UTILISATION D'UN MOTEUR DE DEVELOPPEMENT DE JEUX VIDEO POUR LE DEVELOPPEMENT D'UNE APPLICATION BASIQUE D'IMAGERIE MEDICALE

Hugo Rositi (1), Christophe Lohou (1)

1. ISIT UMR 6284 CNRS/UdA, Equipe Thérapies Guidées par l'Image, Le Puy-en-Velay

Introduction

Le marché des jeux vidéo est devenu le premier secteur économique mondial avec comme conséquence une évolution technologique phénoménale en ce qui concerne les moteurs de développement de jeux. Ces moteurs apportent des solutions performantes, par exemple, pour la visualisation 3D, la parallélisation des traitements, un accès simplifié à la réalité virtuelle… Ils sont par conséquent de plus en plus exploités afin de développer des applications interactives pour d'autres domaines que le jeu, tels que l'imagerie médicale.

Nous avons développé une application exploitant le moteur Unity3D [1] – connu pour son prototypage rapide d'applications interactives. Notre application permet le chargement d'images au format DICOM et propose une segmentation interactive plutôt facile et rapide à mettre en œuvre. L'objectif de ce travail est d'apporter à la communauté un aperçu du potentiel accru et récent d'une telle approche.

Méthodes

Les solutions informatiques dans le domaine de l'imagerie médicale sont pour la plupart développées à l'aide des librairies logicielles de traitement d'images Itk et de visualisation Vtk (plutôt compliquées à prendre en main - langage C++) [2] ; la partie interactive est quant à elle traitée par des logiciels tiers (par ex., Qt [3]). Cet ensemble nécessite, de plus, une configuration informatique non triviale (par ex., Visual Studio [4], et CMake [2]). Des alternatives (par ex., Slicer [5], MITK [6]) - surcouches des librairies Itk et Vtk - permettent des résultats rapides pour des opérations « prédéfinies » (segmentation, recalage) et offrent une certaine interactivité mais nécessitent un effort important pour intégrer de nouvelles extensions. Par ailleurs, ces alternatives souffrent d'un retard d'intégration des dernières librairies de base.

De façon générale, les fonctionnalités proposées par un moteur de développement de jeux vidéo intègrent nativement dans un seul framework un grand ensemble de fonctionnalités évoluées et performantes : graphiques (avec possibilité de traitement d'images, shaders), physique (collisions), réseau, animation (déformations), interface, intelligence artificielle, exports de fichiers vers d'autres logiciels (pour le rendu ou l'animation), branchements de casques de réalité virtuelle, et les deux plus importantes l'interactivité (gestion d'événements) et le scripting (possibilité d'ajouter plutôt facilement des fonctionnalités en langage C#).

En connaissant les fonctionnalités de base du moteur Unity, il a été relativement simple de proposer une solution basique d'imagerie médicale – cf. Fig. 1 : (a) chargement d'une image au format DICOM dans une texture 3D et volume rendering [7], (b) extraction d'un plan de coupe texturé d'orientation arbitraire [7], (c) définition interactive d'une zone d'intérêt (script C#), (d) seuillage interactif (script C#).

Résultats

La figure 1 montre les résultats obtenus.

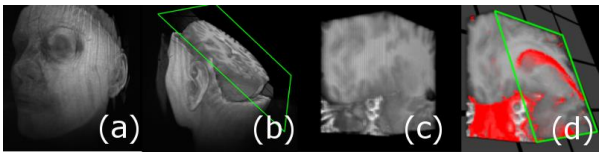

Figure 1: captures d'écran de notre application ; le contour vert représente un plan de coupe.

Discussion

L'approche proposée ici met en avant le côté interactif et natif d'un moteur de développement de jeux vidéo. L'exploitation d'un tel moteur pour la réalisation d'une application d'imagerie médicale semble offrir une plus grande facilité pour les prototypage, maintenance, réutilisabilité et extension d'applications 3D interactives. Il est fort à parier que les solutions logicielles informatiques d'imagerie médicale à venir vont de plus en plus exploiter ces moteurs - gratuits pour la recherche - qui bénéficient d'une avancée technologique à la pointe et apportent des fonctionnalités non encore proposées par les alternatives actuelles (comme l'intégration immédiate de casques de réalité virtuelle).

Références

1. Unity 3D : unity3D.com
2. Itk, Vtk, CMake : kitware.com
3. Qt : qt.io
4. VisualStudio : visualstudio.com
5. Slicer : slicer.org ;
6. Mitk : mitk.org
7. https://github.com/brianasu/unity-ray-marching

Remerciements

Nous remercions Robin Dousse, Youé Graillot et Thomas Bourdon, étudiants en Licence Professionnelle 3D Temps Réel et Réalité Virtuelle, IUT du Puy-en-Velay, pour l'intégration et l'adaptation du shader de ray-marching [7] dans Unity 3D ainsi que pour les captures d'écran.