibles dans le jeu, mais également permettre une sélection fine des faces à la souris
- Utilisation intensive des structures *Quaternion, Plane* et *Vector3*. Bien évidemment la boîte à outils *Mathf* vous sera tout aussi utile. Les méthodes de transformation de la classe Transform sont proscrites: je pense en particulier aux méthodes de rotation *Rotate* et *RotateAround*.
- Les animations éventuelles doivent être gérées par des coroutines.
- L'UI peut être créée via la méthode OnGUI (cela vous fera un entraînement pour coder ultérieurement les extensions de l'éditeur), ou via un *canvas*.

Votre Rendu

- Une archive .zip ou .rar nommée *GP2_2020_2021_RUBIKSCUBE_<Nom1Nom2>.zip* et comprenant:
    - le projet Unity3D
    - Un build Windows ou Android

- L'intérêt de ce mini projet réside dans la technicité mathématique, mais ne négligez pas pour autant la dimension cosmétique.
- Si le jeu du Rubik's Cube est trivial en apparence, son portage informatique ne l'est pas et représente un challenge mathématique et algorithmique intéressant. N'attendez pas le dernier moment pour vous y atteler.
- Réfléchissez bien en amont à la complexité technique et mathématique induite par ce développement. Ne vous lancez pas dans le développement sans une analyse préalable, anticipez au maximum les problèmes que vous allez rencontrer, sinon vous risquez de devoir modifier la structure de vos objets au gré des changements de stratégie.

- Utilisez vos deux mains pour analyser les orientations relatives des référentiels de votre jeu (Rubik's Cube, petits cubes et caméra) … n'ayez pas peur du ridicule en vous tordant les poignets dans tous les sens.
- Votre code ne devrait a priori pas être trop long, autour 400 lignes
- Les méthodes statiques de *System.linq* pourraient être utiles, mais sans obligation.
- Ce type de développement requiert un Debug éminemment visuel. Affichez en Gizmos / Debug tous les vecteurs nécessaires à la bonne compréhension de vos algorithmes.
- Attention aux arrondis successifs induits par certains calculs séquentiels… si vous n'y prêtez pas attention vos petits cubes risquent fort de prendre la tangente
- Mangez équilibré et ne vous couchez pas trop tard.