**Description du projet**

Après avoir découvert l’informatique et l’avoir pratiquée un certain temps je me suis dirigé vers le développement du jeu vidéo. L’étude des différentes technologies du jeu vidéo m’a amené à comprendre que son développement ne peut pas se faire sans outils permettant une création plus rapide et un dialogue plus facile avec les autres membres d’une équipe (développeurs, artistes 2D/3D, scénaristes, etc.). Ce sont les raisons pour lesquelles le développement de SSWE a démarré. SSWE a pour mission de faciliter le développement d’un jeu vidéo quelconque en proposant des outils vitaux, réfléchis, innovants, et d’augmenter la rapidité de création. Un autre atout de SSWE est de proposer un logiciel que toute une équipe peut utiliser afin de centraliser leurs projets.

**Descriptif technique**

* Outils externes :
  + Langage de programmation C++11
  + Utilisation de libraires « open source » :
    - « Irrlicht » en tant que moteur 3D, qui concerne l’affichage à l’écran, les calculs dans l’espace, etc.
    - « LUA » permettant la création de scripts. Autrement dit de créer de petits programmes exécutables au sein de n’importe quel logiciel.
    - « SPARK » en tant que moteur de particules (permet la création de fumée ou de feu par exemple).
    - « Bullet » moteur physique.
    - « OpenAL » lecteur audio simple et dans l’espace.
    - Toutes ces librairies sont compatibles Windows & Mac OS X
  + Développement à l’aide de :
    - Visual Studio 2012 pour Windows
    - XCode pour Mac OS X
  + Tests unitaires à l’aide de CPPUnit.
  + Gestion du code source à l’aide de GIT
* Caractéristiques de SSWE :
  + Création de mondes 3D :
    - Gestion de terrains contenus dans des fichiers 3D externes ou génération directe via la gestion de « height maps ».
    - Gestion de la végétation :
      * Arbres : tant animés que statiques, chargés de façon optimisée soit utilisation des « OctTrees ».
    - Gestion des objets animés tels :
      * Objets quelconques : animation et paramétrage par « key frames » ou manuel
      * Personnages : animation « skeletal »
    - Gestion des lumières :
      * Utilisation des lumières par défaut dans Irrlicht (par point, spot ou directionnelle)
      * Algorithmes plus poussés via l’utilisation des shadow maps. Autrement dit les lumières par défaut d’Irrlicht sont remplacées par des « shadow lights ».
        + Chaque « shadow light » est modifiable
        + Chaque « shadow light » peut avoir une résolution différente
        + Chaque modèle de la scène courante peut avoir un filtre différent
        + Quatre modes pour chaque modèle de la scène courante : CAST, RECEIVE, BOTH, EXCLUDE
        + Chaque « shadow light » peut être recalculée à tout moment par l’utilisateur ou automatiquement
      * Effets de lumière :
        + Un « Lens Flare » paramétrable pour chaque lumière ajoutée à la scène courante. Prise en charge des requêtes d’occlusion pour chaque lumière.
        + Application de lumières volumétriques paramétrable pour chaque lumière ajoutée à la scène courante
    - Gestion des lumières volumétriques :
      * Une lumière volumétrique peut être dissociée d’une lumière
    - Gestion des surfaces d’eau :
      * Une surface d’eau est par défaut un plan sur lequel un effet sera appliqué via un « shader » mais un modèle particulier plutôt qu’un plan peut être chargé via un fichier 3D externe
      * Les surfaces d’eau sont paramétrables ainsi que son « shader » appliqué
    - Tous ces objets que l’on peut ajouter à notre scène courante sont entièrement paramétrables via des éditeurs inclus dans SSWE (Node Editor, Materials Editor).
  + Personnaliser les mondes 3D :
    - Création de matériaux personnalisés au sein de SSWE avec prise en charge de GLSL et HLSL
      * Éditeur de code intégré avec coloration syntaxique
      * Les valeurs modifiables au sein des « shaders » sont paramétrables via des fonctions mathématiques complexes, simplifiées pour l’utilisateur, et mises à jour automatiquement à chaque rendu
      * Prévisualisation de l’effet à appliquer sur différents objets : Cube, Sphère, Plan, ou la scène courante
      * Les matériaux peuvent être exportés sous forme de packages contenant le code de l’effet et les informations relatives
      * Les matériaux peuvent être importés via les packages décris ci-dessus
    - Chaque objet de la scène courante peut être placé rapidement grâce aux outils de placement intuitifs proposés par SSWE (position, rotation, mise à l’échelle)
    - Chaque objet de la scène courante peut contenir différent animateurs. Ce sont des entités invisibles qui donnent un comportement à un modèle de la scène courante
    - Le rendu d’un monde peut être modifié via des filtres visuels contenus dans des fichiers externes et paramétrables via un script LUA. Ces filtres visuels sont également des « shaders » GLSL ou HLSL
    - Le comportement de SSWE peut être personnalisé via la création de scripts LUA :
      * Ces scripts permettent d’ajouter des objets à la scène courante :
        + Charger des modèles 3D externes
        + Ajouter des objets dynamiquement (cube, sphère, etc.)
      * Permettent de modifier les paramètres d’un objet quelconque (position, rotation échelle, matériaux, flags, etc.)
      * Peuvent utiliser le système de fichiers de l’utilisateur, et les fonctions liées au « driver » courant (OpenGL ou Direct3D)
    - Prise en charge d’effets complexes : Motion Blur, SSAO
    - Un manipulateur d’objets intégré :
      * Recalculer un objet avec différentes propriétés non disponibles à l’origine (recalcule des tangentes, du lissage, des normales, etc.)
      * Modifier les méthodes de « mapping » de chaque « mesh buffer »
  + Services SSWE
    - Système de plugins pour les « moniteurs » qui permet de créer son propre rendu. Autrement dit de pouvoir utiliser par exemple des périphériques externes facilement, périphériques comme « l’Oculus Rift » ou faire du multi-moniteur.

Les screenshots sont à venir (je peux pas là =/ et j’ai aucun screenshot qui montre précisément une de ces parties, j’ai que des screenshots qui montrent l’ensemble)